Interoperable DRM Framework for Multiple Devices Environment

Seong Oun Hwang and Ki Song Yoon

As networks increase and cross-convergence occurs between various types of devices and communications, there is an increasing demand for interoperable service in the business environment and from end users. In this paper, we investigate interoperability issues in the digital rights management (DRM) and present a practical framework to support interoperability in environments with multiple devices. The proposed architecture enables end users to consume digital content on all their devices without awareness of the underlying DRM schemes or technologies. It also enables DRM service providers to achieve interoperability without costly modification of their DRM schemes.

Keywords: DRM, interoperability, content protection, security.

Manuscript received Nov. 6, 2007; revised Feb. 26, 2008; accepted Apr. 15, 2008.

This work was partly supported by the postdoctoral fellowship program of IITA, MKE (Ministry of Knowledge Economy) and 2008 Hongik University Research Fund, Rep. Korea.

Seong Oun Hwang (phone: 82 41 860 2298, email: sohwang@hongik.ac.kr) was with Broadcasting & Telecommunications Convergence Research Laboratory, ETRI, Daejeon, Rep. of Korea, and is now with the Department of Computer and Information Communication Engineering, Hongik University, Chungnam, Rep. of Korea.

Ki Song Yoon (email: ksyoon@etri.re.kr) is with SW & Content Research Laboratory, ETRI Daejeon, Rep. of Korea.

I. Introduction

The Internet has greatly facilitated the distribution and exchange of information, and this has led to increasing problems regarding the issue of intellectual property and copyright infringement. Digital content is by nature very vulnerable to unauthorized distribution and use. To cope with this problem, digital rights management (DRM) technologies have been developed. Although they have functioned as intended, they have caused an unexpected problem. Most DRM systems are not interoperable. As a result, consumers cannot play their purchased, DRM-protected products on all of their devices.

For example, a mobile phone service provider in South Korea prohibited playback on their mobile phones of digital songs purchased from other online sites. It caused many users to complain and to file lawsuits. A survey by Indicare [1] shows that 86% of potential digital music customers would rather pay 1 Euro for interoperable content (having the right to use it in more than one device) instead of paying just 50 cents for device bound content. The same applies to the business sector. Recently, as DRM 2.0 specifications of Open Mobile Alliance (OMA) [2] have been implemented and commercialized, interoperability issues between OMA DRM and MS DRM have risen. Therefore, the issue of DRM interoperability has become a problem which should be solved to benefit businesses as well as consumers.

The contribution of this paper is to present a practical framework to support DRM interoperability among resource-limited devices such as mobile phones and portable devices. The framework is based on two concepts: domain and profile. The domain concept enables the framework to be interoperable regardless of underlying devices including consuming environments. The profile concept enables the framework to achieve interoperability without modification of participating DRM schemes.

The remainder of this paper is organized as follows. In section II, we provide an overview of previously proposed approaches toward DRM interoperability. Section III presents our approach at a very high level. Section IV gives detailed information of system components comprising our prototype DRM framework. Section V presents an analysis of our framework by comparing it with other schemes. Section VI concludes the paper with a discussion of the contribution of the paper and future work.

II. Related Work

In this section, we first give an overview of some of existing approaches toward interoperability. Then we analyze their characteristics and classify them into types of interoperability.

1. Overview

A. MPEG-21

MPEG-21 [3] provides standards for protection and management of multimedia content by introducing a *hooks* architecture and interfaces between intellectual property management and protection (IPMP) tools. That is, MPEG IPMP does not standardize IPMP itself; rather, it standardizes the IPMP interface, which allows flexibility between the IPMP system and applications. A terminal accesses the IPMP tool list of protected content media and determines the IPMP tool that is required to consume the content. MPEG IPMP tries to solve interoperability issues by searching for and installing the required DRM tools whenever they are needed.

B. OMA

The OMA DRM specification version 2.0 released in February 2004 provides additional features and a significantly higher level of security through mechanisms based on public key infrastructure (PKI) to protect high-value digital contents, such as MP3 audio files or video clips. OMA DRM specifies export of content and its associated usage rights to other DRM specifications, but it does not address the security issues which arise during the export process. It specifies these as being beyond the scope of the standardization.

C. Digital Media Project

DMP [4] released a comprehensive technology specification for interoperable digital rights management as well as applications within and across media value chains. The documents specify that all the actors in the value chain perform some functions to do business. Those functions can be decomposed into smaller primitive functions, which seem to be implemented into a set of tools called interoperable DRM platform (IDP) toolkit. Unlike MPEG, DMP defines its own DRM formats such as DRM/Authentication Messages, Domain/ Access Protocols, and so on.

D. Coral

The Coral Consortium [5] approach is a DRM-neutral interoperability framework to solve multiple DRM problems. While maintaining current DRM systems and devices, it tries to achieve service-level interoperability by providing trusted interfaces and functions that mediate differences between DRM schemes, such as rights meditation, content transformation, and repackaging.

