MPEG-4 IPMP Extension

- For Interoperable Protection of Multimedia Content[1]

Ming Ji, SM Shen
Panasonic Singapore Laboratories Pte Ltd,
Blk 1022 Tai Seng Avenue #04-3530 Tai Seng Ind Est, Singapore 534415
mji@psl.com.sg; shen@psl.com.sg
Wenjun Zeng
Dept. of Computer Science, Univ. of Missouri-Columbia, MO 65211
zengw@missouri.edu
Taka Senoh, Takafumi Ueno
Matsushita Electric Industrial Co., Ltd.
Terumasa Aoki, Yasuda Hiroshi, Takuyo Kogure
University of Tokyo

Abstract

To ensure secure content delivery, the Motion Picture Experts Group, MPEG, has dedicated significant effort to the DRM (Digital Rights Management) issues. MPEG is now moving from defining only hooks to proprietary systems (e.g., in MPEG-2, MPEG-4 Version 1) to specifying a more encompassing standard in Intellectual Property Management and Protection (IPMP). MPEG feels that this is necessary in order to achieve MPEG's most important goal: interoperability. The design of the IPMP Extension framework also considers the complexity of the MPEG-4 standard and the diversity of its applications. This architecture leaves the details of the design of IPMP tools in the hands of applications developers, while ensuring the maximum flexibility and security. This paper first briefly describes the background of the development of the MPEG-4 IPMP Extension. It then presents an overview of the MPEG-4 IPMP Extension, including the architecture of MPEG-4 IPMP Extension, the flexible protection signaling, and the secure messaging framework for the communication between the terminal and the tools. Two sample usage scenarios are also provided to illustrate how a MPEG-4 IPMP extension compliant system works.

1. Background and Introduction

1.1 Problems in the Existing DRM Market

With the advent of digital technologies, many new market opportunities have emerged for content owners, content distributors, and consumer electronics/information technology industries. An essential requirement for developing a thriving marketplace is the protection of copyrighted content in digital form. Digital Rights Management (DRM) is a technology that has been developed to protect against the illegal distribution of copyrighted digital content such as music, video or documents. However, there are some problems remained to be solved in the existing DRM market

[1] Authors of this paper have been actively involved in the standardization of MPEG-4 IPMP Extension, and have made significant contribution to the IPMP Extension standards.

The first problem is the lack of interoperability. Different content providers tend to use different protection mechanisms (hence, different DRM systems) to protect and distribute the content. For example, Content Provider A may prefer to use the Advanced Encryption Standard (AES) [15] for encryption, while Content Provider B may prefer to use his own proprietary encryption tool. This results in the lack of interoperability as illustrated in Fig. 1, where Terminal A can not play back content distributed by Content Provider B, and vice versa.

