Multimed Tools Appl (2012) 60:97–128
DOI 10.1007/s11042-011-0802-5

Secure interoperable digital content distribution mechanisms in a multi-domain architecture

Lei Lei Win · Tony Thomas · Sabu Emmanuel

Published online: 27 April 2011

Abstract Current DRM systems use the Authorized Domain concept to allow sharing of DRM-enabled multimedia contents across multiple devices. However, some devices in an authorized domain may support only a limited number of DRM systems of the content providers due to their heterogeneous capabilities. Lack of interoperability among DRM systems enforces these devices to stick to a common DRM system which restricts the sharing of different DRM-enabled multimedia contents among them. Most of the current solutions use a translation entity to provide interoperability among different DRM standards with a trust assumption over that entity. This assumption may not assure the content providers that their contents and licenses will be translated and distributed in a secure and legal way. In this paper, we propose a secure interoperable content distribution mechanism for commercial and user generated contents among multiple authorized domains without any trust assumption on the translation entity.

Keywords DRM · Authorized domain · Interoperability · TPM · Personal content sharing

1 Introduction

Digital Rights Management (DRM) technologies have been adopted by multimedia content providers, distributors, and device manufacturers in order to prevent illegal

L. L. Win (✉) · T. Thomas · S. Emmanuel
School of Computer Engineering, Nanyang Technological University, Singapore, Singapore
e-mail: leiwin@ntu.edu.sg

T. Thomas
e-mail: ttony@ntu.edu.sg

S. Emmanuel
e-mail: asemmanuel@ntu.edu.sg

content distribution and to provide authentic digital contents to the consumers. These technologies have focused on protecting the copyrights by binding a content to a device. This restricts a user from accessing a multimedia content item on multiple devices seamlessly. However, a user may be interested in accessing a purchased content item on multiple devices he/she possesses or in sharing that content with his or her friends/family members. This has lead to the concept of *Authorized Domain* [5, 18]. An *Authorized Domain* is a logical collection of devices owned by an individual or a family or a group of users who share the same interest. In an *Authorized Domain*, a digital content item can be shared among the devices belonging to it in a seamless manner while following the rights specified in the usage license of the content.

In order to share DRM enabled content from different *Content Providers* within an *Authorized Domain*, the DRM system of those *Content Providers* need to be installed in all the domain devices. This requires more network resources, and can lead to communication overhead. Further, some of the domain devices may not be able to support all the DRM systems due to the limitations of its storage and processing capabilities. A naive approach to avoid this problem is to make the *Authorized Domain* to stick to a common DRM technology supported by all the domain devices. However, this will restrict the content availability from different *Content Providers* to the *End Devices* due to the lack of interoperability among various DRM systems.

DRM interoperability problems have been addressed by various authors in different ways. MPEG-21 introduced architecture and interfaces between IPMP tools [10]. This approach requires an end user to download and install appropriate DRM tools whenever DRM interoperability is required. However, this mechanism is not suitable for end-devices having less capability or network resources. In [7, 11, 22, 24] a local middle entity functionally situated within a home network does the content and license translation for the home devices. In this approach, the *Content Providers* cannot control the content translation once the license for the content has been issued. Nam et al. [15] proposed an interoperability approach where *End Devices* do the required translation. This method has some weakness because it allows the *End Devices* to translate the content from source DRM format to a neutral format for exporting and from the neutral format to the destination DRM format for importing.

Another approach for interoperability among DRM systems is by using an online [4, 23] third party. Coral DRM [4] proposed a DRM interoperability tool and a framework where an online third party provides a license translation mechanism. Coral splits a normal license transactions into two phases: acquisition of content rights encoded in a DRM-independent rights token; fulfillment of those rights using a native DRM technology. In [23], the authors suggested to use an online brokerage entity which acts as a trusted environment. This entity passes rights management related messages between two different DRM agents of client devices, requests raw digital content to be repackaged from source DRM agent and sends the received raw digital content to the destination DRM agent to do repackaging by the destination DRM agent. However, this approach requires the content providers to keep a trust on the broker that it is not misusing the raw contents or messages. Lee et al. [13]

presented a secure interoperability scheme that allows the content providers to designate a proxy server to perform re-encryption of the content. In this scheme, the content providers cannot control the translation and redistribution actions of the proxy server. A malicious proxy server can share the delegated re-encryption capability with another or can misuse its translation rights. Thus, [4, 13, 23] need the assumption on the existence of a trusted third party (TTP) connected to the network. Moreover, each device requires continuous online connectivity for requesting translations. The mechanisms to assure reliability of the middle entity and to achieve secure and legal distribution of different DRM-enabled contents have not been satisfactorily worked out yet.

In this paper, we propose an interoperability mechanism using an intermediate brokerage system called Local Domain Manager (**LDM**) for a multi-domain architecture comprising of several authorized domains. The *Local Domain Manager* entity need not have to be a trusted party to provide interoperable content distribution and adaptation services to multiple authorized domains. Our scheme prevents any illegal content distribution and translation to other DRM supported formats by the *Local Domain Manager* using cryptographic mechanisms, a TD-license concept and secure key management mechanisms using trusted platform module (TPM). Content providers can control their content translation and distribution by specifying allowable amount of translation and allowable destination DRM systems to which the contents and licenses can be translated.

Nowadays user generated content (UGC) sharing has become very common [2, 14]. User generated contents are the multimedia content created by an individual user or a group of users. Those users may want to share such contents with their local domain members, friends, colleagues and relatives in secure and flexible manner. There exist many papers on secure sharing of user generated contents. In [14] a point to point personal content sharing scheme using smart cards is given. This mechanism incurs extra cost and requires individual devices to equip with a smart card reader. In [2] some solution to share a content using a dynamic domain concept is given. Each domain has an associated domain key. When a user wants to access two shared contents that belong to two different domains, the user is required to join two domains. Thus this approach has a disadvantage that number of the domain keys a user needs to store increases with the number of domains he/she wants to join. Our multi-domain architecture and interoperability mechanism extend to the case of user generated contents also. Our proposed content sharing method allows domain members to share their contents with various flexible sharing models. Users do not need to rely on smart cards or store multiple domain keys to access user generated contents from different domains.

The remainder of the paper is organized as follows. Section 2, reviews the required preliminaries. In Section 3, our multi-domain architecture is described in detail. The proposed mechanisms and protocols for distribution of commercial and user generated contents are given in Sections 4 and 5, respectively. The security analysis and the comparisons are carried out in Section 6. An implementation of the proposed mechanism is given in Section 7. The paper concludes with remarks and future directions in Section 8. A preliminary version of this paper appeared in the '10th Pacific-Rim Conference on Multimedia' (PCM 2009) [29].

2 Preliminaries

2.1 Trusted platform module (TPM)

As per the specifications of the 'Trusted Computing Group' (TCG) [26], a TPM is a tamper resistant module. The TPM contains a set of registers, called Platform Configuration Registers (PCR) containing measurement digests. PCR values are temporal and are reset at system reboot. The only way for a software to change the value of a PCR is by invoking the TPM operation: **Extend**(*index*, *data*).

Let $H(.)$ be the SHA-1 hash function and $||$ be the concatenation operation. When the **Extend** operation is invoked on the TPM, it updates the value of the PCR indicated by the *index* as,

$$PCR_{index} \leftarrow H(PCR_{index}||data).$$

We now describe the main functionalities of a TPM below.

2.1.1 Authenticated boot

TPM measures the state of a platform during the boot process and stores the measurements which are typically based on the executable code involved in each stage of the boot process. Malicious code can be detected because it will cause the measurements to deviate from the expected values.

2.1.2 Remote attestation

Remote attestation is an operation that provides proof of a set of integrity measurements of the terminal platform (host of the TPM) to a remote party. The platform can create reports of its integrity and configuration state that can be verified by a remote verifier. To guarantee the trustworthiness and freshness, the report is signed by the TPM.

2.1.3 Secure storage

User processes can store contents that are encrypted with keys available only to the TPM. The TPM protects all its secret *data* by encrypting them using a non-migratable Storage Root Key (K_{root}) bound to it. The *data* is bounded to the current platform configuration (as defined by the PCRs) via the following TPM operations:

$$\text{Seal}(i_0, i_1, \ldots, data) \rightarrow (C, MAC_{K_{root}}((i_0, PCR_{i_0}), (i_1, PCR_{i_1}) \ldots)),$$

where C is the encryption of *data* with the key K_{root}. The 'Unseal' command takes the ciphertext C and the PCR list created by the 'Seal' command.

$$\text{Unseal}(C, MAC_{K_{root}}((i_0, PCR_{i_0}), (i_1, PCR_{i_1}) \ldots)) \rightarrow data.$$

The TPM verifies the integrity of the list of PCR values, and then compares them against the current values of those PCRs. If they match, the TPM decrypts C and outputs the resulting *data*. If any of the checks fail, the TPM returns an error.

2.2 DRM systems

In general, DRM technologies can be viewed as a group of rules, formats and components which may vary from one DRM system to another. Contents from each DRM systems are protected with their own DRM mechanisms and formats, requiring end users to install different DRM specific playback equipments to use different DRM-protected contents. Some of the current DRM system specific players are Windows media player with the windows media right manager [21] and Real player with the real system media commerce suite [19]. Different DRM systems can be distinguished based on the cryptographic mechanisms, right expression language and content packaging formats used. Any interoperable content distribution mechanism has to address these three issues.

2.2.1 Cryptographic mechanisms

Different content providers encrypt their contents based on some desirable features such as complexity, compression efficiency, perceptibility, format compliance, error resilience, scalability and bandwidth expansion. A comprehensive survey on various encryption mechanisms is given in [12]. If X is a content, a content provider encrypts X using its encryption algorithm $Enc_{CP}(,)$ with the content encryption key $\mathcal{CEK}_X$ and obtains the encrypted content,

$$X_1 = Enc_{CP}(X, \mathcal{CEK}_X).$$

On the other hand, an *End Device* at a client with a specific DRM system may have a different encryption algorithm $Enc_{ED}(,)$ installed in it. If $\mathcal{CEK}_C$ is an encryption key for the encryption algorithm $Enc_{ED}(,)$ the challenge is to provide the encrypted content X_2 to the *End Device* securely instead of X_1 where,

$$X_2 = Enc_{ED}(X, \mathcal{CEK}_C).$$

2.2.2 Right expression language formats

A Rights Expression Language (REL) provides a means to present a formal description of the rights associated with a content. There are various RELs such as MPEG [27], ODRL [16] and XrML [30]. Most of the RELs are based on XML language and looks syntactically similar. However, a DRM agent of a specific DRM system at the client side may not be able to interpret a license written in a different REL corresponding to another DRM system.

2.2.3 Content packaging formats

There are many content packaging standards used in the digital content distribution. For example, OMA DRM [17] uses DCF profile for discrete media files (e.g. still images) and PDCF profiles for continuous media files (e.g. music or video). Motion Picture JPEG Group has adopted Digital Item Declaration (DID) & Digital Item Declaration Language (DIDL) to represent a digital item. A DRM-enabled player from a DRM system may not be able to recognize or use a content with a different content packaging format.