2. Classifications of Existing Approaches

The following subsection presents a classification of the approaches according to the type of interoperability they provide: full-format, building block, and translation-based interoperability. Refer to Koenen's paper [6] for a similar classification.

A. Full-Format Interoperability (DMP, OMA, etc.)

This seems to be one of the most complete ways of providing interoperability at first glance. This approach has been adopted by most DRM standardization organizations and vendors. In principle, it seems difficult for a single vendor or organization to cope with interoperability issues in the DRM area with this approach. We note that since this approach accompanies disclosure of the entire structure of DRM, it may be very vulnerable from the security point of view.

B. Building Block-Based Interoperability (MPEG-21)

Under the assumption that all the terminals and application players can access all the DRM tools (authentication, encryption/decryption, watermark, and so on), it allows us to search/download and use the appropriate tools when consuming a protected content. Compared to other approaches, it is highly flexible. However, in reality, only a particular tool can be available to a particular platform. It is not clear that all devices provide all the resource required to store and execute when accessing the multimedia content which belongs to a user.

C. Translation (Transformation)-Driven Interoperability (Coral)

This approach assumes the existence of a trusted third party (TTP) connected to the network. The TTP provides interoperability by giving translation operations between different formats – protected content formats, rights, and messages. Translation (or transformation) operations are likely to incur information loss which may occur during incorrect mapping from the source to the destination DRM scheme. Another drawback of this approach is that it requires the devices to be connected to the network, at least one time, when the device initially connects to the interoperable service provider.

Next we classify the approaches into the two categories of intra-DRM and inter-DRM interoperability.

D. Intra-DRM Interoperabililty (DMP, OMA)

This approach tries to achieve interoperability between DRM schemes which implement the same DRM specification. Most standard organizations have focused their activities on supporting interoperable DRM service within their own DRM schemes. So far, there seems to have been no distinct effort or cooperation between standard organizations toward interoperability between heterogeneous DRM schemes.

E. Inter-DRM Interoperabililty (MPEG, Coral, Proposed Scheme)

This approach tries to address the interoperability between different DRM schemes which implement different DRM specifications. This is the level of interoperability that the end users eventually want to be deployed in their devices. To address this issue appropriately, we need to consider the interoperability issues inherent in DRM.

One critical factor that makes interoperability difficult is the trust model. Different DRM schemes are usually designed under different trust models. For example, the trust model of OMA DRM is based on PKI. Through a public key certificate and digital signature, it allows one to identify and authenticate the other party, and it guarantees the integrity and secrecy of an exchanged message. MPEG-21, by contrast, does not specify a particular authentication framework; rather, it determines a specific authentication mechanism based on conversational negotiation. That is, under the MPEG-21 scheme, any authentication mechanism such as X.509 public key certificates, Kerberos shared-secret tickets, or password digests can be deployed. The trust model directly affects the rights model and licensing model. To express a usage rights, OMA DRM uses OMA DRM Rights Expression Language (REL) version 2.0 rooted in ODRL [7], whereas MPEG-21 uses MPEG-21 REL 2.0 rooted in XrML [8]. As Safavi-Naini's paper [9] indicates, it is not straightforward to directly translate rights expressions from MPEG-21 to OMA, or vice versa. Other factors that make it difficult to provide interoperability include differing identification schemes (e.g., content, device) and differing protected content formats (e.g., MPEG-21: DID, DII, OMA: DCF, PDCF) taken by DRM schemes.

So far, we have discussed the inter-DRM interoperability issue at a very high level. There have been similar approaches towards interoperability at the system level. Jamkhedkar and others [10] proposed a layered DRM architecture and approached the issue by standardizing interfaces between the layers. Michiels and others [11] took a layered approach different from Jamkhedkar's and refined the DRM functions within the layers.

III. System Overview

To design an interoperable DRM, we need to consider the problem further on the implementation level as well as the conceptual level. The following conceptual and implementational issues regarding DRM interoperability are drawn from our studies of the previously mentioned approaches.

1. Design Considerations

A. Reduction of Information Leak or Loss

Except for MPEG, most approaches have intrinsic security issues in the sense that an exported content item from a source DRM scheme is decrypted and imported into a target DRM scheme. During the export-import process, the content has a high risk of unintentionally revealing the original content itself to the outside. Another security issue with the existing approaches is that they reveal their DRM structures to each other, for example, what the DRM protected content format looks like, how the DRM scheme works, and so on. These can be further sources of security vulnerability. The revelation problem of DRM structure was also addressed in the paper [12]. To cope with these issues, we need to prevent the extraction of original content media from protected content media during the process. We also need to protect each participating DRM scheme's internal structures. Information loss may be incurred while transforming a content protected by source DRM scheme to the one protected by target DRM scheme.

B. Transparent Service to End User

A DRM system usually works by controlling the access of the user to a protected content based on the allowed rules described in the rights objects. It puts some restrictions on the user's use of content. An interoperable DRM system usually requires the installation of additional modules on the user's side, which exacerbates the situation or problem. Therefore, additional processes which overburden users should be kept hidden or minimized as much as possible from the user's point of view.