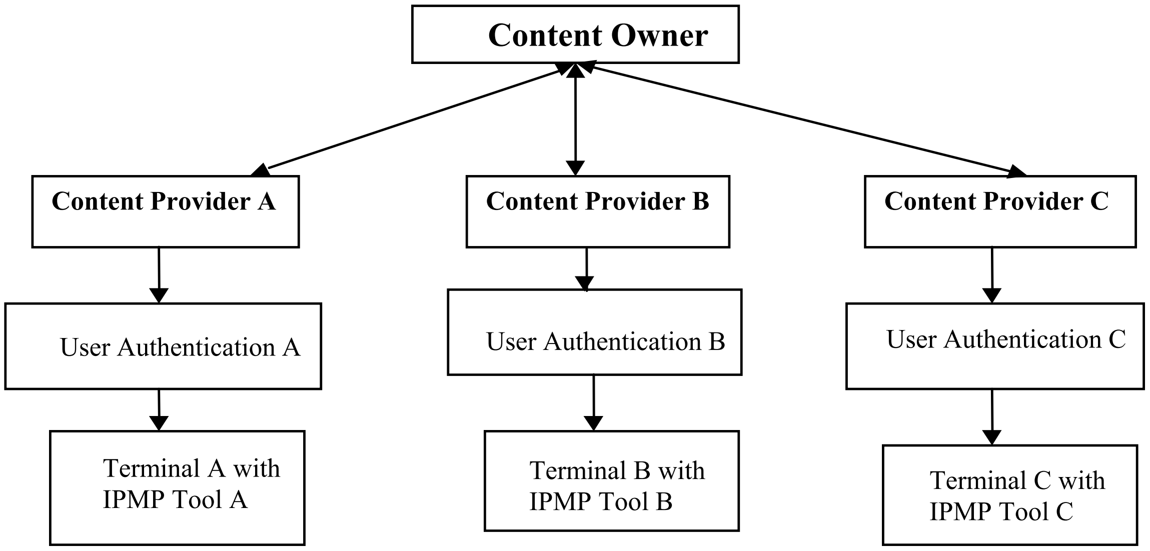

Figure 1: Existing DRM market

The second problem of the existing DRM market is the lack of renewability. Many existing DRM systems are likely to be broken, due to the rapidly growing computer technology. This is one of the serious problems encountered in digital content delivery business. It is therefore desirable to establish a robust and flexible DRM system, where one can easily renew a broken DRM system.

1.2 MPEG-4 IPMP Extension, the Answer to the Problems

The lack of interoperability problem demands an international standardization effort, so that contents can be delivered anytime, and to anywhere in the world. Being able to expect different vendors' content to play on a single player is a big deal. Not having to re-engineer a given player to work with every other IPMP system is an even bigger deal.

With the above considerations, MPEG, the Motion Picture Experts Group, has been pushing for the goal of establishing a Digital Rights Management standard enabling the functionalities of renewability and interoperability. The MPEG specific term for DRM is "Intellectual Property Management and Protection", i.e. IPMP. The latest IPMP standard for MPEG-4 system is the MPEG-4 IPMP Extension (IPMPX) [1].

During the development of the IPMP Extension, a real world scenario that has been discussed intensively, in order to understand more about the scope and the problems that the IPMP Extension should resolve, is the *Gobi Desert Scenario*.

> *Gobi Desert Scenario* -- Living in a rather rainy place, Mr. MPEG loves to go to arid places. The Gobi desert is his favorite. Before leaving, imagine that he loads some protected songs in his Panasonic MIEP (MPEG IPMP Extension Player). His wife does the same on her Philips MIEP, but with different songs. When they are in their tent in the middle of the Gobi desert, Mr. MPEG starts listening to his MIEP. He finds a new hit that he feels great and would like to share it by transferring that song to his wife's MIEP (and, being a rule-abiding guy, he has acquired the rights to do so). Unfortunately, this song has been protected with tools that are new to his wife's MIEP. To make his life harder, there is no Internet connection available in the desert that would allow the required tool to be downloaded to Mrs. MPEG's MIEP. Luckily, being the dictator of MPEG, Mr. MPEG has the power to demand that IPMP Extension supports transferring IPMP tools intended for one device to a device of a different make. This would save the trip because otherwise his wife will start asking why he has spent all those years in MPEG if such a simple thing like moving a song from one MIEP to another is not possible and the discussion is likely to degenerate. This demand, however, would make the lives of the MPEG-4 IPMP committee members miserable, but that is not what Mr. MPEG cares about anyway…

The *Gobi Desert Scenario*, explicitly or implicitly, suggests that several factors be considered in the standardization of the MPEG-4 IPMP.

- There should be a way to signal to the terminal what IPMP tools are required to consume the contents.
- If the required IPMP tools are not available in the terminal, there should be a way to acquire the missing tools from a remote location.
- There should be a way to securely transfer the content and the IPMP tools from one device to another.
- To ensure interoperability, there should be a way to allow different IPMP Tools (potentially from different vendors) to be plugged into the terminal and interact with each other in a normative manner.
- There should be a way to renew the potentially compromised tools.
- There should be a way to specify where and which MPEG-4 content streams the required IPMP tools should be applied, and in what order.
- There should be a way for the terminal to *securely* communicate with the tools (potentially a plug-in), and to enable tools to communicate *securely* with each other.
- There should be a way to convey the IPMP information such as key and rights information to the terminal and to the IPMP tools.
- The terminal should comply to the usage rights associated with the user.
- Should MPEG-4 IPMP standardize the tools?
- Should MPEG-4 IPMP standardize the key management systems?
- Should MPEG-4 IPMP standardize the rights management systems?

