Protection of MPEG-2 Multicast Streaming in an IP Set-Top Box Environment

Seong Oun Hwang, Jeong Hyon Kim, Do Won Nam, and Ki Song Yoon

The widespread use of the Internet has led to the problem of intellectual property and copyright infringement. Digital rights management (DRM) technologies have been developed to protect digital content items. Digital content can be classified into static content (for example, text or media files) and dynamic content (for example, VOD or multicast streams). This paper deals with the protection of a multicast stream on set-top boxes connected to an IP network. In this paper, we examine the following design and architectural issues to be considered when applying DRM functions to multicast streaming service environments: transparent streaming service and large-scale user environments. To address the transparency issue, we introduce a 'selective encryption scheme'. To address the second issue, a 'key packet insertion scheme' and 'hierarchical key management scheme' are introduced. Based on the above design and architecture, we developed a prototype of a multicasting DRM system. The analysis of our implementation shows that it supports transparent and scalable DRM multicasting service in a large-scale user environment.

Keywords: Copy protection, DRM, multicast.

Manuscript received Feb. 16, 2005; revised May 31, 2005.

Seong Oun Hwang (phone: + 82 42 860 4990, email: sohwang@etri.re.kr), Jeong Hyon Kim (email: bonobono@etri.re.kr), Do Won Nam (email: dwnam@etri.re.kr), and Ki Song Yoon (email: ksyoon@etri.re.kr) are with Digital Content Research Division, ETRI, Daejeon, Korea.

I. Introduction

The rise of the Internet has led to great changes in our lives, both physically and psychologically, over a very short period of time. In fact, the Internet has led to the creation of another world, the so-called digital world. It has also greatly facilitated the distribution and exchange of information. However, a number of problems lurk behind the bright side of the Internet. One key problem is the issue of intellectual property and copyright infringement. Digital content by nature is very vulnerable to unauthorized distribution and use. For example, downloaded content at the user's side is easy to copy, so it is susceptible to illegal copying and has brought about a copy protection problem. Digital rights management (DRM) technologies were developed to prevent users from unauthorized copying of digital content, to control the use of digital content, and to enable the development of digital distribution platforms on which innovative business models can be implemented. They were originally based on work carried out as part of the European Commission-funded IMPRIMATUR project (1995-1998) [1], which included the development of a business model for digital content distribution (which later became the business model of MPEG-21 [2]-[4]), and analyses of rights management information (RMI) and watermarking. Besides MPEG-21 [3], numerous DRM standardization organizations appeared around the year 2000, such as Secure Digital Music Initiative (SDMI), Open eBook Forum (OeBF), DVD Forum, Internet Digital Rights Management (IDRM), Digital Object Identifier (DOI), Open Platform Initiative for Multimedia Access (OPIMA), and Common Intrusion Detection Framework (CIDF). Following those organizations, World Wide Web Consortium (W3C), Internet Streaming Media Alliance (ISMA), TV-Anytime, Open Mobile Alliance (OMA), Digital Home Working Group

(DHWG), and Digital Media Project (DMP) were founded. However, almost all the organizations that started in early 2000 including W3C and DHWG stopped their activities, the exception being MPEG-21. TV-Anytime and DVB provided requirement analysis documents and technological architecture specifications but didn't make any further progress. MPEG, OMA, DMP, and ISMA have continued their work up to the present. MPEG-4 Intellectual Property Management and Protection (IPMP) Extensions [5] and MPEG-2 IPMP Extensions [6] became international standards. MPEG IPMP provides standards for protection and management of multimedia content by introducing a 'hooks' architecture and interfaces between IPMP tools. MPEG-21 [3], which is still under development, aims at defining a normative open framework for multimedia delivery and consumption for use by all the players in the delivery and consumption chain. MPEG-21 identifies and defines the mechanisms and elements needed to support the multimedia delivery chain as well as the relationships between and operations supported by them. The functionalities of such a multimedia framework architecture have been grouped into seven elements: digital item declaration, digital item identification and description, content handling and usage, intellectual property management and protection, terminals and networks, and content representation. OMA [7] was formed in June 2002. Its goal is to deliver high quality, open technical specifications based upon market requirements that drive modularity, extensibility, and consistency. In September 2002, OMA released DRM specification version 1.0. The specification concentrates on content packaging and expression of rights and permissions; it does not include strong security mechanisms to protect the content. OMA DRM specification version 2.0 released in February 2004 provides additional features and a significantly higher level of security to protect high-value digital content like mp3 audio files or video clips. DMP [8] was established in October 2003 with the mission to promote the development, deployment, and use of digital media that safeguard the rights of creators to exploit their works, the wish of consumers to fully maximise the benefits of digital media, and the commercial interests of value-chain players to provide products and services. DMP released in April 2005 its Phase I set of specifications that contain a comprehensive technology specification for interoperable digital rights management as well as applications within and across media value chains. The documents address use cases, DRM architecture, interoperable DRM platform, value-chains, registration authorities, and terminology.

There have been many papers about DRM technology. Hwang and others [9], [10] proposed an extended IMPRIMATUR model that supports trust among distribution entities and expresses efficiently multiple steps of content/rights distribution in the real world. Park and colleagues [11] proposed a taxonomy for DRM. They divided the control architecture into eight categories including "no control" and seven control mechanisms that can be applied to DRM architectures. Rosenblatt and others [12] published an excellent book on DRM that addresses all of its aspects including its business models, legal ramifications, standards, and proprietary core technologies. Lee and others [13] proposed four different popular models of content distribution in the real world and also pointed out the weak points of the IMPRIMATUR model from the viewpoint of protecting the rights of distribution participants and supporting the four models. Jeong and colleagues [14] proposed a key management scheme that can deliver the key used to protect a digital content from its packaging point until its consumption point.

Unlike downloaded content, streaming content has avoided the copy protection problem by disallowing local storage at the user's PC. However, due to the appearance of programs such as the VOD Recorder, Hi Net Recorder, and Net Transport that enable users to save streaming content illegally, streaming content is not free from the copy protection problem any more. Content streaming is a technology for real-time transmission and playback of digital multimedia data such as video or music through the Internet. The majority of streaming uses unicast, where a separate copy of the stream is sent to each viewer. However, unicast is very inefficient in terms of bandwidth when media are delivered to a large number of users. In a multicast, only one copy of a stream is sent out from a source and is replicated as needed in the network to reach as many end-users that want it. Multicast is seen as the means of making the Internet suitable for streaming services and of lowering the costs of distribution.