C. Complexity of Implementation and Modifications to Existing Environment

A DRM system should be designed to apply DRM functions easily without major modifications of participating DRM modules or protected content format as well as existing player environment such as media players. In principle, it might be possible to achieve interoperability between totally different DRM schemes if we invest a huge effort, time, and resources, although that does not seem very cost-effective. Most DRM schemes that aim to achieve interoperability disregard this aspect and do not succeed.

2. DRM Architecture

We define interoperability as the ability to enable users to consume their protected content regardless of the underlying infrastructures, such as DRM schemes, devices, networks, or services, in a transparent way and with no information loss.

We describe our DRM architecture in the abstract view of component models, each of which constitutes a consistent DRM scheme. This high level approach is convenient for designing and analyzing a DRM scheme. From this point of view, a DRM scheme, a collection of component models, is thought to be an enabler that performs a service to establish a trust environment where digital content and its embedded rights are executed. In our approach, we define a basic foundational role model and further construct additional models on that.

A. Role Model

We identify actors who perform major functions in our DRM scheme. An actor is a logically independent entity in terms of performing functions. It does not need to exist independently in the physical sense. A set of functions performed by a number of actors can be done by one physical entity in the real world. In the following, we use the term "traditional" to indicate the context of existing DRM schemes which do not provide interoperability. The proposed DRM scheme referred to as S-DRM (Standard-DRM) which is intended to provide interoperability comprises the following actors:

- **Certificate authority (CA)** is a TTP that provides services for the creation and distribution of electronic certificates.
- **T-DRM service provider** is an existing, traditional DRM service provider which issues usage rights and manages DRM tools on the end user's side.
- **S-DRM service provider** provides an interoperable DRM service environment where a content item protected under a traditional DRM scheme can be consumed using S-DRM modules, such as an S-DRM profile generator, an S-DRM rights generator and an S-DRM agent. The key role of the S-DRM service provider is to provide domain functions which enable interoperable DRM service irrespective of underlying traditional DRM schemes.
- **S-DRM profile generator** generates a profile for traditional, protected content. It is usually done by a T-DRM service provider which provides the corresponding content.
- **S-DRM rights generator** issues a rights object that enables interoperable use of traditional protected content. It is usually done by a T-DRM service provider which provides the corresponding content.
- **S-DRM agent** is a minimal unit trusted by the S-DRM service provider. It is installed on the end user's device and provides access control of protected content, resolution of usage rights, and domain processing.
- **End user** must belong to at least one domain to receive an interoperable DRM service and access to protected content using the S-DRM agent.

B. Domain Model

A domain is a logical concept for grouping entities, including devices, users, organizations and so on, that access and consume protected content under the same usage rights. In this paper, a domain consists of multiple participating devices. On the participating devices side, a domain manager is installed. Domain managers can be classified into two types according to function: master and slave. All devices in a domain except for the master are considered peers. The master functions as a server for the client of peers. Therefore, a domain consists of a master domain manager and one or more slave domain managers. A master domain manager creates a domain and registers it with the S-DRM domain server. During the registration, the master domain manager establishes an authorized domain (domain ID, domain key, domain policy) from the S-DRM domain server. After a domain is created, devices can join or leave the authorized domain by having their slave domain manager run a join or leave protocol with the master domain manager. Unlike a slave domain manager, a master domain manager executes a registration or update protocol with the S-DRM domain server. We assume that a slave domain device can be connected to the master domain device through various kinds of device-to-device networks including IEEE 1394, USB, Ethernet, Bluetooth, and so on. We also assume that a master domain device can be temporarily or intermittently connected to the network. It is not necessary for a master domain device to be always connected to the network. The separation of roles between the master and slave domain manager reduces the device network connectivity requirement.

C. Profile Model

It is usually difficult to get inter-DRM interoperability between heterogeneous DRM schemes because DRM vendors transform original content into protection formats of their own. They only have access to their own DRM (client) modules, which interpret and process the formats on the end user's side. We call this client DRM module a DRM agent. To enable a uniform, generic interpretation and processing of different DRM schemes in our interoperable DRM framework, we introduce the concept of profile. A profile can be defined as an information structure representing each traditional DRM scheme. It is a minimal, essential information structure that enables equivalent simulation of each traditional DRM process in our framework by providing several kinds of information, such as protected content structure, usage rights, data extraction method, and data processing method. The traditional DRM service provider, which is intended to provide an interoperable DRM service, describes or puts DRM structure information into the profile template which is provided by S-DRM service provider. The profile model defines content structures, rights, and the required processing method of traditional DRM schemes.

D. Trust and Security Model

We assume that there is an S-DRM service provider which is trusted by the traditional DRM service providers. The assumption is very natural because a trust relationship is commonly achieved based on bilateral agreements between an S-DRM service provider and traditional DRM service providers. An S-DRM service provider enables interoperable DRM service by providing trust infrastructure and software tools.