These issues need to be addressed carefully and in an elegant way to avoid problems experienced in some previous standardization effort, e.g., some technologies chosen by the DVD Forum [19] and the Secure Digital Music Initiative (SDMI) [20], an industry forum that intended to develop open technology specifications that protect the playing, storing, and distributing of digital music, have been claimed to be hacked. We will show how these considerations have been addressed in MPEG-4 IPMP Extension in the following Sections.

1.3 History of the MPEG-4 IPMP Extension

MPEG started its IPMP effort in the development of MPEG-4. The first attempt is often referred to as the "hooks" approach, where normative syntax is defined in MPEG-4 system to allow the bitstream to carry information that informs the terminal which (of possibly multiple) IPMP system should be used to process the governed objects in compliance with the rules declared by the content provider. The respective IPMP systems themselves were not specified within MPEG-4 [3]. MPEG-4 integrates the "hooks" tightly with the MPEG-4 Systems layer, which makes it possible to build secure MPEG-4 delivery chains in very smart and efficient ways.

This "hooks" model, however, appears to have many significant problems. For example, IPMP systems can be "hooked" into the MPEG-4 terminal, but it can only be done on a proprietary basis. Since the protection is normally required to be associated with some elements of the MPEG-4 terminal, and its behavior cannot be independent of other parts of the MPEG-4 terminal, if the IPMP system is not interoperable, a MPEG-4 terminal with IPMP protection would also become non-interoperable.

As a simple example, if the encryption used to protect the video content is different from one IPMP system to another, the Consumer Electronics (CE) manufacturers would have to build multiple versions of the MPEG-4 terminal to deal with different protection systems used by different content providers. This would significantly increase the cost of building a terminal, and as a result, the consumers would have to bear the high cost. Therefore, the question the MPEG-4 committee faced was whether MPEG can define and standardize an IPMP framework for both the content providers and the CE manufactures to follow, so that IPMP systems can become interoperable.

In the year 2000, a new call for proposal (CfP) [11] was issued. Particularly, it aimed to address the interoperability between different products, often for similar services, as developed within the IPMP framework of the MPEG-4 standard. In addition, with convergence becoming a reality, e.g. through the deployment of broadband Internet access and the start of new services on mobile channels, inter-working between different types of devices and services becomes a more important requirement. The new Call requests submission of proposals that would allow inter-working between different devices and services designed to play secure digital MPEG-4 content from multiple sources in a simple way, e.g. without the need to change the devices.

One issue that particularly needs to be considered when standardizing an IPMP framework in MPEG is the balance between interoperability and security, since these two factors usually contradict each other. Can we standardize every piece of the IPMP system, including a single encryption tool, a single watermarking tool, a single user authentication tool, as well as the key management?

Depending on the scale of the industrial domain and the preference on simplicity or security, one might have different answers to the above question. However, from an international standard (MPEG) point of view, our answer to the above question is no. The first reason is that it will introduce the security issue, e.g., sometimes the security of the video watermarking tool depends on the secrecy of the watermarking algorithm, so standardizing a single watermarking tool is not practical. Furthermore, many DRM systems prefer a black-box key management too. Besides the security issue, the second reason is that we have to take care of flexibility as well as renewability. In the current business environment, there are various contents with different importance levels, which are usually protected using different algorithms (AES, DES (Data Encryption Standard) [18], Triple DES, for example) with different security levels. If we would like the same terminal to be able to consume different contents protected with different algorithms, the IPMP framework to be defined has to be flexible. Once the IPMP framework can deal with the flexibility issue, it will be able to support renewability, which is required for IPMP systems for security reason, since an algorithm typically cannot survive many years of attack. After all, MPEG is targeting a large number of industrial domains with different requirements. MPEG4 IPMP should focus on standardizing the most common framework/base for various target applications.

The CfP on the IPMP Extension resulted in numerous submissions from various industries, including many from the authors of this paper. MPEG's Systems Group has been working with the proponents and started an extension to the MPEG-4 Systems standard in the form of an amendment and a new part of MPEG-4 standard. It has reached FDIS (Final Draft of International Standard) stage in Oct 2002 [1]. A significant part of the standard was contributed by the authors of this paper.

This paper is organized as follows. Section 2 presents an overview of the architecture of the MPEG-4 IPMP Extension. Section 3 and Section 4 detail the core components of the MPEG-4 IPMP Extension. In Section 5, two sample usage scenarios are presented for a MPEG-4 IPMP Extension compliant system. Section 6 concludes the paper.