Recently, researches on DRM in the content streaming environment have been done, most of which cover only unicast streaming. One of these researches is being done by ISMA. ISMA, a non-profit organization whose mission is to accelerate the adoption and deployment of open standards for streaming rich media content such as video, audio, and associated data over Internet protocols, released ISMA Encryption & Authentication Specification V. 1.0 [15] in February, 2004. The purpose of ISMA Encryption & Authentication Specification (also called 'ISMACryp') is to provide interoperability between ISMA-compliant [16] streaming servers and players, when DRM is added in an ISMA-compliant environment. ISMA Encryption & Authentication Specification describes how to encrypt, authenticate and packetize MPEG-4 contents. There are many companies that provide proprietary DRM technology to protect streaming contents. Microsoft [17], Widevine [18], and Verimatrix [19] released their products to provide DRM functions in a video-on-demand (VOD) service environment where media is streamed in a unicast. Although

research on VOD unicast streaming DRM has been done actively, DRM research in a multicast streaming environment has been partly done by a few companies [20], [21]. Their approaches are very limited in a multicasting environment. Their key servers issue keys per user just before a user gets a multicast streaming service. Therefore, a sudden increase of joining users in a multicast streaming service might cause substantial overhead or system failure to key servers. This licensing mechanism is also vulnerable to users' frequent changing of channels. Here we have a strong need to develop DRM functions considering the features of a multicasting service environment: large number of user groups, real-time service requirements, support of a live stream, frequent changing of channels, and size dynamics of users joining/leaving channels at a particular time.

The remainder of this paper is organized as follows. In section II, we provide an overview of the proposed DRM system. The design principles and architecture issues related to a multicasting DRM environment are discussed. The key architecture and key update mechanism to enhance security and performance of the proposed system are also introduced. Section III presents implementation details of system components comprising our prototype DRM system. Section IV presents an analysis of our system. Section V concludes the paper with a discussion of the contribution of the present paper and future work.

II. System Overview

The proposed DRM system is one that prevents the illegal use of content and controls the use of content according to legally purchased usage rights. DRM is applied to MPEG-2 media content in a set-top box (STB), which is connected to an IP network such as ADSL, VDSL, or Ethernet. A decoder and player are installed on the STB to receive and process streamed data from streaming servers. Decoded data are displayed on a TV set connected to the STB. This section discusses major design considerations for a DRM system in such an environment.

1. Design Considerations

- Support of DRM independent of existing streaming systems

DRM systems should be designed to apply DRM functions easily without major modifications of existing encoders/decoders, stream servers, and streaming players. It should be applicable to all streaming servers that support MPEG-2 standards and standard streaming protocols such as real-time transport protocol (RTP) [22] and real-time streaming protocol (RTSP) [23]. To achieve this, all data created during the packaging process should be inserted appropriately without breaking the MEPG-2 standard format. Packaging is the process to transform original contents into a protected distribution format by binding content with metadata and using a copy protection mechanism (for example, encryption). When an appropriate insertion method is not available, it should be stored separately and appropriate binding mechanisms should be provided.

- Transparent (seamless) streaming service

Any DRM-enabled streaming service should be indistinguishable from normal streaming services with no DRM applied. Any degradation of performance or display quality should not occur at the user's end. VCR functions such as fast-forward or rewind should be supported the same way as they are in non-DRM streaming systems. Real-time unpackaging is critical to performance. To alleviate a delay time occurring during the unpackaging process, we introduce a *selective encryption scheme* that allows us to select portions of a multimedia data stream for encryption.

- Toolkit-embedding approach

Under streaming system environments, it is not easy to provide DRM functions as a form of standalone systems or packages. To support and comply with existing streaming servers and players, an embedded toolkit approach is safer and more easily applicable than the system approach. In addition, it gives more flexibility in supporting DRM functions in a variety of streaming service models.

- Support of a large-scale user environment

A multicast streaming service shares some of the following features of a broadcasting service: a large-scale group of concurrent users should be supported seamlessly and when users change channels, and real-time switching of channels without any recognizable delay should be provided at the user's viewpoint. Existing approaches deployed by most VOD DRM systems and a few multicast DRM systems [20], [21] using a separate key transport channel per user are not appropriate in a multicast streaming service environment because a sudden increase of users joining in a multicast streaming service usually causes substantial overhead or a system failure at the key server. To deal with this problem, we insert key packets into the multicast streaming channel rather than opening separate key transport channels per user (the so-called *key packet insertion scheme*).

- Minimization of key update overhead

A DRM system should be designed to make as small as possible the number and quantity of key transmissions that are needed to rekey a multicast group. The design should also

minimize the storage requirements for a multicast group and support a scalable group key update operation. The most critical point is that the key update process should be done real-time enough so that it should not cause any delay or degradation during a multicast streaming service. To achieve efficiency of the key update process, we introduce a *hierarchical key management scheme* consisting of media keys, channel keys, and package keys.

- End-to-end security

Advanced encryption key distribution techniques should ensure that content is only delivered to authorized users and is used under the authorized control of the DRM facilities.

- Support of live real-time streaming

Unlike VOD streaming services, multicast service usually entails live streaming service. To support DRM functions in the case of live streaming, we adopted a real-time encryption mechanism using a DRM multicaster. Therefore a streaming server sends packets in a multicast to a particular internal multicasting address that our DRM multicaster listens to. To support this, our DRM multicaster is designed to encrypt incoming packets in real-time and send them out to external multicasting addresses.

- Security requirements

Keys should have a predetermined lifetime and be periodically refreshed to provide a high level of security. Key materials should be delivered securely to members of the group. Key mechanisms should also support secure recovering from a compromise of some or all of the key material. To achieve these goals, we provide a key interface to set a key lifetime and key update mechanism based on the *current key-next key scheme*.

2. Functional Architecture

Figure 1 shows the functional architecture of the proposed system. The content portal authenticates a user and authorizes the user to use a service. The key management server (KMS) issues keys to allow the user to play the protected stream. A streaming server streams the requested content using a streaming protocol to the internal multicast address using the multicast channel. DRM multicaster encrypts incoming packets from a streaming server in real-time and streams them to the users using the multicast channel.

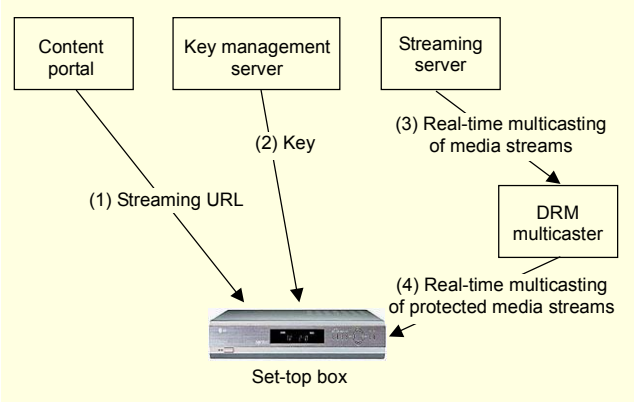


Fig. 1. Functional architecture.