We also assume that all the entities described in this role model have their own unique private/public key pairs and certificates. This assumption is also appropriate in the sense that public key infrastructure has becomes an essential part of current digital commerce. To ensure the enforcement of DRM policies, it is recommended that rendering devices, such as mobile phones or MP3 players, be equipped with a trusted computing base (TCB) which can include a tamper-resistant module, such as a trusted platform module (TPM) [13]. A TPM can be used to provide secure storage for important information, such as keys, and cryptographic operations, such as encryption and random number generation. As in most classical DRM security models, we assume that content is encrypted under a content encryption key (CEK). However, the CEK is delivered as a form of rights to the DRM agent on the user's side as encrypted under the key of the domain which the DRM agent belongs to. The CEK should have a high enough security level so that without the CEK itself, the attacker cannot retrieve the original content item from the encrypted content. We assume that the attacker cannot access the DRM agent's domain key which will be explained later. To this end, it is recommended to keep the domain key in TPM storage.

E. Rights Expression Model

There is also a real need for interoperability in the rights expression model, particularly between the ODRL series (OMA REL) and the XrML series (MPEG-21 REL, Microsoft REL). As is well known, those RELs are based on the same language, the DPRL [14]; therefore, they are broadly very similar in structure and semantics. We define our REL by starting on the basic structure of ODRL V1.1 and augmenting it with elements in XrML 2.0 as needed. ODRL is comparatively simple and clearly defines both the formal model for rights expression (rights, permission, assets, and so on) and the semantics of the concrete elements which are used to express a rights instance (play, pre-pay, and so on). The key reason we chose this approach is our observation that people want simple but clear rights expression in the real world, at least regarding interoperability. Our approach is partly supported by the results presented in [9], which concludes that direct translation between ODRL and XrML is possible without loss of information except for several complex contexts and situations.

In the following, we show how the traditional DRM and proposed DRM systems work regarding interoperability.

First, we assume that there are two different DRM schemes, T-DRM A and T-DRM B, respectively installed on two terminals, and S-DRM not installed. At this stage, there is no interoperability between two terminals. For example, content that is protected by the T-DRM A service provider can be consumed by the T-DRM A agent on terminal I under the permission dictated by T-DRM A rights, but not on terminal II. This is because T-DRM A content cannot be interpreted and processed by the T-DRM B agent on terminal II or vice versa.

Figure 1 shows the functional architecture of the proposed system. Here we explain how to provide interoperability when moving protected content from one terminal to another. We assume at this time that S-DRM modules are installed on two terminals. A user downloads and installs both the S-DRM rights and the T-DRM A profile from T-DRM service provider A. The S-DRM on terminal I sends both the S-DRM rights and T-DRM A profile to the S-DRM on terminal II. At this time, the T-DRM A content can be sent between terminals using the S-DRM transfer function or as stored in the memory card. The T-DRM A profile can be sent using the same transfer function or over the server transmission channel. By accessing the