2. MPEG-4 IPMP Extension Architecture

2.1 Key Concepts

It is important to achieve robustness and flexibility in the interoperable framework of a standard. To achieve the robustness, MPEG-4 IPMP Extension provides the tool renewability, which protects against security breakdown. The flexibility allows the use of various cipher tools, as well as decoding tools. The interoperable framework enables the distribution and consumption of content all over the world. MPEG-4 IPMP Extension defines 5 key elements as described below.

1) IPMP Tools
IPMP tools are modules that perform (one or more) IPMP functions such as authentication, decryption, watermarking, etc. A given IPMP Tool may coordinate other IPMP Tools. Each IPMP Tool has a unique IPMP Tool ID that identifies a Tool in an unambiguous way, at the presentation level or at a universal level.

During the standardization of the IPMP Extension, the MPEG-4 IPMP committee realized that it is not possible to standardize all IPMP Tools due to two main reasons. The first is that different content providers have different preferences on the IPMP Tools as explained in Section 1.1. The second reason is that there are some tools that are difficult to standardize, for example, it's not possible to standardize a video watermarking tool, as there is no proven robust watermarking algorithm yet. With the above considerations, MPEG-4 IPMP Extension is designed to differ from many prior approaches in that it intelligently provides an *open secure* framework allowing tools from different vendors to cooperate with each other.

2) IPMP Descriptors
This is a part of the MPEG-4 object descriptors (OD) that describe how an object can be accessed and decoded. These IPMP Descriptors are used to denote the IPMP Tool that was used to protect the object. An independent registration authority (RA) is used so any party can register its own IPMP Tool and identify this without collisions.

3) IPMP Elementary Stream (ES)
IPMP specific data such as key data, rights data are carried by the IPMP ES. All MPEG objects are represented by elementary streams, which can reference each other. These special elementary streams can be used to convey IPMP specific data. Their syntax and semantics are further specified in MPEG-4 IPMP Extension [1].

4) IPMP Tool List
IPMP Tool list carries the information of the tools required by the terminal to consume the content. It is carried in the Initial Object Descriptor (IOD) of the MPEG-4 system stream. This mechanism enables the terminal to select, manage the tools, or retrieve them when the tools are missing, etc [12].

5) Secure Messaging Framework

The MPEG-4 IPMP Extension framework did not choose the approach of defining functional interfaces, instead, it is based on secure message communication [1]. This is one of the most important concepts in MPEG-4 IPMP Extension. Interaction between the terminal and the IPMP Tools are realized through the messages via a conceptual entity called "Message Router". Syntax and semantics of the messages are clearly defined to facilitate full interoperability. Mutual authentication and secure messages are also introduced to achieve a secure framework. Note that the normal functional interfaces are unlikely to cover various kinds of interfaces for different algorithms, even for the same encryption function. Furthermore, the normal functional interfaces are highly dependent on the operating system and the implementation.

The message based architecture has three advantages over functional interface based architectures. The first is that security can be more easily maintained, as messages are easier to protect in an open framework than the parameters in a function parameter list. The second is that the only entities that need to be concerned with a given message's definition are those that need to generate or act upon a given message, so additional functionality can be created and supported simply through the addition of the required messages. The third is that full interoperability with IPMP tools can be easily achieved by registering the messaging API to a Registration Authority (RA) and carrying the registered API ID in the IPMP_ToolAPI_Config information in the IPMP Descriptor, or by defining a single messaging API by a third-party forum which adopts MPEG-4 IPMP Extension. Note that MPEG is not taking the role of defining a single messaging API, since MPEG is targeting a large number of industrial domains. Individual industrial domain should take MPEG-4 IPMP Extension as a base, and fill in the gap in order to make IPMP Extension truly interoperable.

Note that in the "hooks" approach [3], MPEG-4 IPMP defines how an object is treated and how the IPMP specific data are carried. In other words, (2) and (3) discussed above are included in the "hooks" approach. In the IPMP Extension, (4) and (5) are added while (2) and (3) are further improved, and the concept of IPMP system in IPMP "hooks" is changed to IPMP Tool as discussed in (1). IPMP Extension enhances the original "hooks" approach so that tool renewability and flexibility can be achieved.