2.3 Content scrambling

Scrambling is a light weight encryption mechanism where digital contents are scrambled to protect the security of the contents and to prevent human visual system or computer vision system from understanding the real meaning of the original digital content. The scrambled media can be recovered by the authorized users using the corresponding unscrambling algorithm and the keys. There exists vast literature on scrambling mechanisms for image, audio and video [3, 8, 25, 31] and industrial standards such as CSS [6] and AACS DRM scheme [1].

In this paper, we propose that the *Content Providers* first scramble their contents using a scrambling algorithm (common to all the content providers) and then encrypts the scrambled content using their encryption algorithm. The unscrambling operation can be performed only by the *End Devices* (by bounding a key with the usage license). The *Content Provider* first scrambles its content X using the scrambling algorithm $Scr(,)$ with the key $\mathcal{SCK}_X$ and encrypts the resultant using its encryption algorithm $Enc_{CP}(,)$ with the content encryption key $\mathcal{CEK}_X$ and obtains,

$$X_1 = Enc_{CP}(Scr(X, \mathcal{SCK}_X), \mathcal{CEK}_X).$$

An *End Device* at a client with a specific DRM system may have a different encryption algorithm $Enc_{ED}(,)$ installed in it. In this case, a middle entity can decrypt X_1 and obtain $Scr(X, \mathcal{SCK}_X)$. The middle entity will not be able to get X as the scrambling key is not available to it. Let $\mathcal{CEK}_C$ be an encryption key for the encryption algorithm $Enc_{ED}(,)$. The middle entity re-encrypts $Scr(X, \mathcal{SCK}_X)$ to obtain X_2 and pass to the *End Device* where,

$$X_2 = Enc_{ED}(Scr(X, \mathcal{SCK}_X), \mathcal{CEK}_C).$$

3 Proposed multi-domain architecture and licenses

In this section, we describe the main components of the proposed multi-domain architecture and the special licenses used.

3.1 Components of the multi-domain architecture

The structure of the proposed multi-domain architecture and the communication flow within the architecture are illustrated in Fig. 1. The various entities involved in the architecture are the following:

1. *Content Providers* (**CP***s*);
2. *Registration Server* (**RS**);
3. *Authorized Domains* (**AD***s*);
4. *Local Domain Manager* (**LDM**);
5. *Log Collection and Analysis Center* (**LCAC**).

There can be many *Content Providers* or owners which use different business rules and DRM formats to provide their contents. Each *Content Provider* is equipped with a *Content Server* (**CS**) and a *License Server* (**LS**).

Registration Server is a TTP (trusted third party) that manages all the entities involved in the architecture. All the entities in the architecture have to register using

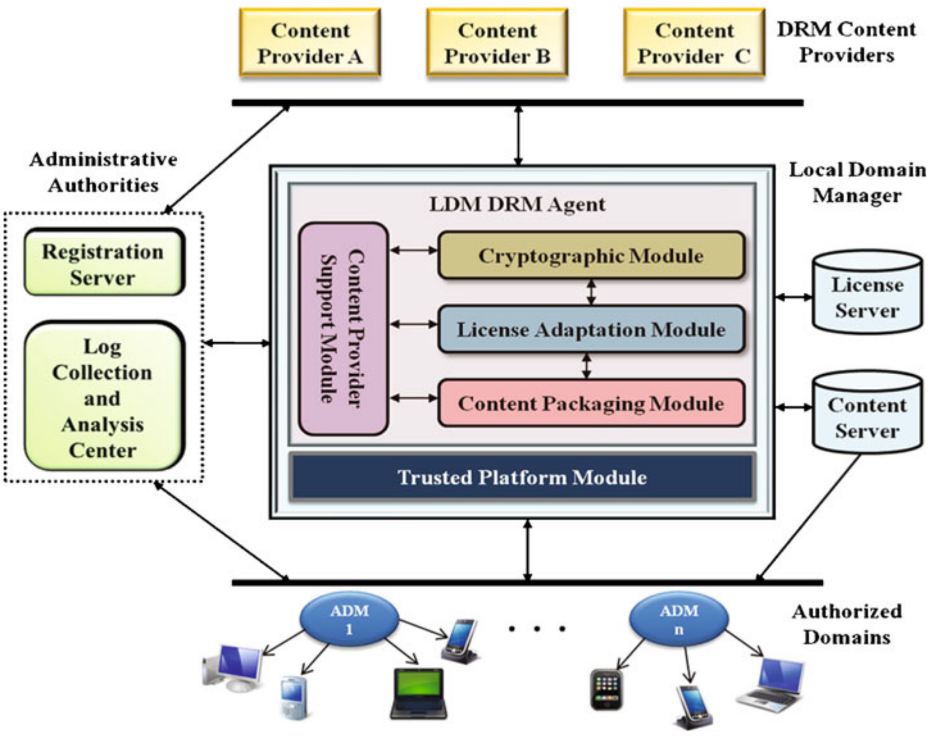


Fig. 1 The proposed multi-domain architecture

their PKI certificate in their first contact with the *Registration Server* and may be in the future. *Registration Server* will issue a certificate to each successfully registered entity and device.

An *Authorized Domain* is a logical group of devices belonging to a department in an organization or a group of users who have similar interest or are members of a family. An *Authorized Domain Manager* (**ADM**) is one of the domain member devices that has a DRM agent DRM_{ADM} installed by the *Registration Server* after successful authentication. DRM_{ADM} is a trusted software that is responsible for managing the domain devices and the content flow within the domain. The *Authorized Domain Manager* device should have good computational power, enough storage capacity and be equipped with a TCG compliant TPM.

A *Local Domain Manager* is a device that provides content distribution and negotiation services to the *Authorized Domains* or *End Devices*. A device used as *Local Domain Manager* should be equipped with a TPM platform. *RegistrationServer* installs a DRM agent DRM_{LDM} in it after successful authentication. DRM_{LDM} is a software agent that has the following four functional modules (see Fig. 1):

1. *License Adaptation Module*;
2. *Cryptographic Module*;
3. *Content Packaging Module*;
4. *Content Provider Support Module*.

License Adaptation Module of the *Local Domain Manager* is responsible for license translation and generation process. Cryptographic Module is to do re-encryption of the contents and adaptation of the content encryption keys as per the destination DRM format. Content Packaging Module performs the change of content container formats. Content Provider Support Module collects, stores and updates the DRM information of each registered *Content Providers*.

A *Local Domain Manager* system has its own *Content Server* (**CS**), and a *License Storage Unit*. The *License Storage Unit* is a secure storage unit of the *Local Domain Manager* where it stores various licenses after encryption. In the *License Storage Unit*, the *Local Domain Manager* securely stores TD-Licenses and U-Licenses (explained in Section 3.2) issued by the content providers and the U-Licenses the *Local Domain Manager* generates for various *Authorized Domain Managers*.

Log Collection and Analysis Center is a trusted third party of all *Content Providers* involved in this architecture that performs collection, storage, and analysis of audit log files to detect violations of licenses by *Local Domain Manager* and *End Devices*. It collects the log files from the *License Server* of each *Content Provider* for its generated licenses, from the *Local Domain Manager* when it distributes and translates U-licenses, and from the *End Devices* (to get their usage patterns) whenever they request licenses to the *Local Domain Manager* via an *Authorized Domain Manager*.

3.2 TD-Licenses and U-Licenses

In our architecture, to allow secure interoperable commercial content distribution, we introduce the notion of a TD-license (Translation and Distribution License). This license enables the *Content Providers* to distribute their contents to as many consumers (with different DRM systems) as possible in a secure and legal manner through the *Local Domain Manager*. *Local Domain Manager* will be bound to the permissions/constraints in the TD-licenses while translating and distributing the contents and license to the end-users. Usage licenses are generated by the *Local Domain Manager* based on the TD-license to allow the consumers to use the content according to the allowed permissions/constraints of the *Content Provider*.

A license usually specifies the information such as content identifier, user information, permissions/constraints, content encryption key(s) and authentication information. A general usage license is expressed in the abstract form as

$$\text{Usage License} = (\mathcal{R}, \mathcal{ID}_C, \mathcal{CEK}),$$

where $\mathcal{R}$ is the usage permissions and constraints (rights), $\mathcal{ID}_C$ is the content identity and $\mathcal{CEK}$ is the key used for encryption of original content [20].

A usage license used in the proposed method is denoted as U-License. A U-License is expressed in the abstract form as

$$\text{U-License} = (\mathcal{UR}, \mathcal{ID}_C, Enc(\mathcal{CEK}, K_{AB}), Enc(\mathcal{SCK}, K_{AB})),$$

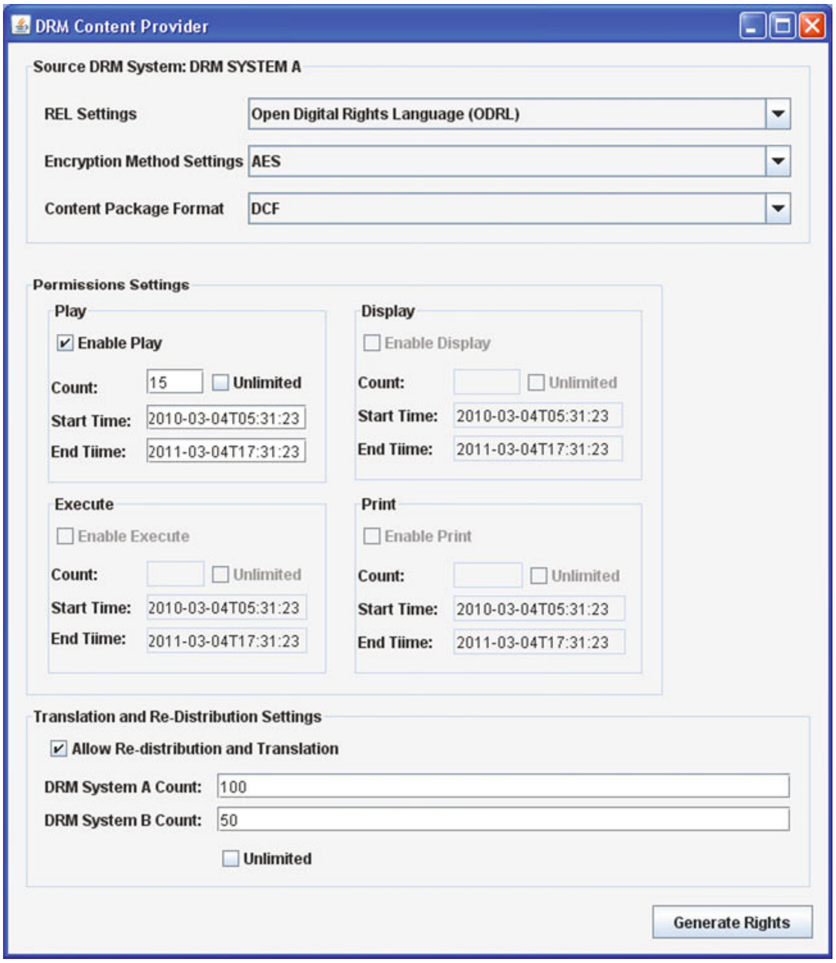
where $\mathcal{UR}$ is the usage rights (permissions and constraints), $\mathcal{ID}_C$ is the ID of the content, $Enc(\mathcal{CEK}, K_{AB})$ and $Enc(\mathcal{SCK}, K_{AB})$ are the encryption of $\mathcal{CEK}$ (Content Encryption Key) and $\mathcal{SCK}$ (Scrambling Key) with a session key K_{AB} shared between A and B. A and B can be either *Content Provider* and *Local Domain Manager* or *Local Domain Manager* and *Authorized Domain Manager* respectively.