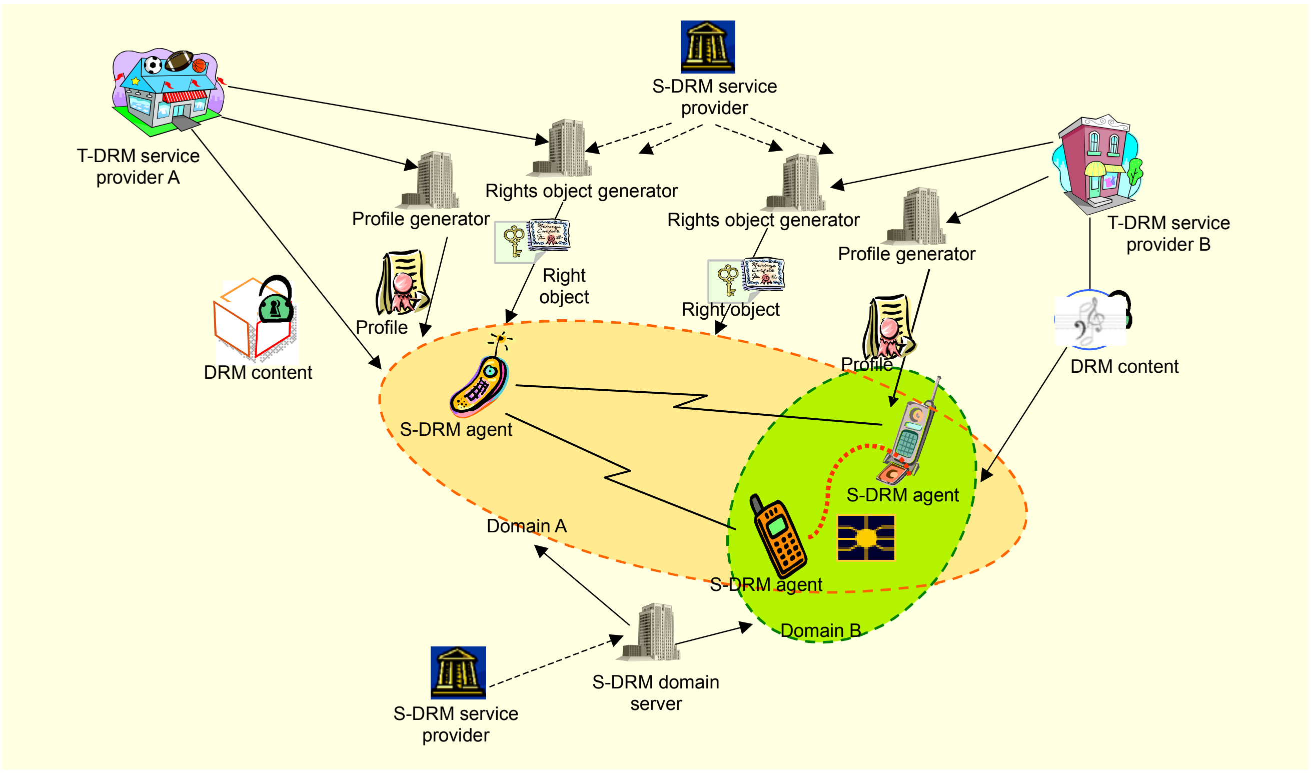

Fig. 1. Functional architecture of the proposed system.

T-DRM A profile, the S-DRM on terminal II retrieves the information needed to access and process the T-DRM A content. From the S-DRM rights, it also retrieves the information needed to decrypt the T-DRM A content.

IV. System Details

This section covers details of the proposed framework: S-DRM internal structure, profile, key architecture, and protocols.

1. S-DRM Internal Structure

S-DRM at the user's side consists of the S-DRM agent and its interface to outer rendering applications. Figure 2 shows the internal structure of S-DRM. The application service interface (API) [15] is open to applications and satisfies two requirements. First, an application should be easily developed by just looking at the interface, without deep knowledge of DRM technologies. Second, it should be flexible enough to support various DRM business models. However a DRM scheme with open interfaces and components may have security weaknesses, so it is necessary to minimize or control the level of openness.

2. Profile

The S-DRM profile specifies the data and the data extraction/processing method which allow the S-DRM to process the protected content formats provided by traditional DRM vendors. It is specified using XML schema. The S-DRM is mostly concerned with extraction of data, interpretation of data, decryption of media data, and communication with rendering application.

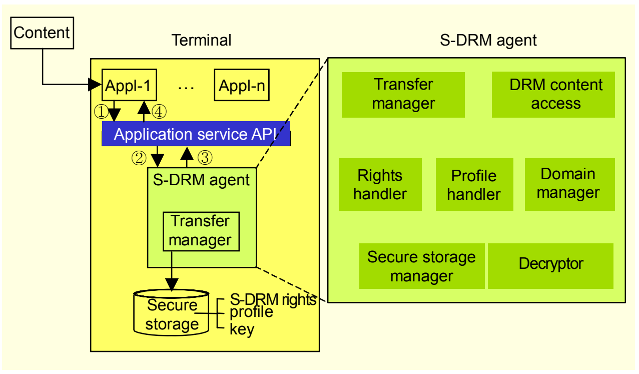

Fig. 2. Internal structure of S-DRM agent.

The S-DRM profile schema [15] consists of information on the profile itself and information on DRM content.

(1) The profile context consists of the profile structure and context structure. The profile structure contains metadata on the profile itself, metadata on the DRM content, and information extracted from the DRM content. The context structure

contains information such as identification and the version of the targeting DRM scheme.

(2) The DRM content largely consists of the DRM metadata, the protected content structure, and the inventory structure. The DRM metadata structure specifies how to extract metadata from the DRM content. The metadata from the DRM content includes the following: DRM scheme ID, DRM scheme version, content ID, MIME type of DRM content, file extension of DRM content, MIME type of raw content, encryption method, and right issuer URL. The protected content structure provides the information needed to retrieve raw content from the DRM content, such as plain_text_length and encrypted data. The inventory structure serves as an efficient tool to express the profile by reducing redundancy, but it does not express the DRM content itself.

To maximize the universality or power of expression, we referred to the DRM reference model by MPEG LA [16]. We also consider exceptional cases for which profiling is not appropriate. For example, for the cases in which extraction or processing of specific data is so complex that the profiling of specific DRM functions is not efficient or not allowed due to security policy, we allow plug-in architecture as an alternative method, which can be implemented using XML-RPC [17] functions provided by a specific DRM scheme or vendor.

3. Key Architecture

Keys are largely classified into three kinds of keys, a device-related one (device key), a content-related one (content encryption key), and a key binding those two keys (domain key). The content encryption key uses the AES [18] encryption algorithm with a 128-bit key. The device key and the domain key use the RSA [19] encryption algorithm with a modulus 1024-bit key.

A. Device Key

Every device has its own public key/private key pair and a device certificate which is installed at manufacturing time by the device manufacturer. Its form can vary depending on the type of device and manufacturer. We assume the existence (use) of a secure storage area for the key storage. For a mobile terminal, a universal subscriber identity module (USIM) is an example of a secure area. The device key is used to encrypt information, for example, a domain key that should be transmitted securely to each individual device.