Considering the diverse applications (e.g., real time communications, Internet streaming, surveillance, broadband, wireless, studio, DVD, set top box, etc) that MPEG-4 intends to address [5], it is very difficult to have a complete "one-fit-all" solution. For example, as discussed above, it would be very difficult to standardize tools in MPEG, a standardization body whose main mission is to standardize core technologies, rather than meta-data or making business decision. Instead, MPEG-4 chose to standardize a flexible architecture that would allow individual industry to extend the framework and further define their own complete standard to achieve full interoperability, based on the requirements of the individual industry and business consideration. For example, key management and user registration/authentication are not defined in MPEG-4 IPMP Extension. Their implementations are up to the IPMP Tools on top of MPEG-4 IPMP Extension. This enables using different IPMP Tools for different applications, while providing a common framework to facilitate the support of full interoperability.

2.2 Architecture

Figure 2 shows the terminal architecture under the MPEG-4 IPMP Extension framework. The original MPEG-4 system without IPMP protection is shown at the upper half of the diagram (above the dotted line). The incoming MPEG-4 content stream is de-multiplexed in the DMIF (Delivery Multimedia Integration Framework). Audio, Video, OD and BIFS (Binary Format for Scenes) bitstream are supplied to the Decoding Buffers (DB) and then decoded. The decoded audio and video data are fed to the Audio Composition Buffer (CB) and the Video CB respectively, and then are composed in the Compositor together with the decoded ODs and the decoded BIFS tree or scene graph.

The lower half of the figure (below the dotted line) shows the modules provided by the IPMP Extension. The Tool list is included in the IOD of the MPEG-4 system stream to identify the IPMP tools required to consume the protected content. IPMP stream arrives as an elementary stream multiplexed in the MPEG-4 system stream. Note that the Tool list and the IPMP stream are constructed during the content authoring process (see Section 5.1.1 for an example). The Tool Manager (a conceptual entity) manages IPMP Tools within the terminal (e.g., downloading a missing tool from a remote location) while Message Router routes messages among the terminal and the IPMP Tools using a secure messaging framework (to be introduced in Section 4) to ensure that different IPMP tools from different vendors can work together. IPMP Tools can act on several control points, which are positions along the dataflow where IPMP Tool functions by taking over the protected content bitstream, processing it, and returning it back to the control point for subsequent processing of the content by the MPEG-4 terminal. The supported control points are dictated by the gray circles in the architecture diagram. For example, an encrypted MPEG-4 video stream needs to be decrypted by an IPMP tool (decryptor) at the control point right before the video decoder, and a watermark reader may need to be applied to the watermarked audio stream at the control point right after the audio decoder. If necessary, an IPMP tool can be applied to the control points right before the compositor to control the rendering process. Details about how to signal the protection scope (which objects or elementary streams) and the control points of the IPMP tools when authoring the MPEG-4 content stream are presented in Section 3.2.