3. Key Architecture

Keys used in multicast key management are largely classified into two kinds of keys, a service-related one and a subscriber-related one. Service-related keys consist of a media key, channel key, and package key. Subscriber-related keys include an STB secret key and broadcast key. Figure 2 shows the hierarchical structure of the keys. These keys use the advanced encryption standard (AES) [24] encryption algorithm with a 128-bit key. Note that the term 'group key' is used to collectively refer to the channel key and package key, hereafter.

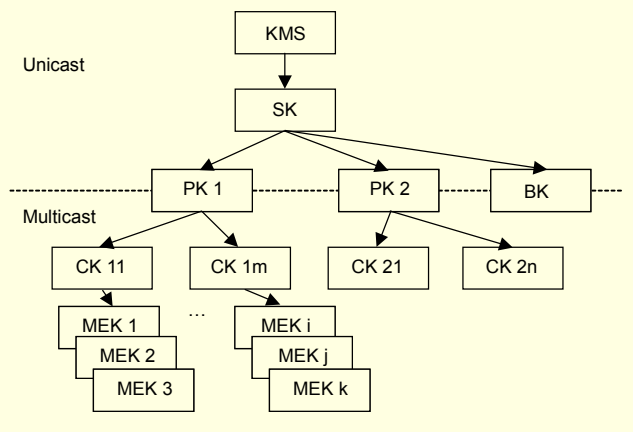


Fig. 2. Hierarchical key structure.

- Media key (MEK)

The media key is used to encrypt a media stream transmitted through a channel. The media key is encrypted under the corresponding channel key and then transmitted to the streaming player as the packet containing the media key is inserted in the multicast streaming channel.

- Channel key (CK)

The channel key is assigned to each broadcast channel and used to encrypt the media keys. The channel key is encrypted under the corresponding package key and then transmitted to the streaming player as the packet containing the channel key is inserted in the multicast streaming channel.

- Package key (PK)

The package key is assigned to each package that consists of a number of channels and is used to encrypt the channel keys.

The package key serves as an access control to a package, that is, a set of channels belonging to a particular package. The package key is encrypted under the receiving STB secret key and then transmitted to the streaming player through the unicast key transport channel when a new user subscribes to a service or when the existing package key expires.

- STB secret key (SK)

The STB secret key is used to encrypt information, for example, a package key that should be transmitted securely to each individual set-top box. Currently, we use a symmetric key scheme as the STB secret key considering the limited performance of a set-top box. The STB secret key is generated when DRM modules are installed on a set-top box. To achieve a higher security level of key transmission, a public key mechanism such as RSA [25] can be deployed without much modification to the proposed architecture.

- Broadcast key (BK)

The broadcast key is used to encrypt information, for example, rekeying a message that should be transmitted to all the users.

4. Service Scenario for MPEG-2 Multicasting Streaming Service

Figure 3 shows the configuration of functional actors. Before servicing users, a content service provider registers the original content to the content portal where a user can browse and select their own content. The original content itself is registered to the streaming server. The streaming server sends multicast streams to the DRM multicaster. The DRM multicaster encrypts incoming streams real-time using keys provided from the KMS and sends encrypted multicast streams to the outside multicast addresses. The KMS provides keys (package/channel keys) to the DRM multicaster and manages the user's package key. To obtain service, a user should subscribe to a particular package that is a set of channels. Note that the content portal consists of a multicast service server and subscriber management server,

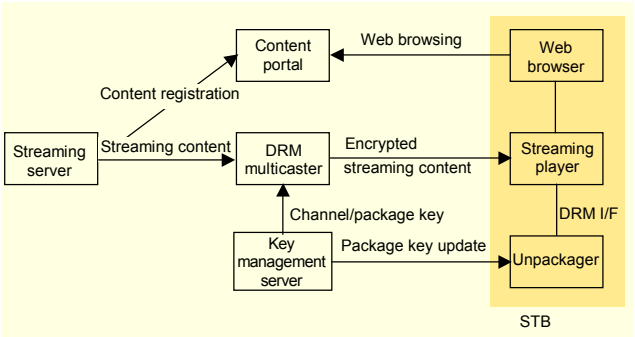


Fig. 3. Configuration of functional actors.

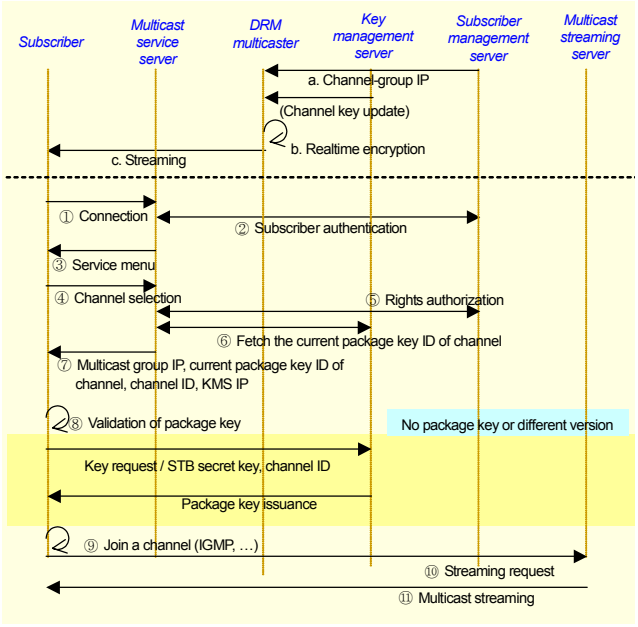


Fig. 4. Process flow in MPEG-2 multicast streaming service.

which are shown in Fig. 4.

The following outlines the process flow in our MPEG-2 multicast streaming service scenario, as illustrated in Fig. 4.

① Service connection

If a user turns on the STB and logs on to the multicast service server, the STB displays the initial service menu.

② Subscriber authentication

A user logs on to a multicast service using his/her user-id and password. The multicast service server authenticates the user by consulting the subscriber's information (for example, usage history, payment status, and so on) from the subscriber management server.

③ Service menu

After the user's authentication is finished, the multicast service server transmits a service menu to the user.

④ Channel selection

A user selects a broadcasting channel from the channel list displayed on a TV set connected to the STB.

⑤ Rights authorization

The multicast service server checks with the subscriber management server to see that the requesting user already subscribed to the corresponding service.

⑥ Fetch the current package key ID of the channel

The multicast service server receives the current package key ID of the channel from the key management server.