A TD-License is expressed in the abstract form as

$$\text{TD-License} = (\mathcal{TR}, \mathcal{DR}, \mathcal{ID}_C, \mathcal{ID}_{CP}, \mathcal{ID}_{LDM})$$

where $\mathcal{TR}$ is the translation rights (permissions and constraints), $\mathcal{DR}$ is the distribution rights (permissions and constraints), $\mathcal{ID}_C$ is the ID of the content C, $\mathcal{ID}_{CP}$ is ID of the *Content Provider* and $\mathcal{ID}_{LDM}$ is the ID of the *Local Domain Manager*. Permissions in $\mathcal{DR}$ allows distribution of U-Licenses and permissions in $\mathcal{TR}$ allows both distribution and translation (into different DRM formats) of U-licenses. Constraints field in $\mathcal{DR}$ shows constraints for distribution count and constraints field in $\mathcal{TR}$ shows constraints for translation count. It contains other information such as validity period of the TD-License and a list of destination DRM systems to which the respective *Content Provider* allows the translation of its content.

TD-License from the implementation of the proposed system (refer to Section 7) is given in the Fig. 2. In this case, a *Content Provider* **A** generates a TD-License with



Source DRM System: DRM SYSTEM A

REL Settings	Open Digital Rights Language (ODRL)
Encryption Method Settings	AES
Content Package Format	DCF

Permissions Settings

Play
[✓] Enable Play
Count: 15 [ ] Unlimited
Start Time: 2010-03-04T05:31:23
End Tiime: 2011-03-04T17:31:23

Display
[ ] Enable Display
Count: [ ] Unlimited
Start Time: 2010-03-04T05:31:23
End Tiime: 2011-03-04T17:31:23

Execute
[ ] Enable Execute
Count: [ ] Unlimited
Start Time: 2010-03-04T05:31:23
End Tiime: 2011-03-04T17:31:23

Print
[ ] Enable Print
Count: [ ] Unlimited
Start Time: 2010-03-04T05:31:23
End Tiime: 2011-03-04T17:31:23

Translation and Re-Distribution Settings
[✓] Allow Re-distribution and Translation
DRM System A Count: 100
DRM System B Count: 50
[ ] Unlimited

Fig. 2 TD-license from our implementation

Table 1 Specifications of **DRM A** and **DRM B**

DRM system	Cryptographic Alg	REL	Content packaging
DRM A	AES	ODRL	DCF
DRM B	3 DES	MPEG-21	DID

distribution and translation rights. The *Content Provider* uses the DRM system **DRM A** and provides the content and usage license in the **DRM A** format. It generates the U-licenses with usage rights for the video after scrambling and encrypting the video. In this case, the TD-License is created for distribution and translation in two DRM systems namely **DRM A** and **DRM B**. The *Content Provider* **A** specifies 100 counts for **DRM A** format and 50 counts for **DRM B** format. This means that a *Local Domain Manager* can distribute 100 copies of the video in the **DRM A** format and 50 copies of the video can be translated into the **DRM B** format for distribution. U-license includes the play permission with constraints of 15 counts and validity period for the usage of the content. The specifications of **DRM A** and **DRM B** are given in the Table 1.

A TD-License is usually sent with a concatenated U-License which is in its source DRM format (i.e., the format of the license supported by the respective *Content Provider*).

4 Interoperable commercial content distribution mechanism

In this section, we describe in detail our mechanisms for distribution of commercial contents in a secure and interoperable manner across devices supporting different DRM systems belonging to multiple *Authorized Domains*. A *Local Domain Manager* is functionally located between different *Content Providers* and different *Authorized Domains*. *Local Domain Manager* performs local content distribution to multiple *Authorized Domains* as well as content negotiation among the different *Content Providers* who are using different DRM systems. The *Local Domain Manager* is not assumed to be a *Trusted Third Party*. The content distribution mechanism involves the following processes:

1. registration process;
2. multi-domain creation process;
3. content and license generation by *Content Providers*;
4. content and license acquisition by *Local Domain Manager*;
5. content and license format adaptations by *Local Domain Manager*;
6. content and license acquisition by *Authorized Domain Managers*;
7. multi-domain modification process.

To make the section not overly long we describe only the core processes (3)–(6) in this section. The remaining processes are described in the appendix. The details are given below.

4.1 Content and license generation by a *Content Provider* (**CP**)

In certain situations, the *Local Domain Manager* will have to decrypt the encryption (with the algorithm of the *Content Provider*) of the content to re-encrypt the content for the destination devices. In such situations, to prevent any possible leakage of the content through the *Local Domain Manager*, the content provider scrambles its content before encryption. Only an *End Device* can descramble the content. The *Content Provider* first scrambles its content X using the scrambling algorithm $Scr(,)$ with the key $\mathcal{SCK}_X$ and obtains,

$$Y = Scr(X, \mathcal{SCK}_X).$$

The scrambled content Y is encrypted using its supported symmetric encryption algorithm $Enc_{CP}(,)$ with the content encryption key $\mathcal{CEK}_X$ and obtains,

$$Z = Enc_{CP}(Y, \mathcal{CEK}_X).$$

Content Provider uploads the encrypted content Z on its content server. The *Content Provider* then generates a U-License (without encrypting $\mathcal{CEK}_X$ and $\mathcal{SCK}_X$) for that content X with its supported RELs format as

$$\text{U-License}_X = (\mathcal{R}_X, \mathcal{ID}_X, \mathcal{CEK}_X, \mathcal{SCK}_X).$$

Content Provider stores this U-License securely and issues to *Local Domain Manager* only after encrypting the $\mathcal{CEK}_X$ and $\mathcal{SCK}_X$. *Content Provider* generates TD-License only upon demand by the *Local Domain Manager*.

4.2 Content and license acquisition by the *Local Domain Manager* (**LDM**)

Local Domain Manager first selects a *Content Provider* (**CP**) from the list of *Content Providers* it has acquired through the protocol described in the Appendix B.1. It then downloads a desired content Z (encrypted and scrambled form of content X) from the available (source DRM-enabled) contents from the content server of the *Content Provider*. *Local Domain Manager* then acquires TD-License and U-License for the content X from the *Content Provider* as follows.

1. **LDM** requests the TD-License and U-License for the content X to **CP**.
2. **LDM** and **CP** perform mutual authentication using the certificates obtained from the *Registration Server* (see Appendix A).
3. **CP** checks the DRM_{LDM} and the platform configuration state of **LDM** using remote attestation protocol given in Section 2.1.2.
4. After successful authentication and attestation, a symmetric session key $\mathcal{K}_{CP,LDM}$ is securely generated by the TPM of the **LDM** and is stored in it. It is then encrypted with the public key of the **CP** and is sent to the **CP**.
5. **CP** decrypts the encrypted $\mathcal{K}_{CP,LDM}$ with its private key. It then encrypts $\mathcal{CEK}_X$ and $\mathcal{SCK}_X$ of the content X with $\mathcal{K}_{CP,LDM}$ and generates the U-License as

$$\text{U-License}_X = (\mathcal{R}_X, \mathcal{ID}_X, Enc(\mathcal{CEK}_X, \mathcal{K}_{CP,LDM}), Enc(\mathcal{SCK}, \mathcal{K}_{CP,LDM})).$$

6. **CP** generates a TD-License for the content X as described in Section 3 as,

$$\text{TD-License}_X = (\mathcal{TR}_X, \mathcal{DR}_X, \mathcal{ID}_X, \mathcal{ID}_{CP}, \mathcal{ID}_{LDM}).$$

7. **CP** concatenates the U-License to the TD-License and encrypts the concatenated licenses with the public key of the **LDM** and sends to the **LDM**.

8. **LDM** decrypts the encrypted concatenated licenses using its private key and obtains the U-License and TD-License and stores them in its *License Storage Unit*.

4.3 Content and license format adaptations by the *Local Domain Manager*

The *Local Domain Manager* has obtained the following encrypted contents and the licenses from the *Content Provider*:

- $Z = Scr(Enc_{CP}(X, \mathcal{CEK}_X), \mathcal{SCK}_X)$.
- TD-License$_X = (\mathbf{TR}_X, \mathbf{DR}_X, \mathcal{ID}_X, \mathcal{ID}_{CP}, \mathcal{ID}_{LDM})$.
- U-License$_X = (\mathbf{R}_X, \mathcal{ID}_X, Enc(\mathcal{CEK}_X, \mathcal{K}_{CP,LDM}), Enc(\mathcal{SCK}, \mathcal{K}_{CP,LDM}))$.

When an *Authorized Domain Manager* requests a content and license in different formats supported by its domain devices that are different from its original format, the *Local Domain Manager* needs to do required translation between those DRM formats based on the TD-License. *Local Domain Manager* and *Authorized Domain Manager* first establishes a symmetric session key $\mathcal{K}_{LDM,ADM}$. Since the keys $\mathcal{K}_{CP,LDM}$ and $\mathcal{K}_{LDM,ADM}$ are shared between the TPMs of *Content Provider* and *Local Domain Manager*, and *Local Domain Manager* and *Authorized Domain Manager* respectively, those keys are stored securely inside the TPMs using sealing mechanism described in Section 2.1.3.