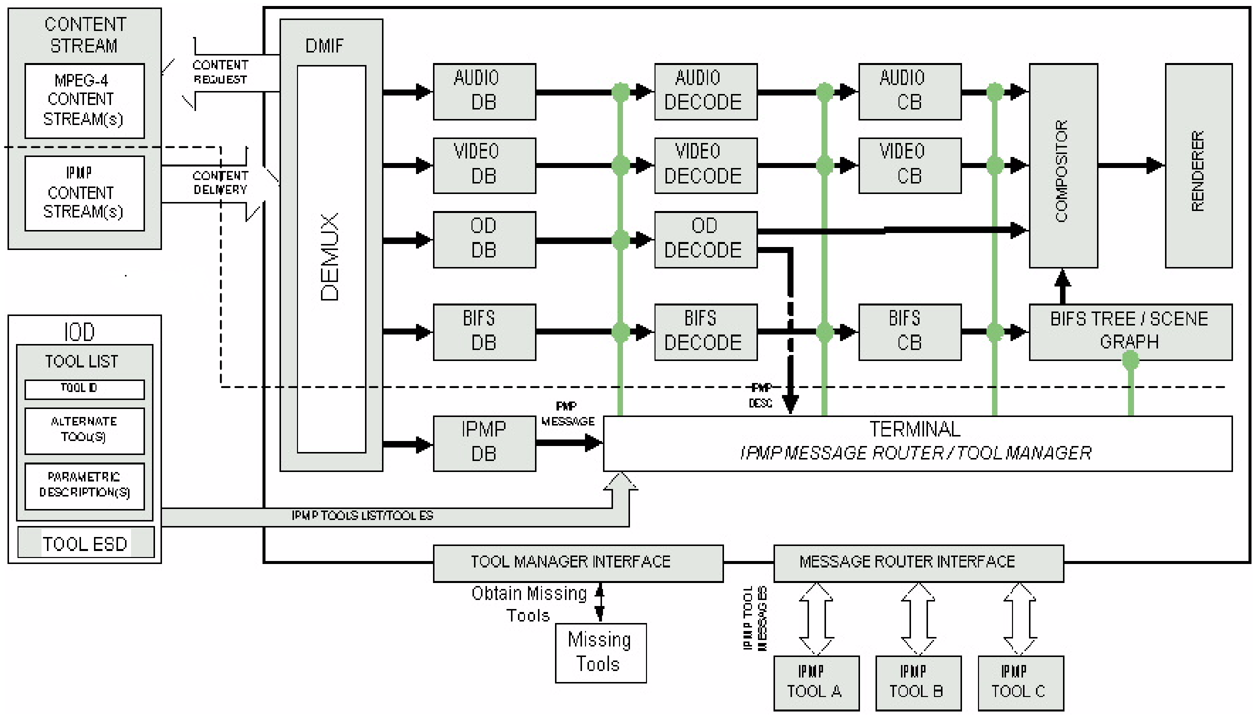

Fig. 2: The MPEG-4 IPMP terminal architecture

2.3 Advantages of the IPMP Extension Architecture

The IPMP Extension architecture achieves several important functionalities:

1) Interoperability
MPEG-4 IPMP Extension standardizes the IPMP messages and the process of message routing. By using a common set of IPMP messages, together with *industry defined* (not MPEG-4 IPMP defined) messaging API and messages extension, different IPMP Tools can be easily plugged into the terminal and interact with each other.

2) Renewability
Through the usage of Tool List and IPMP Descriptor, one can easily renew a tool for better IPMP protection by, e.g., indicating to the terminal that a new tool is needed, carrying the new tool in the Tool elementary stream in the content stream, or downloading the new tool from somewhere. Note that tool downloading is not mandatory in IPMP. IPMP provides the architecture to facilitate tool downloading.

3) Flexibility
MPEG-4 IPMP Extension does not standardize the tools. With the support of independent registration authorities, the ability to carry tools inside the content stream, and the terminal's potential capability to download required IPMP tools from a remote location, one can choose whatever algorithms or tools to perform decryption, watermarking, user authentication or integrity checking.

4) Dynamic operation
Various IPMP Tools protection can be signaled in the content with the help of IPMP Descriptor, control point, and sequence code (see definition in Section 3.2.1). Different Tools can operate at the same or different control points, acting on the same or different streams.

5) Secure tools
Terminal and Tools can choose to perform mutual authentication using the IPMP authentication messages (see discussion in Section 4.2.5) to achieve a secure communication framework.

3 Flexible Protection Signaling

Figure 3 illustrates the structure of a MPEG-4 system content protected by IPMP Extension. The information contained in the IOD and the Content Stream is shown and the relation between them is indicated. More details about each entity in Fig. 3 will be described in the following.