B. Content Encryption Key

The content encryption key is used to encrypt a media content item. The content encryption key is encrypted under the content recipient's domain key (which is contained in the rights) and delivered to the content player of the content recipient device.

C. Domain Key

The domain key is assigned to each domain that consists of a number of devices and is used to encrypt the content encryption keys. When an S-DRM agent is installed, a domain is established. This newly generated domain is managed by a master domain manager. During the registration, the master domain manager gets the domain key from the S-DRM domain server. Every device can be joined to the domain by allowing the slave domain managers on the devices to conduct a join protocol with the master domain manager under the intervention of the user or not. After the join protocols are executed successfully, all the devices pertaining to the domain result in sharing the same domain key. If a device leaves the domain, the corresponding domain manager should update the domain key and distribute it among the participating devices.

4. Protocols

This section explains protocols to support operations introduced in the above architecture: basic setting, S-DRM terminal registration, domain management, and content rendering.

A. Basic Setting

To provide interoperability with traditional DRM schemes, both S-DRM rights and T-DRM profiles should be transferred to the S-DRM agent on a terminal. To do this, the traditional DRM service provider should provide the following services:

- To allow the S-DRM terminal to join or leave the S-DRM domain using the S-DRM domain manager
- To generate a T-DRM profile using the S-DRM profile generator and send it to the S-DRM agent
- To generate S-DRM rights using the S-DRM rights generator and send the data to the S-DRM agent

A profile can be open to or kept secret from outside world. This depends on the security policies of traditional DRM vendors. If they choose an open policy, the secrecy of the DRM system is equivalent to one of the open DRM systems pursued by standardization organizations such as OMA, DMP, and so on. If they keep their profiles secret, the profiles should be securely kept. In this case, profiles should be transmitted to and accessed from the authenticated S-DRM agent. To do this, the vendors may use the domain key to encrypt the profiles. Domain keys are generated during S-DRM terminal registration which will be further explained. We note that a domain is managed by the S-DRM service provider, and different T-DRM service providers do not need to agree upon a common domain.

B. S-DRM Terminal Registration

To get S-DRM rights issued, an S-DRM terminal needs to be registered with a T-DRM service provider. Before registering with the T-DRM service provider, a domain context should be established. The S-DRM terminal requests domain setup to the domain server on the S-DRM service provider's side. During domain setup with the S-DRM service provider, the domain context is established on the S-DRM terminal. The domain context includes domain ID, domain certificate, domain key, list of participants' (devices') IDs, expiration date, and domain constraints (that is, the maxim number of terminals which can join the domain). Then, the S-DRM terminal registers its information with the T-DRM service provider. The information includes the protocol version, T-DRM terminal ID, S-DRM terminal IDs, and domain context. This registration protocol should be executed whenever the domain is updated.

C. S-DRM Domain Management

In this subsection, we describe how to set up, join, or leave an S-DRM domain.

Protocol to Set Up/Update a Domain

1. The device initiates contact with the S-DRM domain server.
2. The S-DRM domain server authenticates the device and communicates the domain context (including a new/updated domain key, domain ID, a list of participating devices, and the maximum number of participating devices) to the device over the secure authenticated channel (SAC).
3. The device establishes its own domain context and stores the domain key at the secure storage of the device.

In this protocol, for authentication of the device, the S-DRM domain server sends an authentication request to the device. It then extracts the device ID from the certificate sent by the device. It then checks that the certificate sent by the device is valid.

Protocol to Join a Domain

1. An S-DRM slave domain manager device initiates contact with the S-DRM master domain manager.
2. They establish an SAC and authenticate each other.
3. The S-DRM master domain manager checks if a given set of domain rules and policies (such as the number of acceptable devices in the domain) are met. If the check result is OK, then it transmits a domain key to the device over the SAC and updates the domain context.
4. The S-DRM slave domain manager stores the domain key at the secure storage area.

In this protocol, the S-DRM master domain manager takes a delegated role of joining an S-DRM slave domain manager to the domain on behalf of an S-DRM domain server. As another option, to improve the security of the domain key, it is possible to update domain keys whenever the join protocol is performed. In this case, the S-DRM master domain manager performs the update procedure with the S-DRM domain server and broadcasts the new domain key to the connected member devices including the joining S-DRM slave domain manager. To reduce the overhead caused by frequent updates of domain keys and to strike the balance between performance and security, we recommend that the domain key remains the same when the join protocol is performed. This join protocol can be seen as a limited version compared to the full version of the join protocol, where the domain key is updated whenever the join protocol is performed.

Protocol to Leave a Domain

1. An S-DRM slave domain manager device initiates contact with the S-DRM master domain manager.
2. They establish an SAC and authenticate each other.
3. The S-DRM master domain manager performs the update procedure with the S-DRM domain server.
4. The master domain manager broadcasts the new domain key to the connected member devices except the leaving slave domain manager over SAC.