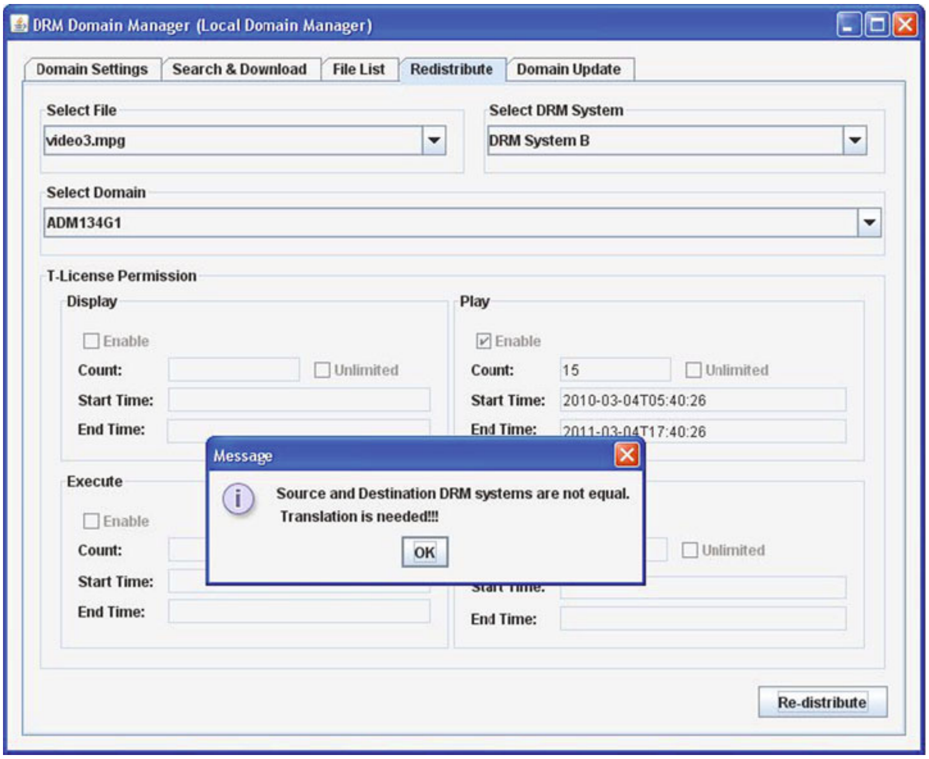
Suppose that a *Content Provider* is using a DRM system DRM_{CP} and an *End Device* **ED** belonging to an *Authorized Domain* is using a DRM system DRM_{ED}. The DRM agent of *Local Domain Manager*, DRM_{LDM} first compares DRM_{CP} with DRM_{ED}. If TD-License allows it carries out the translation and adaptation from DRM_{CP} to DRM_{ED} based on the three criteria mentioned in Section 2.2. A GUI of the process from our implementation (refer to Section 7) is given in the Fig. 3. In this GUI, the administrator of the DRM agent of the *Local Domain Manager* can select the destination DRM system according to the request. When the administrator selects the destination DRM, the DRM agent of the *Local Domain Manager* retrieves the original format of the content and if the source and the destination formats are not the same, it will perform the translation of the content container, encryption and license format in the background. Since the stored secret keys are only released after the platform's software state has been measured and checked, only an authentic DRM agent can access the secret key and can subsequently perform the required translation and distribution. The translation and adaptation are carried out in the order mentioned below.

4.3.1 Re-encryption of content and keys

Suppose that DRM_{CP} uses the content encryption algorithm $Enc_{CP}(,)$ and DRM_{ED} supports the content encryption algorithm $Enc_{ED}(,)$.

Case 1 $Enc_{CP}(,)$ and $Enc_{ED}(,)$ are different.

In this case, to allow the *End Device* to use the content X in DRM_{ED} format, the content needs to be decrypted first and then re-encrypted using the encryption algorithm $Enc_{ED}(,)$. Normally, to change the encryption algorithm, the *Local Domain Manager* needs to decrypt the encryption of *Content Provider*. Since the content has


DRM Domain Manager (Local Domain Manager)

Domain Settings | Search & Download | File List | Redistribute | Domain Update

Select File: video3.mpg

Select DRM System: DRM System B

Select Domain: ADM134G1

T-License Permission

Display
- Enable: unchecked
- Count: (empty) Unlimited: unchecked
- Start Time: (empty)
- End Time: (empty)

Play
- Enable: checked
- Count: 15 Unlimited: unchecked
- Start Time: 2010-03-04T05:40:26
- End Time: 2011-03-04T17:40:26

Execute
- Enable: unchecked
- Count:
- Start Time:
- End Time:

Unlimited: unchecked
Start Time:
End Time:

Message
Source and Destination DRM systems are not equal.
Translation is needed!!!
OK

Re-distribute

Fig. 3 Translation by LDM from our implementation

been scrambled along with the encryption by the *Content Provider* the content will not get leaked even if the *Local Domain Manager* gets compromised.

The *Cryptographic Module* of *Local Domain Manager* gets the encryption algorithm information of DRM_{ED} from the *Content Provider Support Module*. Let $\mathcal{CEK}_X$ denotes the content encryption key corresponding to $Enc_{CP}(,)$ for the content X. The *Cryptographic Module* instructs the TPM to do the following:

– decrypt $Enc(\mathcal{CEK}_X, \mathcal{K}_{CP,LDM})$ using the key $\mathcal{K}_{CP,LDM}$ to get the $\mathcal{CEK}_X$;
– decrypt the encrypted content Z using $\mathcal{CEK}_X$ to get $Scr(X, \mathcal{SCK}_X)$;
– generate a new content encryption key $\mathcal{CEK}_{ED}$ for DRM_{ED} corresponding to the encryption algorithm $Enc_{ED}(,)$.
– compute $Z_{ED} = Enc_{ED}(Scr(X, \mathcal{SCK}_X), \mathcal{CEK}_{ED})$.

Case 2 $Enc_{CP}(,)$ and $Enc_{ED}(,)$ are same.

In this case, the content need not have to be re-encrypted and only the content encryption key and the scrambling key need to be re-encrypted.

DRM_{LDM} instructs the TPM to perform the decryption of $\mathcal{CEK}_X$ and $\mathcal{SCK}_X$ (of U-license) with $\mathcal{K}_{CP,LDM}$, stored in the TPM and re-encrypt them securely inside the TPM module with the key $\mathcal{K}_{LDM,ADM}$.

4.3.2 Different rights expression languages

In our architecture, a module of the DRM of the *Local Domain Manager* called *License Adaptation Module* does the license format conversion from one DRM system to another according to the permissions and constraints for translation specified in the TD-License. If the REL used by the DRM of the *Content Provider* is the same as that of the DRM of the *End Device*, the *License Adaptation Module* only needs to replace the content encryption key and the scrambling key in the original U-License with the re-encrypted content encryption key and scrambling key received from the *Cryptographic Module*.

If the REL used by the DRM of the *Content Provider* and that of the DRM of the *End Device* are different, the *License Adaptation Module* first retrieves the supported REL information of DRM of the *End Device* from the *Content Provider Support Module* (refer to Section 3). It performs license translation process with the REL of DRM of the *End Device* according to the translation constraints such as translation count, allowable destination DRMs and so on. The *License Adaptation Module* then inserts the re-encrypted content encryption key and scrambling key in the new translated license.

The usage licence generated in both cases will be of the form,

$$\text{U-License}_{X,ED} = (\mathcal{R}_X, \mathcal{ID}_X, Enc(\mathcal{CEK}_{ED}, \mathcal{K}_{LDM,ADM}), Enc(\mathcal{SCK}_X, \mathcal{K}_{LDM,ADM})).$$

4.3.3 Different content format

A DRM-enabled player of an *End Device* may not be able to recognize or use a content with a container format of DRM of the *Content Provider* (i.e., different content packaging formats). In this case, the *Local Domain Manager* carries out the content format adaptation by repackaging the encrypted content with the destination DRM supported content container format. Usually *Content Providers* encrypt the contents first and then package the encrypted contents into a supported content container. Therefore, the Content Packaging Module of *Local Domain Manager* can repackage the encrypted content with destination DRM supported content container format without decrypting the media content.

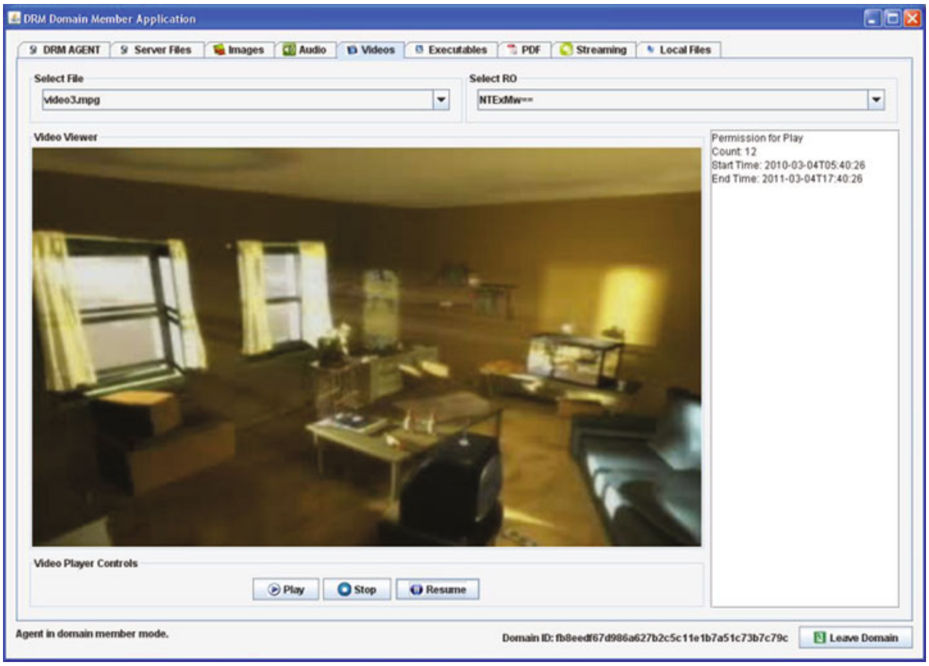
4.4 Content and license acquisition by *Authorized Domain Manager* (**ADM**)

An *End Device* with DRM system DRM_{ED} sends a request to its *Authorized Domain Manager* for a content X it needs to use. The *Authorized Domain Manager* sends a request to the *Local Domain Manager* for the licenses and contents in the DRM formats it needed. *Authorized Domain Manager* acquires the content and license from the *Local Domain Manager* (**LDM**) for a specific DRM system through the protocol given below.

1. **LDM** and **ADM** perform mutual authentication and check the DRM agents and the platform configuration states of each other using the remote attestation protocol given in Section 2.1.2.
2. After successful attestation and authentication, a session key $\mathcal{K}_{LDM,ADM}$ is generated by the TPM of the *Authorized Domain Manager* and is encrypted with the public key of **LDM** and sent to the **LDM**.

3. **ADM** sends a request for the content X and the corresponding U-License with DRM_{ED} format to the **LDM**.
4. **LDM** checks the TD-License and if it has enough re-distribution (same DRM systems)/translation (different DRM systems) counts it performs the license and content format adaptations described in Section 4.3.
5. **LDM** encrypts the U-License with the public key of the **ADM** and sends to the **ADM** along with the encrypted scrambled content.
6. **ADM** first decrypts the U-License with its private key and then decrypts the encrypted content encryption key and scrambling key in the U-License using the key $\mathcal{K}_{LDM,ADM}$ (through the TPM of the **ADM**). **ADM** replaces the encrypted content encryption key and scrambling key in the U-License with the corresponding un-encrypted keys. It then encrypts the new U-license with the domain key and sends to the *End Device* along with the encrypted content.