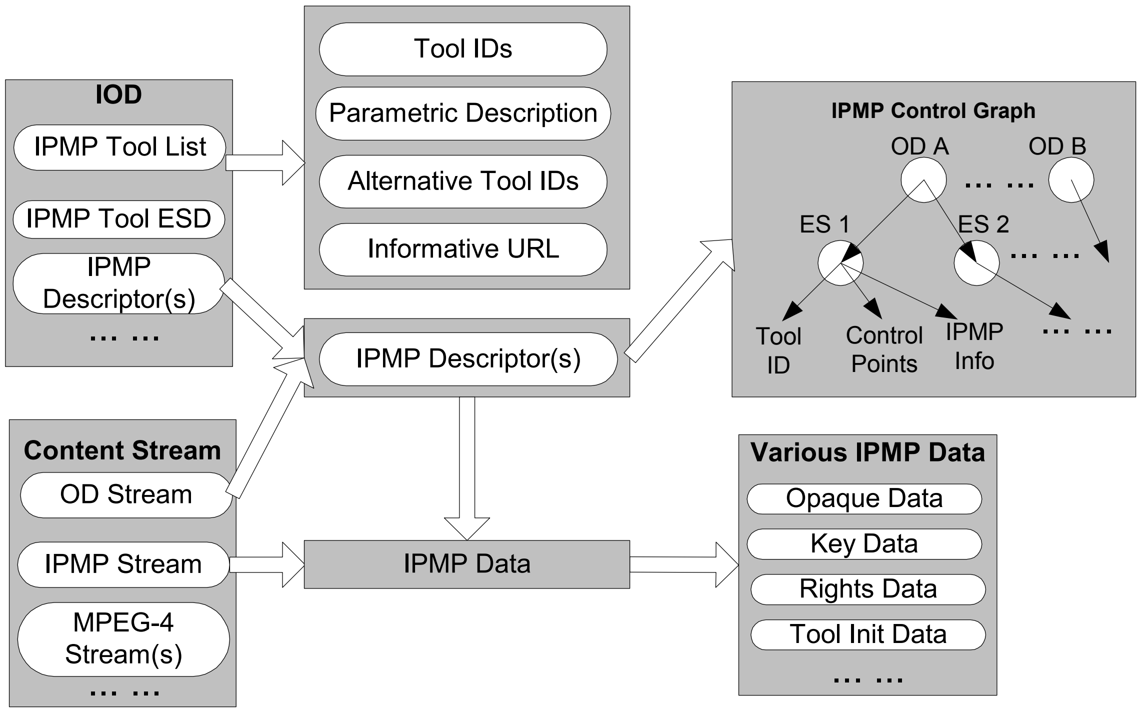

Figure 3: Structure of a MPEG-4 system content protected by IPMP Extension

3.1 Required IPMP Tools and Carriage of IPMP Tools

3.1.1 IPMP Tool List

The idea of IPMP Tool List [12] is an improvement over the IPMP "hooks". MPEG-4 IPMP Extension defines a SDL (syntactic description language [4]) descriptor IPMP_ToolListDescriptor in IOD which supports the indication of independent or alternative IPMP Tools required to consume the protected content. IOD is chosen to carry IPMP Tool List since IOD arrives ahead of OD, BIFS and other elementary streams, hence allows the IPMP Tool Manager to retrieve and make sure every IPMP Tool is present.

For each tool in the IPMP Tool List, the following information is provided:

- IPMP Tool Identifier: A given IPMP tool is identified to other entities via its IPMP Tool Identifier.
- Possible alternatives to a given Tool.
- Optional Parametric Description of the Tool (i.e., information that enables a Terminal to choose a specific Tool implementation)
- Optional informative URL

The above structure of the IPMP Tool List provides the terminal sufficient information to retrieve a tool that is required to consume the protected content. It also provides a flexible way to identify an IPMP tool via its alternatives or parametric description [1].

3.1.2 IPMP Tool ESD

The IPMP tools required to consume the protected content may have already been in the terminal, or may be downloadable from a remote location. One or more Binary Representations of IPMP Tools may also be carried directly or by reference in an MPEG presentation. MPEG-4 IPMP Extension defines a new elementary stream with streamType "IPMPToolStream" for carrying binary IPMP Tools within a MPEG-4 system stream.

One implementation of a given tool is carried as the payload of one IPMP Tool elementary stream, the representation format, packaging information and IPMP Tool ID of which is specified in DecoderConfigDescriptor in the associated ESD (elementary stream descriptor).

The IPMP Tool elementary stream is referenced through the IOD, as illustrated in Fig. 3. The IPMP Tool Manager serves as a decoder for the IPMP Tool elementary streams. IPMP Tools carried within the IPMP Tool ES can be installed, used and retained at the discretion of the Terminal implementation. They are referenced via their IPMP Tool IDs just like any other IPMP Tool.