⑦ Multicast group-IP, current package key ID of the channel, channel ID, and KMS IP

The multicast service server receives a multicast group-IP from a middleware subsystem of the streaming system. The multicast service server transmits information such as a multicast group-IP, current package key ID of the channel, channel ID, and KMS IP address to the user.

⑧ Validation of package key

A streaming player checks whether the package key stored on the STB is valid by comparing its version with the version of the package key that was received from multicast service server. If those two versions are different, the existing package key on the STB is determined to no longer be valid. In case the package key is not valid or there is no package key, a streaming player requests the issuing of a new package key by sending an STB secret key and package ID through the unicast key transport channel. The key management server generates a new package key and sends it encrypted under the receiving STB secret key through the unicast key transport channel. A streaming player receives it, decrypts it using its secret key, and keeps it on a secure DB, a tamper-resistant secure storage space in the STB.

⑨ Join a channel

A streaming player requests a join operation to the corresponding channel.

⑩ Streaming request

A streaming player requests a stream from the streaming server.

⑪ Multicast streaming

A streaming server starts multicast streaming.

5. Key Update Process

Figure 5 outlines the key update process flow in our MPEG-2 multicast streaming service scenario. As we mentioned earlier, key packets such as channel keys and media keys are transmitted to the set-top box's side along the multicast media channel. Although package keys are updated through a separate key channel (for example, when a new user joins a service or when an existing package key expires), channel keys and media keys are updated through the multicast media channel. Therefore, there exist different versions of channel keys on the channel: one is a currently valid key and the other is a key that will be used when the current valid key expires. Therefore, a key synchronization problem of identifying the current key might happen. To solve the problem, we use the *current key – next key scheme* : The DRM multicaster sends key packets where the currently valid key version is set to the current key field, and the key to be used on the next time slot is set to the next key field. Upon receiving these packets, a set-top box retrieves key information and sets its current, next key field on the memory. When receiving a key update signal (related to this, refer to section III.2, unpackaging), a set-top box sets its current key field to its next key field value, and the next key field to the current key value retrieved from incoming key packets.

① Request package key information

The DRM information processing block requests package key information such as package key identifier and package key version to the package key processing block.

② Package key information

Upon request of package key information, the package key processing block gets it from the secure DB.

③ Check package key

After being received from the secure DB, the package key processing block checks the validity of the package key. In case there is no key, old key, or key update, proceed with steps ④ through ⑧. In case there is a valid package key, proceed with steps ⑧ through ⑨.

④ Request key

The package key processing block requests a package key from the key management server by sending a package key identifier, channel identifier, and STB secret key.

⑤ Transmit key

The key management server returns (package key version, package key encrypted under STB secret key) back to the package key processing block.

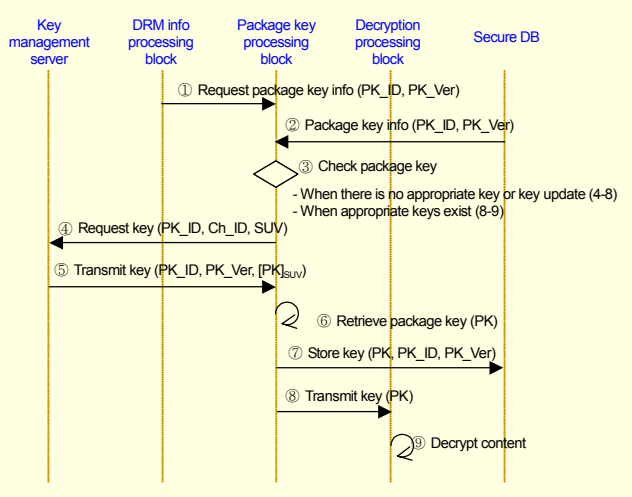


Fig. 5. Process flow in key update.

⑥ Retrieve package key

The package key processing block retrieves a package key by decrypting the transmitted package key using the STB secret key.

⑦ Store key

The package key processing block stores the package key information such as package key and its version at a secure DB.

⑧ Transmit key

The package key processing block sends the package key to the decryption block.

⑨ Decrypt content

The decryption block first decrypts the channel key using the package key. Using the channel key, the decryption block then decrypts the media key that was actually used to decrypt the encrypted media packet.

III. System Details

This section covers details regarding the packaging toolkit and unpackaging toolkit. The packaging is done on the server's side and unpackaging on the client's side. The packaging toolkit largely consists of DRM multicaster and key management server. The unpackaging toolkit consists of DRM client and secure DB. Each module was implemented based on the design principles and architecture introduced in section II.

1. Packaging

The packaging toolkit receives multicast packets from streaming servers or other multicasting modules, encrypts TS packets, and sends them to a user group. The packaging toolkit consists of the following modules shown in Fig. 6.

- Multicast (UDP) packet receiving block

It stores payloads (TS packet) of the received packets in the buffer.

- Multicast (UDP) packet sending block

It reads TS packets from the buffer and sends them as multicast packets to the destination IP address of the channel's group.

- Middleware communication block

It requests and receives the group IP information of the corresponding channel. The group IP information is sent to the multicast packet sending block.

- Key management server communication block

It requests/receives channel keys and package keys of the corresponding channel to/from the key management server. It also keeps information such as channel, channel keys, and package keys. Those keys are sent to the real-time encryption block.

- Key period setting block

It provides a user interface for
- setting the update period of channel keys and
- setting how often the channel keys should be sent.

According to these settings, it sends an event request of the keys update to the key management server.

- Real-time encryption block

It reads TS packets from the receiving buffer, encrypts and writes them on the sending buffer.

Encryption Procedure

Step 1) The real-time encryption block obtains a channel key and a package key, and encrypts the channel key with the