A GUI from our implementation (refer to Section 7.1) showing the play of the content in the *End Device* is given in the Fig. 4. A domain member can access the protected DRM content in their supported DRM format as shown in the Fig. 4. The domain member first browses the contents in the *Authorized Domain Manager* and downloads the content and license with its supported DRM format. Then the DRM agent at the device of the domain member will handle the playing of the content according to the permission and constraints specified in the U-license.



Select File: video3.mpg

Select RO: NTExMw==

Permission for Play
Count: 12
Start Time: 2010-03-04T05:40:26
End Time: 2011-03-04T17:40:26

Agent in domain member mode.

Domain ID: fb8eedf67d986a627b2c5c11e1b7a51c73b7c79c

Fig. 4 Client device playing the content

5 User generated content (UGC) Sharing

Our architecture supports efficient user generated content sharing among the users within the architecture in a secure way. A user in an *Authorized Domain* may not be able to use any standard DRM system or create its own DRM system to share the contents created by him/her. Hence, to protect his/her content and to prove its ownership, some other security mechanism should be employed. In our proposed method, *Authorized Domain Managers* of all *Authorized Domains* registered for *Local Domain Manager* service will provide a basic DRM system DRM_{UGC} to its domain members. Therefore, user generated content can be protected and securely shared with the domain devices that have DRM_{UGC} installed. We consider two content sharing models, viz intra-domain sharing and inter-domain sharing. Intra-domain sharing is sharing of a user generated content among the domain members of an *Authorized Domain*, whereas inter-domain sharing refers to sharing of a user generated content from an *Authorized Domain* to other *Authorized Domains* under the same *Local Domain Manager*.

5.1 Intra-domain sharing

User generated contents to be shared within an *Authorized Domain* are uploaded on the content server of the *Authorized Domain Manager*. This server can be connected locally or remotely by any domain member using Universal Plug-n-Play (UPnP) Audio/Video (AV) control point or via Virtual Private Network (VPN) to browse the content.

5.1.1 Content encryption, uploading and license creation

When a *User* from an *Authorized Domain* wants to share his/her content X, he/she needs to perform the following protocols with the *Authorized Domain Manager* (**ADM**) of that *Authorized Domain*. In the following, we denote the public-key pair of the *User* as $(\mathcal{PK}_U, \mathcal{SK}_U)$.

1. *User* and **ADM** performs mutual authentication using Domain User Certificate and **ADM** Certificate.
2. DRM of **ADM** generates a session key $\mathcal{K}_U$ and securely sends to the *User* after encrypting with $\mathcal{PK}_U$.
3. *User* encrypts the content X and the set of permissions and constraints (rights) for its use with the key $\mathcal{K}_U$ and sends to **ADM**.
4. DRM of **ADM** decrypts the encrypted content using $\mathcal{K}_U$ and obtains X.
5. To detect duplication of a content, DRM of **ADM** computes the hash value of m (m is a sufficiently large positive integer) parts of the content X separately and stores in its content server. **ADM** compares each hash value of the content with the corresponding hash value of the already registered contents in its content server. If the hash values of at least one part of any two contents match, the **ADM** rejects the content and notifies the *User*.
6. If the hash verification is successful, **ADM** assigns a unique content identifier ID_X to the content X and encrypts the content with a symmetric content encryption key $\mathcal{CEK}_{U,X}$.

7. **ADM** creates a usage license with the rights specified by the *User* and $\mathcal{CEK}_{U,X}$. It then encrypts the usage license with the session key $\mathcal{K}_U$.
8. **ADM** generates a *Content Ownership Certificate* with the ID of the content and ID of the *User*.
9. **ADM** uploads the encrypted content on its **CS** and sends the *Content Ownership Certificate* and the encrypted usage license to the *User*.
10. *User* decrypts the usage license with the session key $\mathcal{K}_U$ and stores the *Content Ownership Certificate* and the usage license securely.

5.1.2 Accessing content by domain members

Intra-domain sharing is a protocol involving the three entities *Authorized Domain Manager*, a *Domain Member* and the *Owner* of the content. The details are as follows.

1. *Domain Member* browses the content server of the *Authorized Domain Manager* and downloads a desired content X and requests the license for that content to the *Authorized Domain Manager*.
2. *Authorized Domain Manager* and *Domain Member* perform mutual authentication using the domain certificate and the *Authorized Domain Manager* certificate.
3. *Authorized Domain Manager* forwards the *Domain Member* to the *Owner* of the content with enable signal.
4. *Owner* sends the usage license to the *Domain Member* after encryption with the public key of the *Domain Member* if it is willing to share the content with the *Domain Member*.

5.2 Inter-domain sharing

In this case, the *Owner* of a content may wish to share his/her content with the domain members of other *Authorized Domain*s under the same *Local Domain Manager* for the sake of social networking or advertisement. A possible scenario of that kind of sharing of user generated content is as follows. Let a *Local Domain Manager* manages an apartment building, where *Authorized Domain*s are the individual household units in the apartment building. Alice who is a member of an *Authorized Domain* wishes to share her marriage video to her friends and relatives in the apartment building. Our inter-domain sharing mechanism gives a simple efficient mechanism with less communication complexity for sharing such contents with no additional key establishment/exchange.

Suppose that a *User* from an *Authorized Domain* wants to share his/her content X, across a set of other *Authorized Domains*. In this case X will have to be uploaded on the content server of the *Local Domain Manager* by the *Authorized Domain Manager* of the *Authorized Domain* of the *User*.

The *User* performs a protocol similar to that given in Section 5.1.1 with the *Authorized Domain Manager* of the *User*. In this case, the only difference will be that the *User* will request its *Authorized Domain Manager* to generate a TD-License for the content X along with the U-License. The TD-License will specify the allowed destination *Authorized Domains* and translation/distribution counts of the content.

Once the *Authorized Domain Manager* of the *User* has uploaded the encrypted content on its content server and generated the U-License and TD-License as per the requirement of the *User*, it can assume the role of a *Content Provider* as in the architecture given in Fig. 1. Now the *Local Domain Manager* and other *Authorized Domains* can get the content and licenses exactly through the protocols described in Sections 4.2–4.4.

5.3 Advantages of the proposed user generated content sharing mechanism

Compared to the user generated content sharing schemes [14] and [2], the proposed scheme requires neither the users to hold multiple domain keys to access the shared contents of various domains nor additional hardware devices such as smart cards to share its contents. With the proposed method, the users can share their contents with many other authorized domains simultaneously, thereby saving communication and computation overheads for joining multiple domains and performing additional key establishments.

6 Security analysis and comparisons

In this section, we carry out the security analysis of the proposed system and the performance comparison with other related systems.

6.1 Security analysis

The three kinds of entities involved in our architecture are *Content Provides*, *Local Domain Manager* and *Authorized Domains*. In our architecture, the middle entity (*Local Domain Manager*) does not need to be a trusted third party unlike in other architectures. Thus, the possible adversaries in this model are the *Local Domain Manager* and the various *Authorized Domains*. The major security issues are the following:

– illegal content translation and distribution by the *Local Domain Manager*;
– illegal content flow from an *Authorized Domain* to outside of the domain.

The focus of the paper was on protecting a *Content Provider* against an untrusted *Local Domain Manager* which does distribution and translation services for its contents. In Section 4.3, we described the protocols which can protect a *Content Provider* against an untrusted *Local Domain Manager*. We now do an analysis on how the proposed architecture and the mechanisms address the various security requirements in interoperable commercial content distribution. The discussion for user generated content sharing is similar and hence we leave out the details.

The security of the *Local Domain Manager* and the *Authorized Domains* is based on the Trusted Platform Module (TPM) residing in them. A *Local Domain Manager* with the TPM can guarantee a secure content and license adaptation by an authentic DRM agent of the *Local Domain Manager*. The translation of content from the source to destination DRM systems are conducted according to the permission and constraints specified in the TD-license which is enforced by the DRM agent of the *Local Domain Manager*. Since the enforcement of TD-licenses is executed on the side of the *Local Domain Manager*, an attacker can deactivate this enforcement by modifying the DRM software of the *Local Domain Manager*. To avoid this kind of attacks, we used the *Remote Attestation Protocol* of the TPM to attest the status of the platform of the *Local Domain Manager* before sending any sensitive data to it. According to the TCG specification, TPM measures all the software components and stores the corresponding hash-value in the *Platform Configuration Registers* (PCRs) by extending the previous value of a specific PCR by the measurement value (MV) using the SHA1 hash function as:

$$Extend(PCR_N, MV) = \text{SHA1}(PCR_N || MV).$$

TPM creates an event for every measured component and stores it in a *Stored Measurement Log* (SML). In the remote attestation protocol, the PCR values signed by a non-migratable TPM signing key, the Attestation Identity Key (AIK) is sent together with the SML to attest the platforms state to a remote party. The remote party verifies these values with reference values to see whether the platform is in a trustworthy state or not. TPM's seal storage feature is used to avoid accessing of sensitive data (content encryption key) by a compromised DRM agent of the *Local Domain Manager*. TPM binds the data to a certain platform configuration (PCR values and SML) without directly transferring them. The stored secrets are only released if the platform's software state is identical to the previous state.

We now show that the proposed system has all the desirable security features of a good interoperable content distribution system.

1. **Prevention of Translation and Distribution by a Malicious** *Local Domain Manager*: A *Content Provider* should be able to make sure the authenticity and integrity of the translation entity. In the proposed system, in order to perform the translation and the distribution of DRM contents and licenses, a *Local Domain Manager* first needs to request a TD-license. A TD-license for a content can only be available to the entity if the DRM agent and the platform configuration state of the entity has been successfully authenticated by the *Content Provider* using the remote attestation feature of the TPM. Therefore, it is not possible for a malicious *Local Domain Manager* (with manipulated DRM agent) to get the TD-license and to perform translation and redistribution.
2. **Prevention of Excessive Translations and Distributions**: A *Content Provider* should be able to control the amount of translation as well as distribution of the translated contents by a *Local Domain Manager*.
   In our system, a *Content Provider* can specify the amount of translation and distribution that can be performed by the *Local Domain Manager*. Once a TD-License and the corresponding U-License have been acquired by the DRM agent of the *Local Domain Manager* after successful authentication, distribution and translation are enforced by this DRM agent according to the permissions and constraints in the TD-License. Therefore, by using TD-License a *Content Providers* can control the translation and distribution of the content and U-Licenses even if the *Local Domain Manager* is untrusted.
3. **Allow Specifications of Destination DRM Systems**: The system should allow a *Content Provider* to specify the desired or allowable destination DRM systems to which its contents and licenses can be translated.
   In our scheme, a *Content Provider* can specify the desired destination DRM systems to which its contents and licenses can be translated by the *Local Domain Manager*. Once a TD-License and the corresponding U-License have

been acquired by the DRM agent of the *Local Domain Manager*, distribution and translation are enforced by this DRM agent according to the permissions and constraints in the TD-License. Therefore, by using TD-License a *Content Provider* can control the desired destination DRM systems to which its contents and licenses can be translated.