Unlike the join protocol, the domain key should be updated and distributed among the participating devices whenever the leave protocol is performed so that the device which is leaving cannot consume (decrypt) new content items published in the domain after it leaves the domain.

Note that when a user requests to acquire a new content item after a domain key is updated, the content item and a new domain key are transmitted to the device. During this procedure, the domain key version of the device is checked against the version of the S-DRM domain server. If the domain key has not been updated, then new domain keys are transmitted to the device.

In the leave procedure, when the network connection between the S-DRM master domain manager and the S-DRM domain server is not available, steps 3 and 4 can be performed afterwards when the network is available. In this case, the S-DRM master domain manager generates and broadcasts a temporary, short-lived domain key to slave domain managers except the leaving S-DRM slave domain manager. The temporary domain key should be replaced with the one issued by the S-DRM domain server later.

Users, particularly advanced users, can register new devices in their multiple devices environment. However, to reduce the difficulty or complexity a user might feel during registration, we provide an automatic registration method that does not require the user's involvement.

The concept of domain in DRM originally was intended to allow content sharing among devices owned by an entity, for example, the same household [20]-[22]. Although the usage scenarios of the proposed method are somewhat similar to those of previous methods based on the domain concept, the details including implementation mechanisms and protocols are different.

D. Content Rendering

When a user plays DRM-protected content, the following occur in the background.

1. The application gives the name of the protected content and rights to the S-DRM agent.
2. The S-DRM agent authenticates the rights and retrieves the domain key from secure storage of the device.
3. The S-DRM retrieves the content encryption key from the rights using the domain key and decrypts the protected content using the content encryption key.
4. The application renders the decrypted content by the S-DRM agent.

V. Analysis

In this section, we present the case study result and compare the proposed scheme with other interoperable DRM schemes.

Before this research, we implemented two DRM systems in the mobile communication and entertainment environments: an OMA DRM version 2.0 system and our proprietary DRM system. The DRM agents of the systems were employed on a smartphone (ARM9, 400 MHz, CDMA1x EDVO 2.4 Mbps). We produced an OMA DRM protected content item and its related rights. From the rights, we retrieved the content encryption key and constructed other rights which could be interpreted by our proprietary DRM agent. The critical point is how to construct a profile template for OMA DRM 2.0. Note that the objective of the case study is to check the feasibility of the profile concept rather than to establish a normative, extensive form of profile. To name a few items in the profile, it includes an encryption algorithm, a padding scheme, the rights issuer URL, and the offset/size/type of encrypted data. In the profile, we support AES-128-CBC and AES-128-CTR as encryption algorithms. We support RFC 2630 [23] as padding schemes in addition to non-padding schemes.

In the following, we analyzed how well the proposed system addressed the issues that were set forth in section III.1.

From the security point of view, the proposed system achieved the security requirements previously discussed, namely, the prevention of information loss or leakage during interoperable DRM service. The proposed scheme avoids this through the deployment of the secure profiling concept. It avoids direct translation or transformation which might be vulnerable to information loss. That is, the proposed scheme starts off without changing the existing DRM information structure. It only allows the addition of information structures needed to bridge different DRM structures. Therefore, it is relatively free from the problem of information loss.

Content protection in our proposed system is dependent upon the security of domain keys. For attackers to decrypt content, they need to access CEK which was already stored as encrypted under the domain key. However, they fail because the domain key is managed at the access controlled file system provided by mobile device manufacturers in our case study. To improve the security level, we recommend using hardware modules such as smartcards, and in the case of some mobile phones, USIM.

From the user's point of view, the system provides a means to control the level of any awareness or actions at the end user's side. To maximize transparency, the system provides DRM interoperability without requiring any awareness or action on the part of users. It is reported that only Coral provides transparency. The proposed scheme provides transparency based on a domain model which allows actions usually done by a user to be performed by the server on behalf of the user. The user does not need to be aware of the details of devices or protection schemes or which devices support which protection schemes. Instead, a user only needs to perceive the external views without considering the underlying protection schemes.

The traditional approaches toward DRM interoperability require very high implementation complexity, which is not appropriate for devices such as mobile phones or portable media players, which have very limited resources (CPU, memory) and are intermittently connected to the network. From the implementation complexity point of view, many traditional schemes require modifications to protected content format, playback software, or DRM modules, in part or as a whole. In general, it is inevitable that DRM makes some modifications to playback software.[1] To reduce the number of necessary modifications, the proposed scheme opens an API set in order to make possible modifications fewer, easier, and timelier. Another issue to consider when implementing interoperable DRM service is the dependency of network

1) Note that the proposed framework is based on the chains of trust among parties, including playback software providers, T-DRM service providers, and S-DRM service provider.

connectivity. For example, Coral is a highly connected model which cannot be easily extended to intermittently connected devices, such as portable music or video players. The proposed scheme runs even in off-line environments.