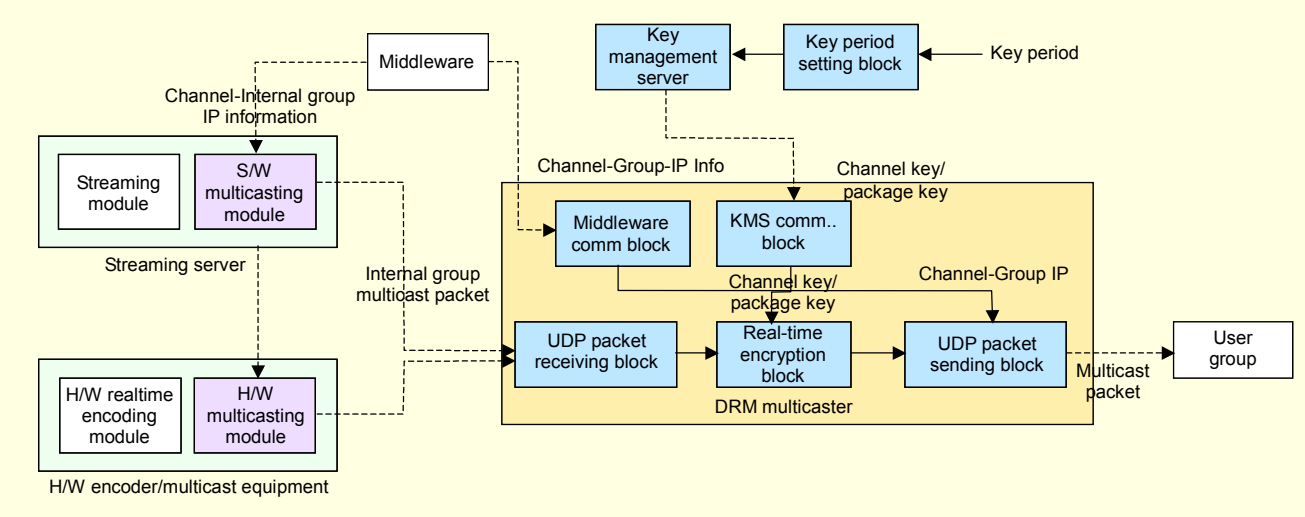


Fig. 6. Packaging process.

package key.

Step 2) It reads in TS packets from the receiving buffer in order. It reads on until a program association table (PAT) TS packet whose PID is '0' appears. Packets located before the PAT are bypassed to the multicast packet sending block.

Step 3) After analyzing the PAT, it retrieves the program map table's (PMT) PID and writes the PAT packet to a sending buffer.

Step 4) It reads in TS packets from the receiving buffer until a TS packet whose PID equals the PMT's PID appears. Packets read before the PMT are bypassed to the multicast packet sending block.

Step 5) It analyzes the PMT packet and obtains both the PIDs of video packets and audio packets, respectively.

Step 6) It inserts new DRM descriptors as shown in Table 1 into the description area of the PMT and writes the packets on the sending buffer. Descriptors are used to describe information about programs and elements that constitute programs. Table 1 shows DRM descriptors to be added within the structure of the PMT. Three new descriptors such as MEK_descriptor, CK_Update_descriptor, and PK_Update_descriptor are inserted into the original PMT.

Step 7) It generates key packets containing key information, that is, media keys encrypted under channel keys, and channel keys encrypted under the package key, and writes them to the sending buffer (key packet insertion). Generated key packets include MEK for the media key, as shown in Table 2, CK_Update for the channel key, as shown in Table 3, and PK_Update for the package key, as shown in Table 4.

Step 8) It reads packets at the receiving buffer. If the packet's PID equals a video PID or audio PID, it performs an encryption. Otherwise, it is bypassed to the sending buffer.

8-1) Encryption is done on the TS packet payload, that is, the TS packet header is excluded from encryption.

8-2) It sets the scrambling control bit of the encrypted TS packet to '11'.

Step 9) If the packet's PID is '0', that is, the packet is a PAT, we repeat steps 2 through 8. The reason we analyze the PAT and PMT again is because if a program is changed (updated), its PAT and PMT information are changed.

The encryption algorithm is AES-ECB (key: 128 bits, data block: 128 bits) and the encryption option supported is as follows:

- video stream encryption
- audio stream encryption
- video key frame encryption

Audio Packet Encryption Procedure

Step 1) The real-time encryption block reads in a TS packet.

Table 1. PMT structure.

Syntax	Description
MEK_descriptor()	Media key
CK_Update_descriptor()	Channel key
PK_Update_descriptor()	Package key

Table 2. The structure of MEK packet.

Syntax	Description
table_id	0x90 or 0x91
section_syntax_indicator	'0'
private_indicator	'0'
reserved	'0'
private_section_length	49 (byte)
CK_version	current CK version
current_MEK	current MEK encrypted under CK
next_MEK	next MEK encrypted under CK
enc_size	encryption size within a packet

Table 3. The structure of CK_Update packet.

Syntax	Description
table_id	0x92 or 0x93
section_syntax_indicator	'0'
private_indicator	'0'
reserved	'0'
private_section_length	64 (byte)
PID	package ID
current_CK_version	current_CK version
content_CK	current CK encrypted under PK
next_CK_version	next CK version
next_CK	next CK encrypted under PK

Table 4. The structure of PK_Update packet.

Syntax	Description
table_id	0x94
section_syntax_indicator	'0'
private_indicator	'0'
reserved	'0'
private_section_length	32 (byte)
PID	package ID
next_PK_version	next PK version

Step 2) It calculates the starting point of the packet's payload. To reduce the time it takes to calculate the header length during decryption, TS packets containing a PES packet header are excluded from encryption.

Step 3) It checks whether the audio frame header exists.

3-1) If it exists, the packet is sent to the output without encryption.

3-2) If it doesn't exist, proceed to the next step.

Step 4) Encryption starts where the payload of the TS packet begins. It sets the scrambling-control bit to '11'.

Step 5) It checks whether there exist encrypted byte streams that are the same with the audio frame header.

5-1) If they exist, encryption is cancelled and the original packet is sent to the output.

5-2) If they don't exist, the encrypted packet is sent to the output.

Video Packet Encryption Procedure

Step 1) The real-time encryption block reads in a TS packet.

Step 2) It calculates the starting point of the packet's payload.

Step 3) It checks whether the video frame header exists.

3-1) If a video frame header exists and the option is key-frame-only encryption, the packet with a picture start code is sent to the output without encryption.

3-2) If a video frame header does not exist, and a PES header exists in step 2, the packet is sent to the output without encryption.

3-3) If neither a video frame header nor PES header exists, proceed to the next step.

Step 4) Encryption starts where the payload of the TS packet begins. It sets the scrambling-control bit to '11'.

Step 5) It checks whether there exist encrypted byte streams that are the same with the video frame header.

5-1) If they exist, encryption is cancelled and the original packet is sent to the output.

5-2) If they don't exist, the encrypted packet is sent to the output.

Note that we do not encrypt the whole payload of the TS packets. TS packets containing video (audio) sequence headers are excluded from encryption because streaming servers usually use them to support trick play functions such as fast-forward and rewind. When encrypted byte streams appear whose byte codes equal the video or audio sequence header, encryption is cancelled and the encrypted byte streams are replaced by the original byte streams.