4. **Secure Content Adaptation and Distribution**: The following security concerns of a *Content Provider* when allowing a *Local Domain Manager* to do required adaptations and distributions should be satisfied.

   – the entity should not have access to the unencrypted content (neutral format/raw data) and the content encryption keys.
   – the adaptations should be done securely without leaking the plaintext of the content and the content encryption keys.
   – the original/adapted contents and licenses should be distributed legally.

   In the proposed system, the adaptation process between different DRM formats and the distribution of the translated or original DRM-enabled contents and licenses are carried out without leaking the content and the content encryption key. This is because the TPM module unseals the key to access content encryption key and the scrambling key only when the PCR value of the platform remains the same as it was at the time of sealing. In the case of re-encryption of the content by the *Local Domain Manager* the scrambling operation performed by the *Content Provider* on its encrypted content ensures that a manipulated DRM agent of the *Local Domain Manager* or a malicious intruder cannot access the content as the scrambling key SCK is not available to that DRM agent/intruder.
5. **No trusted** *Local Domain Manager*: A *Content Provider* should not be required to trust a *Local Domain Manager* in order to allow it to perform the required content and license adaptations.
   In our architecture, the middle entity *Local Domain Manager* need not have to be a trusted third party unlike in other architectures. The *Local Domain Manager* does not perform content and license negotiation under the assumption that it is trusted. We provided mechanisms to assure the reliability of the middle entity to achieve secure and legal distribution of different DRM-enabled contents.
6. **Secure Sharing of a Content within an** *Authorized Domain*: Illegal content flow to the outside from an *Authorized Domain* and from an *End Device* belonging to the *Authorized Domain* should be controlled. Further, the usage of a content within an *Authorized Domain* should be strictly as per the license.
   In the proposed scheme, an *Authorized Domain Manager* gets encrypted contents and licences from the *Local Domain Manager* and distributes to the devices in its domain through the protocol described in Section 4.4. Illegal content flow to the outside of an *Authorized Domain* from an *Authorized Domain Manager* is controlled by the DRM_{ADM} in the *Authorized Domain Manager*. Similarly, illegal content flow to the outside of *Authorized Domain* from an *End Device* **ED** belonging to the *Authorized Domain* is controlled by the DRM_{ED} in the *End Device*. The mechanisms for ensuring content security in *Authorized Domain Manager* or *End Device* beyond the control of the corresponding DRM agents involves detection of U-License violations through collection and analysis of

the log files and/or detection of unauthorized copies through watermarking mechanisms. This discussion is beyond the scope of the paper.

6.2 Comparisons and discussions

In this section, we begin the discussions with an analysis of some well known interoperable content distribution mechanisms [4, 7, 13, 15, 23] (see Table 2).

The interoperability scheme of Lee et al. [13] allows a *Content Provider* to designate a proxy server to perform re-encryption of the content. In this scheme, the *Content Provider* cannot control the translation and redistribution actions of the proxy server if the DRM agent at the proxy server is compromised by a malicious user after getting the PKI certificate. Since a compromised proxy can still present its public key, a *Content Provider* cannot know whether the DRM agent of the proxy server is authentic or not. Their system do not provide any mechanism to prevent the translation and the distribution by a malicious translation entity in this situation. A compromised DRM agent at the proxy server can also share the delegated re-encryption capability with another or can misuse its translation rights. Their scheme needs a trust assumption that the proxy server will not share the re-encryption key under any situation. Moreover, a *Content Provider* cannot provide the translation and distribution rights to the proxy server directly for multiple distribution and translation in a controlled manner. Their method does not allow a *Content Provider* to specify the desired destination DRM systems. A *Content Provider* has to distribute the content with a neutral format which is not a desirable option. Regarding the efficient usage of interoperable contents, an *End Device* needs online connectivity to get a license. License transmission protocol includes nine rounds of communication between different entities. Further, their method does not provide a solution to perform adaptation between different source and destination DRM formats. Instead, they require the source content to be in a neutral format and the destination content to be in a general format.

The Coral Framework Architecture [4] defines some 30 roles that represent the various functions that are active in a typical interoperability transaction. Coral claims that these roles facilitate the management and exchange of 'Rights Tokens' and the generation of DRM-specific licenses from them. Further, the communications happens over trusted interfaces. Trust in a Coral context has the following assumptions:

– roles prove to one another 'they are who they say they are' and that they are trustworthy;

Table 2 Comparison with various schemes

Functionalities	[13]	[23]	[15]	[7]	[4]	Our scheme
Prevention of translation and distribution by a malicious translation entity	N	N	N	N	N	Y
Prevention of excessive translations and distributions	N	N	N	N	Y	Y
Allow specifications of destination DRM systems	N	N	N	N	N	Y
Secure content adaptation and distribution	Y	N	N	N	NC	Y
No trusted translation entity	N	N	N	N	N	Y
Efficient usage of interoperable contents	N	N	N	N	Y	Y

- integrity of messages are safeguarded;
- roles can trust that messages are received from the senders intended;
- confidentiality of messages are safeguarded;
- messages can be protected against eavesdropping.

A device or a system that implements the Coral License Issuer role is responsible for converting messages from the Coral Framework into specific requests to generate DRM licenses from the 'Rights Tokens', and must be integrated with a native license server. The framework architecture does not specify the nature of the integration; instead, it focuses on the interface contract between the DRM-integrated system and the rest of the Coral Framework [4]. The integration imposes a great challenge to the *Content Providers* who do not want to associate their native DRM-License Server with a third party server. For content adaptation, Coral involves 'Content Handling Roles' to provide DRM packaging functionality. However, Coral does not consider how to provide secure access to the content in order to do the content adaptations. Coral does not consider how to prevent illegal translation (derivation in Coral) and distribution by a malicious entity if the Coral License Issuer role has been attacked by a malicious entity. Coral uses the 'Rights Tokens' to allow the native DRM license server to generate the required native DRM-licenses. Therefore they provide mechanisms to prevent excessive translations (derivation) and distributions of licenses, provided that the Coral License Issuer role is a trusted role to deliver correct messages from the Coral framework to the native license server. However, Coral does not allow a *Content Provider* to specify the desired destination DRM systems to which its contents and licenses can be derived or adapted. To evaluate the efficiency of devices to use interoperable contents, we need to consider the network, time delay to get the license and storage resources required. According to the protocols provided in Coral, the devices requesting license and/or requesting license transfer to other device need to be online in the license acquisition transaction. Therefore, the devices require its own network and storage resources for the license acquisition process. License acquisition transaction involves communication between domain device(s) and different Coral roles (Right Locker, Right Mediators, Domain Manager and License Issuer). Licenses are generated only after the Rights Mediator (which is online or local to domain devices) performs 'Rights Tokens' and 'Domain Membership' queries with the Rights Locker and the Domain Manager respectively.

Serrao et al. [23] uses an brokerage entity called 'iRMBroker' which acts as a middle entity that routes the translation requests to the appropriate rights management regimes that are able to access, produce and manipulate the digital object. This entity passes rights management related messages between two different DRM agents of client devices. It requests raw digital content from source DRM agent and sends the received raw digital content to the destination DRM agent to do repackaging by the destination DRM agent. [7, 15] proposed a DRM interoperability solution using a local translation entity. In the above schemes [7, 15, 23], a *Content Provider* has to assume the translation or brokerage entity as in trusted environment since the entity has open access to the digital content. These schemes do not provide a mechanism that provides secure content adaptation and distribution. Therefore, there is no prevention of illegal content distribution if the entity is compromised or malicious. Consequently, these schemes cannot prevent illegal translation and distribution by a malicious entity and do not support specification of destination DRM systems.

We now give a comparison between the proposed approach of *Local Domain Manager* with TPM and other approaches using brokers and middle entities.

Comparison between the proposed approach of "LDM with TPM" and other approaches using brokers and middle entities

The main difference between the proposed approach of *Local Domain Manager* with TPM and other approaches using brokers, middle entities and DRM interoperability tools is that a *Local Domain Manager* with TPM can guarantee secure content and license adaptation by an authentic LDM DRM agent. By using an LDM DRM agent with TPM, the proposed system has the following advantages over the others.

1. Our proposed system prevents man-in-the-middle attack as the key shared between the *Content Provider–Local Domain Manager* or the *Local Domain Manager–Authorized Domain Manager* are generated by the TPM and is sent by wrapping with the public key of the other entity only after a successful remote attestation protocol.
2. A *Local Domain Manager* with a manipulated DRM agent will not be able to establish a shared session key with the server since its remote attestation will fail.
3. When a malicious user gets access to a *Local Domain Manager*, he may try to manipulate the DRM agent of the *Local Domain Manager*. This is because DRM agents are responsible for enforcing permissions and constraints associated with a DRM content as well as controlling access to the DRM content. However, even if a malicious user succeeds in the manipulation of a DRM agent, the TPM will not allow the DRM agent to access the key which is bounded with DRM agent's PCR value (current platform software state) as the manipulated DRM agent's PCR value will be different from the expected PCR value.

We now discuss the strengths and weaknesses of the proposed system.

Strengths of the proposed system

1. It supports both personal and commercial content sharing.
2. It can prevent content and license sharing with illegal devices. First, only the legitimate DRM agent of an *Authorized Domain Manager* which has the expected PCR value can access the shared session key and redistribute it among the domain devices. To allow only the legal domain members to access the domain contents, licenses are wrapped with the last updated domain key.
3. It can support devices with less capability to get a license as the proposed method does not require continuous online connection and high storage capacity for an *End Device*.
4. It allows the devices to get interoperable contents without being online.
5. It allows, the *Content Providers* to distribute their contents to devices regardless of their underlying DRM in a large scale.
6. It allows *End Devices* in a domain to access the different DRM-enabled contents from different content providers without installing multiple DRM agents or any additional tools or software.

7. The *End Devices* can be directly transfered across *Authorized Domains* with a simpler transfer domain protocol compared to the complex 'leave domain and a join domain' in other systems [17].

Weakness in the proposed system

1. The proposed system requires a dedicated device in an *Authorized Domain* to function as an *Authorized Domain Manager*.
2. The proposed system assumes that all the *Content Providers* use a common scrambling algorithm (in addition to their specific content encryption algorithm) and all the *End Devices* have the corresponding unscrambler installed in it.

The comparison of the proposed scheme with various other schemes based on the security attributes discussed in the Section 6.1 is given in the Table 2. In the table 'Y' denotes 'Yes/Supported', 'N' denotes 'No/Not Supported' and 'NC' denotes 'Not Considered'. As is evident from the table the proposed schemes achieves all the desirable properties of a good interoperable content distribution scheme.

7 Implementation and application scenarios

In this section, we describe an implementation and application scenarios of the proposed system. A video demonstration of the working of the proposed system can be found online [28].

7.1 Implementation

The implementation is done using Java^{TM} JDK 1.6 with the aid of Netbeans IDE 6.5. Java^{TM} has been chosen as the base language because it is supported in multiple operating systems such as Microsoft Windows and most Linux distributions.