VI. Conclusion

In this paper, we presented an overview of some existing approaches toward interoperability of DRM schemes. We analyzed their characteristics, classified them, and identified challenging issues in this area, including information loss/leak, transparency and implementation complexity. To address these issues, we proposed an interoperable DRM framework based on the concepts of domain and profile. The domain concept allows users to consume content on their devices in a more natural, and transparent way. We also provided domain protocols that do not require the involvement of users. The profile concept enables interoperability through add-on functionality. It does not require major modification of existing protected content formats or DRM modules; therefore, the proposed scheme can considerably reduce the possibility of information loss or leak. Our framework also solved the implementation issues by opening the API, which enables rapid development of applications by developers without deep knowledge of DRM technologies.

In the proposed system, all participating DRM schemes are based on PKI. Under the proposed framework, it may be possible to provide interoperability between non-PKI DRM schemes and our S-DRM scheme in an ad hoc way. Surely it is not a desirable direction. As a future research topic, we could consider how to support DRM interoperability between PKI and non-PKI based DRM schemes.

References

[1] INDICARE, "Consumer Survey on Digital Music and DRM," http://www.indicare.org, May 2005.

[2] OMA (Open Mobile Alliance) 2.0, http://www.openmobilealliance.org, September 2007.

[3] MPEG-21, http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm.

[4] DMP (Digital Media Project), http://www.dmpf.org.

[5] Coral, "Coral Consortium Core Architecture 3.0," http://www.coral-interop.org, June 2006.

[6] R.H. Koenen, J. Lacy, M. Mackay, and S. Mitchell, "The Long March to Interoperable Digital Rights Management," *Proc. the IEEE*, vol. 92, 2004, pp. 883-897.

[7] ODRL (Open Digital Rights Language), "Open Digital Rights Language v1.1," http://www.odrl.net.

[8] ContentGuard, eXtensible Rights Markup Language (XrML) 2.0 Specification, Nov. 2001.

[9] R. Safavi-Naini, N.P. Sheppard, and T. Uehara, "Import/Export in Digital Rights Management," *Proc. ACM Workshop on Digital Rights Management*, 2004.

[10] P. Jamkhedkar and G. Heileman, "DRM Interoperability Analysis from the Perspective of a Layered Framework," *Proc. ACM Workshop on Digital Rights Management*, 2005.

[11] S. Michiels, K. Verslype, W. Joosen, and B. De Decker, "Towards a Software Architecture for DRM," *Proc. ACM Workshop on Digital Rights Management*, 2005.

[12] B. Choi, Y. Byun, J. Nam, and J. Hong, "A Tool Pack Mechanism for DRM Interoperability," *ETRI Journal*, vol. 29, no. 4, 2007, pp. 539-541.

[13] Trusted Computing Group, *Trusted Computing Platform Alliance Main Specification Version 1.1b*, Feb. 2002.

[14] Xerox Corporation, *The Digital Property Rights Language*, 1998.

[15] S.O. Hwang and K.S. Yoon, *The Mobile DRM Specifications V 1.0*, Telecommunications Technology Association (TTA), Korea, 2006.

[16] MPEG LA (Licensing Authority), "DRM Reference Model 3.0," http://www.mpegla.com.

[17] XML-RPC, http://www.xmlrpc.com.

[18] Advanced Encryption Standard (AES), NIST FIPS 197, http://csrc.nist.gov/encryption/aes/index.html.

[19] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, 1978.

[20] DVB (Digital Video Broadcasting), "Part 7: CPCM Authorised Domain Management," in *Content Protection & Copy Management Specification,* 2007.

[21] B.C. Popescu, F.L.A.J. Kamperman, B. Crispo, and A.S. Tanenbaum, "A DRM Security Architecture for Home Networks," *Proc. 4th ACM Workshop on Digital Rights Management*, 2004.

[22] T.S. Messerges and E.A. Dabbish, "Digital Rights Management in a 3G Mobile Phone and Beyond," *Proc. the 3rd ACM Workshop on Digital Rights Management*, 2003.

[23] R. Housley, *Cryptographic Message Syntax*, IETF RFC2630, http://www.ietf.org/rfc/rfc2630.txt, 1999.

Seong Oun Hwang received his BS degree in mathematics in 1993 from Seoul National University, his MS degree in computer and communications engineering in 1998 from Pohang University of Science and Technology, and his PhD degree in computer science from Korea Advanced Institute of Science and Technology. He worked as a software engineer at LG-CNS Systems, Inc. from 1994 to 1996. He worked as a senior researcher at Electronics and Telecommunications

Research Institute (ETRI) from 1998 to 2007. Since 2008, he has been working as an assistant professor with the Department of Computer and Information Communication Engineering of Hongik University, Korea. His research interests include cryptographic algorithms, protocols, and applications.

Ki Song Yoon received his BS degree in shipbuilding engineering in 1984 from Pusan National University, Pusan, Korea. He received his MS and PhD degrees in computer engineering from City University of New York, New York, USA, in 1988 and 1993, respectively. Since 1993, he has been working as a principal researcher with Electronics and Telecommunications Research Institute (ETRI), Korea. His research interests include network protocols and applications.