Table 5 shows encryption options supported by the proposed system. When video I-frame-only encryption is selected among the encryption options, it checks whether a video frame header is an I-frame and retrieves TS packets within the I-frame. When encryption within a TS packet is selected, it only encrypts data of specific size within a payload of a TS packet. Our selective encryption is different from the MPEG-4 IPMP Extension specification [5]: The specification of our selective encryption deals with whether or not a specific part of a payload is encrypted, while MPEG-4 IPMP Extension selective encryption usually deals with whether or not specific bit stream syntax elements (motion vectors, DCT, audio codewords, and so on) are encrypted. Our selective encryption is similar to ISMACryp selective encryption that deals with whether or not a sample is encrypted. The reason behind why we chose our selective encryption is that if we chose IPMP Extension selective encryption, we should make additional modifications on the decoders that are usually implemented in HW in set-top boxes, which contradicts our design principle, that is, support of DRM independent of existing streaming systems without major modifications. Readers who are more interested in other selective encryption schemes should refer to survey paper [26]. According to the packet skip count, it selectively skips the encryption of packets. If the packet skip count is set to 5, it encrypts every 5th packet among all the input packets.

Table 5. Encryption options.

Options	Description
Media type	- video-only encryption - audio-only encryption - video/audio encryption
Video frame selection	- I-frame-only encryption - Entire frame encryption
Encryption size within TS packet	- 16 to 128 bytes within a payload of a TS packet
Packet skip count	The ratio of packets to be encrypted among incoming packets

• Key management server (KMS)

The KMS generates keys and updates them periodically or upon requests from the DRM multicaster. It also receives package key requests from and sends them to the package key processing block. It keeps the following key information:

- channel ID, channel key, effective period of channel key
- package ID, package key, effective period of channel key, the list of the channel IDs comprising the package.

2. Unpackaging

The Unpackaging toolkit consists of the following modules as shown in Fig. 7

• DRM information processing block

The DRM information processing block receives target channel information such as multicast IP, package ID, channel ID, and KMS IP and transfers the data to its corresponding processing block. It also requests a join operation to the target (or corresponding) service using a multicast group IP.

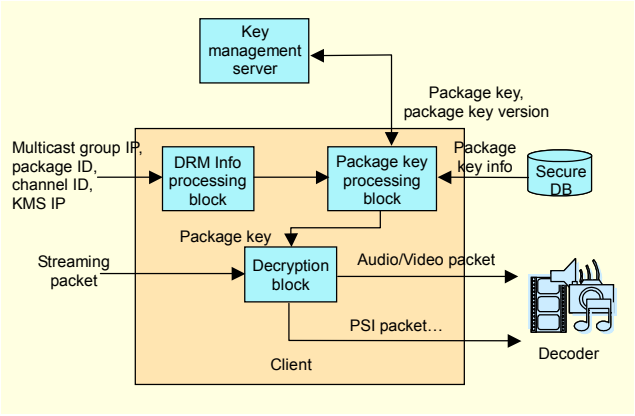


Fig. 7. Unpackaging process.

• Package key processing block

Using the package ID received from DRM information processing block, the package key processing block checks the version information of the package key stored in the secure DB. If those two versions are equal (or identical), it obtains the package key from the secure DB and sends it to the decryption block. If they are not equal, it requests a package key from the key management server, using the KMS IP that was received from the DRM information processing block. When requesting a packaging key from the KMS, it uses information such as package ID, channel ID, and STB secret key. An STB secret key is used to identify an STB to the KMS and to set up a secure key transport channel between the KMS and STB secure DB.

• Secure DB

The Secure DB is a secure storage area within a set-top box. The secure DB manager generates an STB secret key (AES-128 bit) and registers it with KMS as well as keeping it on a secure DB. A package key is transferred to an STB under encryption via a set-top box's secret key. Using the STB secret key, the secure DB manager decrypts the package key that was originally encrypted with the STB secret key, stores the decrypted package key at the secure DB, and sends it to the decryption block.

• Decryption block

The decryption block analyzes the program specific information (PSI) and decrypts video and audio packets using channel keys.

Decryption Procedure

Step 1) The decryption block reads in TS packets continuously from the receiving buffer until a TS packet whose program ID (PID) is '0' (that is, PAT) appears. Packets read before the PMT are bypassed to the decoder (or decoder's buffer).

Step 2) It analyzes the PAT packet and obtains the PID of the PMT packet.

Step 3) It reads in TS packets continuously from the receiving buffer until a packet appears whose PID equals the PMT's PID. Packets read before the PMT are bypassed to the decoder (decoder's buffer).

Step 4) It analyzes the PMT packet and obtains a video PID and an audio PID. It also analyzes the DRM descriptor and obtains the PID of the packet that contains channel key information.

Step 5) It reads in TS packets continuously from the receiving buffer until a packet appears whose PID equals the key packet's PID. Packets read before the key packet is bypassed to the decoder.

Step 6) It retrieves the channel key by decrypting the received channel key under the package key. The channel key exists only on the memory of the STB and is accessible from the decryption block.

Step 7) It reads in packets from the receiving buffer. If its PID equals a video PID or audio PID, encryption is done using the media key that is obtained by decrypting under the channel key. In the case of a video (or audio) packet, it checks the scrambling-control bit. If it is '11', decryption is done and the decrypted data are sent to the decoder. Otherwise, it is bypassed to the decoder without decryption.

Step 8) In the case where PID is '0', we repeat steps 2 through 7.

In step 6, the current key of a set-top box becomes obsolete, and the next key of a set-top box becomes the current key by which decryption is done. The next key of a set-top box are replaced by the current key that was retrieved from a new key packet. In our implementation, a set-top box does not need to retrieve all the key packets to check whether key updates happen. To signal that a key update event happened, our DRM multicaster uses two key packet ids alternately: 0x90, 0x91 for media keys, and 0x92, 0x93 for channel keys. Therefore, when the key packet id kept in a set-top box is '90' and the incoming key packet id is '91', it indicates that a media key update has happened, so the set-top box retrieves keys from the key packet and updates media keys.

IV. Analysis

This section analyzed how the proposed system achieved the goals that were set forth in section II. Based on the above modules, we constructed a demonstration site that consisted of a streaming server, a DRM multicaster, key management server, two set-top boxes, and web server as shown in Fig. 8.

Two set-top boxes are connected to two separate subnets that

are also connected to a servers' network via a multicast-supporting router. Through this demonstration system, we first confirmed that the system worked well as integrated with an existing streaming server called CasterNets without any major modification to it. In a security point of view, the proposed system achieved the security requirements set before: end-to-end security and a periodic rekeying mechanism. Content is only delivered to authorized users whose devices have appropriate package keys, which will be used to restore the channel keys and media keys. Users who don't belong to a specific package group, and therefore don't get a package key, cannot decrypt the received streaming content even when they can access them. We also control the security level of the proposed system by supporting a secure key update mechanism based on the current key-next key scheme.