Apache Tomcat 6.5 is used as the web server for the server components such as the **CS** and **LS** of *Content Provider*s, *Registration Server*, *Local Domain Manager* and *Authorized Domain Manager*s. Tomcat is a Java^{TM} Servlet and Java^{TM} Server Pages (JSP) container, which implements specifications defined by Sun Java^{TM} and processes logics written in Java^{TM} language. Microsoft SQL Server 2005 is also used by the server components as the database to store system information. Relational database techniques are used when implementing the tables to improve searching complexity. Graphical User Interfaces (GUIs) of our prototype system are created and designed using Java^{TM} Swing toolkit in Netbeans to provide smooth interaction between the users and the components.

The main entity of our architecture, the *Registration Server* stores the information of all the registered devices and domains by SQL statements in the *Registration Server*. This provides scalability as only the stored procedure requires modification if the database schema has to be changed. In the registration process, information is exchanged between *Registration Server* and the registering entity in XML format via the HTTP protocol.

The performance of the implementation of the proposed system for an image and a video is given in Table 3. The *Content Provider* (CP A) uses the AES encryption

Table 3 Performance of the system

Function/Process	Time (in ms)	
	Image (221 KB)	Video (9.90 MB)
AES Encryption and ODF packaging time at CP A	44	294
3DES Encryption and MPEG 21 packaging time at CP B	324	2,100
Scrambling at CP	718	906
Change of encryption algorithm from AES to 3DES at LDM	343	2,423
Change of encryption algorithm from 3DES to AES at LDM	108	1,366
Change of content container format (DID → ODF) at LDM	278	2,571
Change of content container format (ODF → DID) at LDM	293	2,669
Change of License format (MPEG21 → ODRL) at LDM	186	197
Change of License format (ODRL → MPEG21) at LDM	154	168

algorithm and the ODF container format whereas the *Content Provider* (CP B) uses the 3DES encryption algorithm and MPEG21 DID content container format. The *Content Providers* use the format-compliant content scrambling algorithm of Kiaei et al. [9]. We measured the time taken to perform the following for a sample image and a video: change of content encryption algorithm from AES to 3DES and from 3DES to AES; change of content container format from DID to ODF and from ODF to DID; change of license format from MPEG21 REL to ODRL REL and from ODRL REL to MPEG21 REL. The system parameters of the *Content Providers*, *Registration Server*, *Local Domain Manager* and *Authorized Domain Managers* are: Processor: Intel(R) Core(TM)2 Quad CPU 2.67GHz, Memory(RAM): 8.00GB, System type: 32-bit Operating system and OS: Window Vista Ultimate. The system parameters of the *End Devices* are: Processor: Intel(R) Core(TM)2 CPU 2.67GHz, Memory(RAM): 2.00GB, System type: 32-bit Operating system and OS: Window Vista Business.

7.2 Application scenarios

The proposed system is designed in such a way that it can support a variety of applications where interoperability of different DRM systems is required. Our system can be used in apartment buildings and collaborative work environments where there are many groups of devices that want to share multimedia contents in a legal and controlled manner. We now describe these application scenarios below.

Scenario 1: apartment block environment

In an apartment block (or set of blocks) setting, a multi-domain environment can be setup using the protocols given in the Section 4 and the Appendix. A *Registration Server* can be setup for the apartment blocks in the city or the county. A *Local Domain Manager* for the apartment block can be setup by registering it to the *Registration Server* and binding it with the apartment block number. Devices (along with *Authorized Domain Manager* and domain devices) in each apartment form an *Authorized Domain* and register to the *Registration Server* by selecting the *Local Domain Manger* of the apartment block using the protocol given in the Appendix. With the traditional DRM, end users in the apartment block need to register to different content providers in order to get different contents. Moreover, they need to worry about all the DRM agent's compatibility with their devices. Now with the proposed solution, each user (apartment) needs to register only to the *Registration Server* and join the *Local Domain Manager*. With this setup in an apartment block, residents can get different contents regardless of the underlying DRM packages or the content providers. For example, suppose that a resident Alice has two devices which support DRM_1 and one device which supports DRM_2. Among the 3 devices, Alice has dedicated and registered a device which has online connectivity as the *Authorized Domain Manager*. The *Registration Server* will authenticate its integrity using the remote attestation protocol and then install the ADM DRM agent after successful authentication. Then this device can serve as the *Authorized Domain Manager* which manages the domain devices and control the content flow within the domain.

Scenario 2: a collaborative work environment

In this scenario, we assume that there are n different work groups $G_1, \ldots, G_n$ in an organizations which have setup a collaborative multimedia work environment. They use a *Local Domain Manager* to allow them to work together on a distributed project. Assume that for $1 \leq i \leq n$, the group G_i uses the DRM system DRM_i. $G_1, \ldots, G_n$ form different *Authorized Domains* and register to the *Registration Server* and join the *Local Domain Manager* of the organization. These groups can get protected content resources from various content provider in their supported DRM formats. Moreover, they can also share their created work or drafts with each other using the user generated content sharing mechanism to work collaboratively. As an illustration suppose that a video protected with DRM_i has been used by the group G_i and G_i wants the group G_j to use the same video in their part of the project. The group G_j normally cannot use the content with DRM_i. Since the groups G_i and G_j are domains managed by a *Local Domain Manager*, G_j now can request the *Local Domain Manager* to do the content adaptation of the same video to the format DRM_j and use it in the project.

In addition to the above two scenarios, the proposed scheme can also be applied in other cases such as sharing of contents among different organization, sharing of medical results of patients among different hospitals and sharing of educational contents among different schools of a university. Our scheme can be extended to work with authorized domains which are located at geographically dispersed locations. For example, the *Local Domain Manager* can be setup by binding the geographical positions if the *Local Domain Manager* supports GPS. The *Registration Server* will bind the geographical position of the *Local Domain Manager* with those of the *Authorized Domains* registering to that *Local Domain Manager*.

8 Conclusion and future work

In this paper, we have presented a secure interoperable content distribution mechanism using a multi-domain architecture. The proposed method is ideal for content distribution across a group of domains which share common interests on certain digital contents. By using the proposed architecture and the interoperability mechanisms, end devices belonging to various domains can securely and efficiently purchase locally and then use DRM-enabled contents from various content providers.

We have considered different possible situations which can occur during the translation of source DRM-enabled content to the destination supported DRM format. Possible content leakage and illegal content translation and distribution by the middle negotiating entity (local domain manager) are taken care of by using the TD-licence concept and the trusted TPM module installed in the middle entity. We further presented mechanisms for secure sharing of user generated contents within a domain and across domains using the proposed architecture.

In future, we plan to extend our interoperability mechanisms to include the streaming type of media content. Further, we will improve the proposed interoperability mechanisms to accommodate more heterogeneous DRM systems and devices, enhancing the security at the middle entity, improving the violation detection and providing privacy to the end devices and users.

Acknowledgements Thanks to the Agency for Science, Technology and Research (A*STAR), Singapore for supporting this work under the project 'Digital Rights Violation Detection for Digital Asset Management' (Project No: 0721010022).

Appendix A: Registration process

In the registration process, the *Local Domain Manager*, *Content Providers*, *Authorized Domain Managers* and the *End Devices* register to the *Registration Server* (**RS**) by sending their public-key certificate (PKI) to the *Registration Server*. The detailed protocols are given below.

A.1 Registration of a *Content Provider* (**CP**)

1. **CP** sends its PKI certificate and its adopted DRM system to the **RS**.
2. After verification of the **CP**'s authenticity, **RS** adds its details to a list that shows the available *Content Providers* and its DRM agents.

A.2 Registration of *Local Domain Manager* (**LDM**) and *Authorized Domain Managers* (**ADM**)

For registration of a device that is going to function as an *Local Domain Manager* or an *Authorized Domain Manager*, the device has to be TPM-enabled and has a pair of public-key cryptographic keys called $\mathcal{AIK}$ keys generated by the TPM. The **LDM/ADM** obtains an $\mathcal{AIK}$-certificate which contains its $\mathcal{AIK}$ public key signed by a trusted *Certifying Authority*.

1. TPM sends its $\mathcal{AIK}$-certificate and platform configuration state to the **RS**.
2. **RS** verifies the authenticity of TPM using $\mathcal{AIK}$-certificate and **LDM/ADM** device's trust level using the remote attestation protocol described in Section 2.1.2.
3. In the case of **LDM** after successful verifications, **RS** installs the DRM agent DRM_{LDM} in the **LDM** device and sends an **LDM**-certificate that contains general description about its services and security properties signed by the **RS**.
4. In the case of **ADM** after successful authentication, **RS** installs a DRM agent DRM_{ADM} in the **ADM** and sends to the **ADM** a set of domain credentials (i.e., a unique domain key, a domain certificate consisting of a unique domain ID

and the maximum number of devices allowed for that domain, public key of the **ADM** device etc.) for the domain that the **ADM** device is going to manage.

A.3 Registration of an *End Device* (**ED**)

An *End Device* that wants to join an *Authorized Domain* first registers to the *Registration Server* as follows.

1. **ED** sends its PKI certificate to the **RS**.
2. **RS** verifies the authenticity of the **ED**, stores the credentials of the **ED** in its database and sends the list of available *Authorized Domains* and DRM systems to the **ED**.
3. **ED** selects a DRM system from the list which it can support and informs **RS**.
4. **RS** installs the selected DRM system in **ED**.

Appendix B: Multi-domain creation process

The creation of a multi-domain involves the following processes: *Local Domain Manager* adding to it various *Content Providers* who are willing to distribute their contents; joining and registration of *End Devices* to various *Authorized Domains*. The detailed protocols are given below.

B.1 *Local Domain Manager* (**LDM**) adding a *Content Provider* (**CP**)

After the registration, the *Local Domain Manager* can proceed to add *Content Providers* who are willing to distribute their contents.

1. **LDM** selects a desired **CP** from the available list of **CP**s with the **RS**.
2. **RS** forwards the **LDM** to the selected **CP** for authentication and agreement.
3. TPM of **LDM** sends its $\mathcal{AIK}$ certificate and platform configuration state to the **CP**.
4. **CP** verifies authenticity of TPM using $\mathcal{AIK}$ certificate and the trust level of the **LDM** device using the remote attestation protocol described in Section 2.1.2.
5. **CP** sends its PKI certificate to the **LDM**.
6. **LDM** verifies the authenticity of the **CP**.
7. After successful mutual authentication, **CP** grants its approval and sends its specific DRM rules and mechanisms to the **LDM** securely.
8. **LDM**’s *Content Provider Support Module* stores the DRM rules and mechanisms of that **CP**.
9. **LDM** repeats steps 1–8 for all the **CP**s for which it wants to do content negotiation.

B.2 *End Device* (**ED**) joining an *Authorized Domain*

1. **ED** selects a desired **ADM** from the available **ADM** list and sends the request to the **RS** to join that domain.