We also got some performance data as shown in Tables 6 and Figures 10, 11: a comparison of the level of encryption according to various encryption options, the multicaster's performance, and overhead on the set-top box's side. Table 6 shows that as the packet skip count decreases and the encryption size within the TS packet increases, the resulted encrypted streams become less recognizable. Figure 9 shows a screenshot of the original and encrypted streams described in Table 6.

Figure 10 shows how a DRM multicaster works as the number of channels increases. The system specification of the multicaster is as follows: CPU (Pentium IV 2.0 GHz), Memory (512 M), and NIC (100 M) on Windows 2003 Server. As sample content, we used 4 Mbps MPEG-2 TS contents. In Fig. 10, when there are zero channels, the usage (occupancy) rates each denote the basic CPU and networking rates. According to our experiment, we expect that when using a generic PC as a DRM multicaster, we can provide a stable multicast service on up to 10 channels.

Figure 11 shows the distribution of packet decryption times of 4 Mbps streaming content on the set-top's side. The system specification of the set-top box is VIA C3 800 MHz and 128 MB of memory. The encryption option is all audio/video frame encryption with packet skip count = 0. From Fig. 11, we find

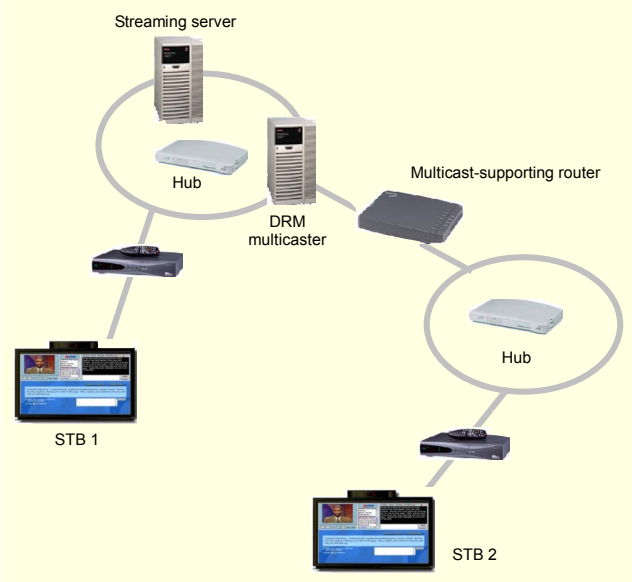


Fig. 8. Configuration of a demo site.

Table 6. Encrypted results.

	Audio	Video		Packet skip count		Payload encryption size
		All-frame	I-frame	audio	video	
(a)	X	X	X	X	X	X
(b)	X		O	X	10	32 byte
(c)	O		O	5	5	64 byte
(d)	O	O		0	0	128 byte

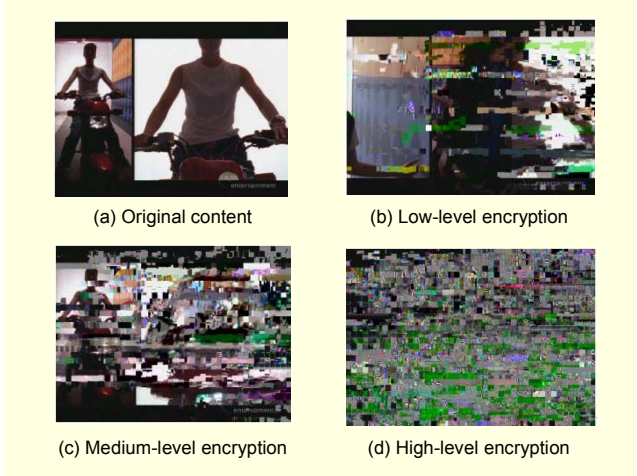


Fig. 9. Encrypted results.

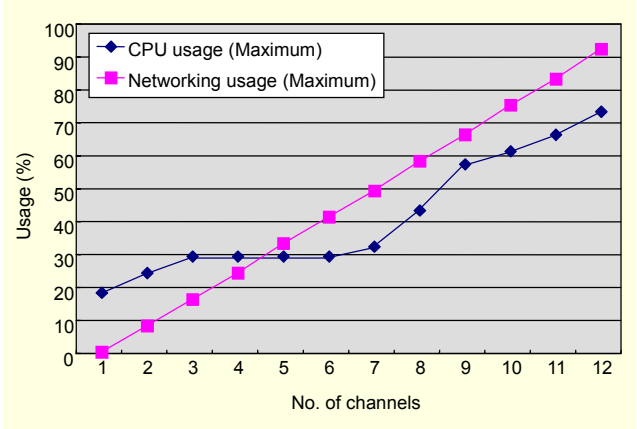


Fig. 10. The usages of CPU and networking.

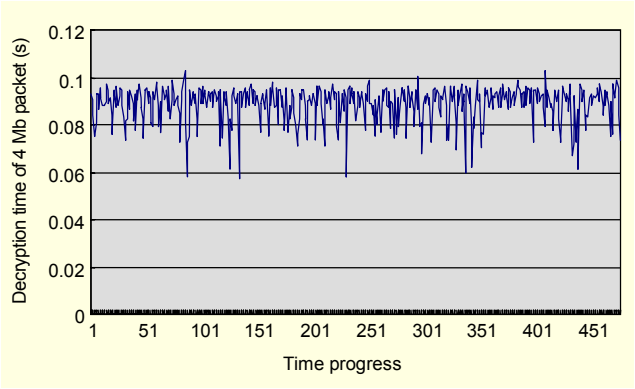


Fig. 11. Distribution of packet decryption time.

that the time to decrypt takes less than 0.1 seconds when processing 4 Mb during one second. Considering that the time to decode (play) is 0.2 seconds, we use only 0.3 seconds per one second. As the CPU on the set-top box is in an idle state during the remaining 0.7 seconds, real time performance during the unpackaging process is guaranteed.

From our performance testing, we confirmed that no degradation of performance or display quality occurred on the client's side.

We conclude the analysis section by addressing the scalability issues, comparing our scheme with another scheme [20], [21]: The total key update time increases linearly as the number of users N increases in the other scheme, whereas in our scheme it takes O(1) due to our key packet insertion scheme. However, we also observed that our key packet insertion scheme increased the total streaming data size by 0.003 % per second because of the inserted key packets. We note that the size of the inserted key packets relates to channel switching, but not to the number of users. The amount of key material transmitted during the rekey process takes O(N) in the other scheme. However, it takes O(PN) in ours, where PN (<< N) is the number of packages that streaming service providers can determine freely according to their service policies and strategies.

V. Conclusion

In this paper, a new multicasting DRM system has been proposed. The distinctive feature of the proposed system is that it supports transparent and scalable DRM multicasting service in a large-scale user environment. To achieve these features, we introduced a selective encryption scheme that allows us to control the encryption overhead on the client's side. We deployed a key packet insertion scheme to transport keys to large-scale user groups securely and efficiently. A hierarchical key management scheme was also introduced to reduce key update overhead caused by periodic key updates. To check whether the system works as designed and analyze the proposed system, we built a real DRM multicast streaming service environment. We confirmed the following: the key transport process and key update process worked well, and encrypted multicast streams are unpackaged on the client's side without any delay or degradation of quality. To estimate how many channels our DRM multicaster supports, we monitored the CPU and network usage rates as the number of channels increased. Our experiment results suggest that our DRM multicaster can provide a stable multicast service up to 10 channels. To increase the number of channels and decrease the CPU usage rate, we plan to increase NIC from one to two or more. We also plan to migrate our DRM multicaster platform from Windows 2003 to a more network-optimized platform (for example, FreeBSD) to improve the performance of the DRM multicaster. Furthermore, the distribution of packet decryption times of 4 Mbps streaming content on the set-top box's side shows that real-time performance during the unpackaging process is guaranteed.

References

[1] *IMPRIMATUR Business Model*, Version 2.1, June 1999.

[2] S. Hwang, J. Kim, K. Yoon, and M. Kim, "Trends of MPEG-21 IPMP Standardization," *Electronics and Telecommunications Trends*, vol. 17, no. 4, Aug. 2002, pp. 51-64.

[3] ISO/IEC JTC1/ SC29/WG11 MPEG/N 3939, *Information Technology — Multimedia Framework (MPEG-21) — Part 1: Vision, Technologies and Strategy*, Jan. 2001.

[4] ISO/IEC 21000-4, *Information Technology – Coding of Moving Pictures and Audio: Intellectual Property Management and Protection in MPEG Standards*, 2001.

[5] ISO/IEC 14496-1:2001/FDAM 3:2003(E), SC 29/WG11 W5282, *Information Technology – Coding of Audio-Visual Objects – Part 1: Systems, AMENDMENT 3: Intellectual Property Management and Protection (IPMP), Extensions,* Dec. 2002.

[6] ISO/IEC 13818-1:2000/FDAM 2:2003(E), SC 29/WG11 W5604, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information – Part 1: Systems, AMENDMENT 2: Support of IPMP on MPEG-2 Systems, Apr. 2003.

[7] OMA, Http://www.openmobilealliance.org.

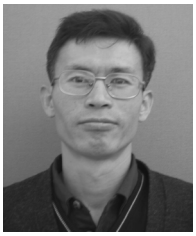
[8] DMP, Http://www.dmpf.org.

[9] S. Hwang, K. Yoon, K. Jun, and K. Lee, "Modeling and Implementation of Digital Rights," *J. of Systems and Software*, vol. 73, no. 3, Nov. 2004, pp. 533-549.

[10] S. Hwang, K. Yoon, and K. Lee, "A Modeling of Multilevel DRM," *IEICE Trans. on Comm.*, vol. E88-B, no. 5, May 2005, pp. 2168-2170.

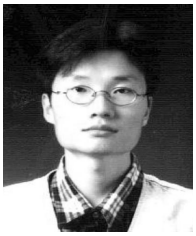
[11] J. Park, R. Sandhu, and J. Schifalacqua, "Security Architectures for

Controlled Digital Information Dissemination," *Proc. of the 16th Annual Computer Security Applications Conf. (ACSAC)*, Dec. 2000, pp. 224-233.
[12] B. Rosenblatt, B. Trippe, and S. Mooney, *Digital Rights Management – Business and Technology*, M&T Books, 2002.
[13] J. Lee, S. Hwang, S. Jeong, K. Yoon, C. Park, and J. Ryou, "A DRM Framework for Distributing Digital Contents through the Internet," *ETRI J.*, vol. 25, no. 6, Dec. 2003, pp. 423-436.
[14] Y. Jeong, K. Yoon, and J. Ryou, "A Trusted Key Management Scheme for Digital Rights Management," *ETRI J.*, vol. 27, no. 1, Feb. 2005, pp. 114-117.
[15] ISMA, *Internet Streaming Media Alliance Encryption and Authentication Specification Version 1.0.*, Feb. 2004.
[16] ISMA, *Internet Streaming Media Alliance Implementation Specification Version 1.0.*, Aug. 2001.
[17] Microsoft, Http://www.microsoft.com.
[18] Widevine, Http://www.widevine.com.
[19] Verimatrix, Http://www.verimatrix.com.
[20] Coretrust, Http://www.coretrust.com.
[21] Sealtronix, Http://www.sealtronix.com.
[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," July 2003.
[23] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," Apr. 1998.
[24] NIST, *Advanced Encryption Standard (AES)*, NIST FIPS 197, Http://csrc.nist.gov/publications/fips/fips197/fips197.pdf, Nov. 2001.
[25] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Comm. of the ACM*, Feb. 1978, pp. 120-126.
[26] Xiliang Liu and A. Eskicioglu, "Selective Encryption of Multimedia Content in Distribution Networks: Challenges and New Directions," *Proc. of the 2nd IASTED Int'l Conf. on Comm., Internet, and Info. Technology (CIIT 2003)*, Nov. 2003.



Seong Oun Hwang received the BS degree in mathematics in 1993 from Seoul National University, the MS degree in computer and communications engineering in 1998 from Pohang University of Science and Technology, and the PhD degree in computer science in 2004 from Korea Advanced Institute of Science and Technology, all in Korea. He worked as a Software Engineer at LG-CNS Systems, Inc., from 1994 to 1996. Since 1998, he has been working as a Senior Member of Engineering Staff at Electronics and Telecommunications Research Institute (ETRI), Korea. His research interests include cryptographic algorithms, protocols, and applications.

Jeong Hyon Kim received the BS and MS degrees in computer science in 1999 from Jeonnam National University, Korea, in 1999 and 2001. Since 2001, she has been working as a Member of Engineering Staff at ETRI. Her research interests include image compression, multimedia telecommunication and applications.



Do Won Nam received the BS degree in computer science in 1995 from Korea Advanced Institute of Science and Technology, and the MS degree in computer and communications engineering in 1998 from Pohang University of Science and Technology. He has been working as a Senior Member of Engineering Staff at ETRI since 2001. His research interests include algorithms and systems for digital rights management and data mining.

Ki Song Yoon received the BS degree in shipbuilding engineering in 1984 from Pusan National University, Pusan, Korea. He received the MS and PhD degrees in computer engineering from City University of New York, USA, in 1988 and 1993. Since 1993, he has been working as a Principal Member of Engineering Staff at ETRI. His research interests include network protocols and applications.