2. **RS** approves the request and sends a digitally signed user certificate which contains the information of the device and its owner, the public key of the device and the approval from **RS** to the device.
3. **ED** forwards the certificate and request for joining the domain to the **ADM**.
4. After verification of the certificate, if the device satisfies the rules of that domain, **ADM** approves the device as a domain member and stores the certificate in its database.
5. **ADM** sends a domain user certificate and a domain key to the device.
6. **ADM** updates its domain device list and sends the list to the **RS** and all the current domain members.

Appendix C: Multi-domain modification process

A multi-domain gets modified when one of the following happens:

– an *End Device* joins/leaves/removed from an *Authorized Domain*;
– an *End Device* gets transferred from the *Authorized Domain* $\mathbf{AD}_{cur}$ to the *Authorized Domain* $\mathbf{AD}_{new}$;
– a *Content Provider* leaves/joins the multi-domain.

We describe in detail the protocols for the following three cases. The other cases are repetitions of earlier protocols given in the section.

C.1 Leaving/removing of an *End Device* (**ED**) from a *Authorized Domain*

When an *End Device* wants to leave its *Authorized Domain*, it sends a digitally signed Leave Domain Request to *Authorized Domain Manager* (**ADM**). When an *End Device* belonging to an *Authorized Domain* is found to be violating the U-License for any content, *Registration Server* will request the corresponding *Authorized Domain Manager* to remove the device from the domain. *Registration Server* then adds the device identity to the list of revoked devices. The remaining of the protocols in both cases are the following.

1. **ADM** verifies the request, approves the request and removes the certificate of **ED** from the database.
2. **ADM** updates its domain device list and sends the list to the **RS** and all the current domain members.
3. **ADM** updates the domain keys and sends the updated keys to the members of the domain.

C.2 Transferring an *End Device* (**ED**) between *Authorized Domains*

There are some situations in which an *End Device* has to be transferred from an *Authorized Domain* $\mathbf{AD}_{cur}$ to another domain $\mathbf{AD}_{new}$. In OMA DRM [17], transferring of devices across *Authorized Domains* requires to execute a leave domain and a join domain protocols which is inefficient and requires more network resources. In our case, the device need not have to again get authenticated by the *Registration Server*. Instead of executing the join and leave domain protocols, the device can be directly transfered to $\mathbf{AD}_{new}$ with a simpler transfer domain protocol. Devices can check the free slot availability of any $\mathbf{AD}_{new}$ with the *Registration Server* and can send the request of transferring to the *Authorized Domain Manager* of $\mathbf{AD}_{cur}$. We will denote the *Authorized Domain Manager* of $\mathbf{AD}_{cur}$ as $\mathbf{ADM}_{cur}$ and the *Authorized Domain Manager* of $\mathbf{AD}_{new}$ as $\mathbf{ADM}_{new}$. The protocol for transferring an *End Device* from $\mathbf{AD}_{cur}$ to $\mathbf{AD}_{new}$ is as follows.

1. **ED** sends a digitally signed Transfer Domain Request to $\mathbf{ADM}_{cur}$.
2. $\mathbf{ADM}_{cur}$ verifies the request, attests the state of **ED** and approves the request. It sends to **ED** a digitally signed Transfer Certificate showing the device's attestation status and the eligibility to join another domain.
3. $\mathbf{ADM}_{cur}$ removes the domain user certificate of **ED** from the database.
4. $\mathbf{ADM}_{cur}$ updates its domain device list and sends the list to the **RS** and all the current domain members.
5. $\mathbf{ADM}_{cur}$ updates the domain keys and sends the updated key to domain member devices.
6. **ED** sends the Transfer Certificate to the $\mathbf{ADM}_{new}$.
7. $\mathbf{ADM}_{new}$ verifies the Transfer Certificate and generates a domain user certificate and sends along with the domain key to the device.
8. $\mathbf{ADM}_{new}$ updates its domain device list and sends the updated list to **RS** and all the domain members.

References

1. AACS, Advanced Access Content System. http://www.aacsla.com/home
2. Alawneh M, Abbadi IM (2008) Sharing but protecting content against internal leakage for organizations. LNCS 5094:238–253
3. Che S, Che Z, Ma B (2008) An improved image scrambling algorithm. In: Proceedings of the 2008 second international conference on genetic and evolutionary computing, pp 495–499
4. Coral Consortium Whitepaper, Tech. Report, February 2006. http://www.coral-interop.org
5. DVB—The Digital Video Broadcasting Consortium. http://www.dvb.org/
6. DVD Copy Control Association and the Content Scramble System. http://www.dvdcca.org/
7. Jeong Y, Park J, Kim J, Yoon K (2007) DRM content adaptation scheme between different DRM systems for seamless content service. In: Proceedings of ICME 2007, pp 867–870
8. Kanjanarin W, Amornraksa T (2001) Scrambling and key distribution scheme for digital television. In: Proceedings of the 9th IEEE international conference on network. Bangkok, Thailand
9. Kiaei MS, Ghaemmaghami S, Khazaei S (2006) Efficient fully format compliant selective scrambling methods for compressed video streams. In: Advanced international conference on telecommunications and international conference on internet and web applications and services (AICT-ICIW 06)
10. Koenen RH, Lacy J, Mackay M, Mitchell S (2004) The long march to interoperable digital rights management. Proc IEEE 92(6):883–897
11. Kravitz DW, Messerges TS (2005) Achieving media portability through local content translation and end-to-end rights management. In: Proceedings of the ACM workshop on digital rights management, pp 27–36
12. Kulkarni NS, Raman B, Gupta I (2009) Multimedia encryption: a brief overview. In: Recent advances in multimedia signal processing and communications, pp 417–449
13. Lee S, Heejin P, Jong K (2010) A secure and mutual-profitable DRM interoperability scheme. In: 2010 IEEE symposium on computers and communications (ISCC), pp. 75–80. Riccione, Italy
14. Li H, Petkovic M (2007) DRM for protecting personal content. In: Security, privacy, and trust in modern data management (book), data-centric systems and applications (book series), pp 333–346
15. Nam DW, Lee JS, Kim JH (2007) Interlock system for DRM interoperability of streaming contents. In: Proceedings of IEEE international symposium on consumer electronics 2007 (ISCE 2007)

16. ODRL, Open Digital Rights Language, version 1.1, 2002. http://www.w3.org/TR/odrl/
17. OMA, Open Mobile Alliance DRM specifications, version 2.0, July 2004. http://www.openmobilealliance.org/technical/release_program/drm_v2_0.aspx
18. Popescu BC, Crispo B, Tanenbaum AS, Kamperman FLAJ (2004) A DRM security architecture for home networks. In: Proceedings of the 4th ACM workshop on digital rights management, pp 1–10
19. RealSystem Media Commerce Suite, 2001 Technical White Paper, RealNetworks, Inc., 2001. http://docs.real.com/docs/drm/DRM_WP1.pdf
20. Sachan A, Emmanuel S, Das A, Kankanhalli M (2009) Privacy preserving multiparty multilevel DRM architecture. In: Workshop on digital rights management, 6th annual IEEE consumer communications and networking conference, CCNC 2009, 10–13 Jan 2009
21. Security Overview of Microsoft Windows Media Rights Manager, Microsoft, Microsoft Digital Media Division, 2001. http://download.microsoft.com/download/3/7/3/3731217a-7949-4571-8f22-03528ae398ff/WMRM_security.pdf
22. Serrao C, Dias M, Delgado J (2005) Bringing DRM interoperability to digital content rendering applications, CISSE05. The international joint conferences on computer, information, and system sciences, and engineering, Univ. Bridgeport, USA
23. Serrao C, Rodriguez E, Delgado J (2010) Approaching the rights management interoperability problem using intelligent brokerage mechanisms. Computer and Communications 34(2):129–139
24. Taban G, Cardenas AA, Gligor VD (2006) Towards a secure and interoperable DRM architecture. In: Proceedings of the ACM workshop on digital rights management, pp 69–78
25. Takayama M, Tanaka K, Takagi K, Nakajima Y (2008) A scalable video scrambling method in MPEG compressed domain. In: ISCCSP 2008, Malta
26. TCG specification architecture overview, specification revision 1.4, 2 August 2007. http://www.trustedcomputinggroup.org/developers/trusted_platform_module/specifications
27. The MPEG-21 Rights Expression Language, A White Paper, 2003. http://www.xrml.org/reference/MPEG21_REL_whitepaper_Rightscom.pdf
28. Video demonstration, video can be accessed through the link **Interoperable Content Distribution** under **Project Demos** at http://www3.ntu.edu.sg/home/asemmanuel/ or at http://www.youtube.com/watch?v=_KONGNTEDSc
29. Win LL, Thomas T, Emmanuel S, Kankanhalli MS (2009) Secure domain architecture for interoperable content distribution. In: PCM 2009, LNCS 5879, pp 1313–1318
30. Xrml technical overview, version 1.0, 2002. http://www.xrml.org/Reference/XrMLTechnicalOverviewV1.pdf
31. Yan W-Q, Fu W-G, Kankanhalli MS (2008) Progressive audio scrambling in compressed domain. IEEE Transactions on Multimedia 10(6):960–968

Lei Lei Win received her B.E. degree in information technology engineering from Yangon Technological University, Yangon, Myanmar, in 2006 and her M.Sc. degree in digital media technology from School of Computer Engineering, Nanyang Technological University, Singapore in 2008. She is currently working as a research associate with the School of Computer Engineering, Nanyang

Technological University, Singapore. Her current research interests are multimedia and information security, digital rights management systems and architectures, and privacy preserving techniques for DRM. She served as a reviewer in IEEE TENCON, 2009 conference.

Tony Thomas obtained his M.Sc. and Ph.D. in Mathematics from the Indian Institute of Technology, Kanpur (IIT Kanpur), India. He is currently working as a research fellow at the School of Computer Engineering, Nanyang Technological University, Singapore. Prior to that, he worked as a researcher at the General Motors Research Lab, Bangalore, India and Korea Advanced Institute of Science and Technology, Daejeon, South Korea. His current research interests are in cryptography, DRM, digital watermarking, multimedia security, security in mobile ad-hoc domain, secure multiparty computations and visual surveillance.

Sabu Emmanuel received the B.E. degree in electronics and communication engineering from Regional Engineering College, Durgapur, India, in 1988, the M.E. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, in 1998, and the Ph.D. degree in computer science from the National University of Singapore in 2002. He is currently an assistant professor in the School of Computer Engineering, Nanyang Technological University (NTU), Singapore. He began his career as a Research and Development Engineer and then moved on to the academic profession. He has taught engineering students of Mangalore University, National University of Singapore, and NTU. His current research interests are in multimedia and software security and surveillance media processing. He has been a Reviewer for ACM Multimedia Systems. He is a member of the Technical Program Committee of several conferences. Dr. Emmanuel has been a Reviewer for *IEEE Transactions on Circuits and Systems for Video Technology*, *IEEE Transactions on Multimedia* and *IEEE Transactions on Information Forensics and Security*.