3.2 Signaling of Various IPMP Tools Protection Scope

It is necessary to signal in the MPEG content stream which objects or elementary streams a particular IPMP tool should be used to protect, and where in the data flow of the MPEG-4 terminal the tool should be applied. The signaling of the protection scope and its control point inherits from the IPMP "hooks" [3] by using IPMP Descriptors and IPMP Descriptor Pointers. However, both IPMP Descriptor and IPMP Descriptor Pointer have been improved to allow a more flexible indication and to provide more functionality.

3.2.1 IPMP Descriptor

The IPMP_Descriptor carries IPMP information for one or more IPMP Tool instances. It may also contain optional instantiation information for one or more IPMP Tool instances. IPMP_Descriptors are conveyed and updated in either initial object descriptors, object descriptors or object descriptor streams.

Each IPMP_Descriptor has an IPMP_ToolID, which identifies the required IPMP tool for protection. The control point of the IPMP Tool's protection is signaled by another element in IPMP_Descriptor: controlPointCode, which specifies where the IPMP Tool resides (see control points illustrated in Fig. 2).

Sequence Code is another element in IPMP_Descriptor that is used to signal the sequencing priority of the IPMP Tool instances at the given control point. In the case that multiple tools are governing the same control point on a given stream, the tool with the highest sequence code shall process data first for that control point for that stream.

3.2.2 Using IPMP Descriptor to Signal Protection at Different Control Points.

The IPMP_DescriptorPointer appears in the ipmpDescPtr section of an OD or ESD structures. Different presence locations signal different protection scopes. The presence of this descriptor pointer in an object descriptor indicates that *all* streams referred to by embedded ES_Descriptors are subject to protection and management by the IPMP Tool specified in the referenced IPMP_Descriptor. The presence of this descriptor pointer in an ES_Descriptor indicates that *only* the stream associated with this descriptor is subject to protection and management by the IPMP Tool specified in the referenced IPMP_Descriptor.

IPMP_DescriptorPointer also has an IPMP_ES_ID that is the ID of an IPMP stream that may carry messages intended to the tool specified in the referenced IPMP_Descriptor. In case more than one IPMP stream is needed to feed the IPMP tool, several IPMP_DescriptorPointer can be given with the same IPMP_DescriptorID and different IPMP_ES_ID.

By utilizing IPMP_Descriptor and IPMP_DescriptorPointers, the terminal can build an abstract IPMP Control Graph (see Fig. 3), which bears a tree-like hierarchy. One example is shown in Fig. 4 where an elementary stream VIDEO-EL Stream is associated with the elementary stream descriptor ESD=C under Object Descriptor A. OD A contains an IPMP descriptor pointer that points to an IPMP descriptor (IPMP DSCR=X) which carries Tool ID of the IPMP tool required to consume the VIDEO_EL Stream, and information about where the IPMP tool should be applied (i.e., Control Points) and other IPMP information. Different IPMP Tools can be specified to protect different objects, or different elementary streams under that object, at different control points, or at the same control point but bearing different sequence codes.

3.3 Delivery of IPMP Data to the Terminal and/or IPMP Tools

IPMP Data is the information directed to a given IPMP Tool or terminal to enable, assist or facilitate its operation. It is sometimes referred to as IPMP Information. IPMP Data includes but not limited to key, usage rights, tool initialization, mutual authentication information.

3.3.1 Places to carry IPMP Data

IPMP Data can come from various sources. When it is carried in the content, it can be contained in IPMP_Message class in an IPMP Stream or IPMP_Descriptor [1]. IPMP_Message is the data class defined to carry IPMP Data in the IPMP Stream, which includes the identification of the recipient of this IPMP_Message as well as a place holder for IPMP Data to be carried inside.

IPMP Data can also be generated by an IPMP Tool or IPMP terminal and delivered to other IPMP Tools or the IPMP terminal as a payload of IPMP_MessageFromTool (see definition in Section 4.2.1).

3.3.2 Delivery of IPMP Data to IPMP Tools

IPMP Information is routed using *normative* addressing methods, as discussed in Section 4.2. The addressee of a specific message is implicit either by bitstream context or by process context. In the MPEG-4 bitstream context, the addressee is the IPMP Tool whose identity is indicated in the IPMP message or IPMP descriptor header. Information is delivered at a specific time, specified in the bitstream or implicit by process.

IPMP Data carried in IPMP_Descriptor is delivered to the IPMP Tool declared in the descriptor. The IPMP Data is sent as a payload of the message IPMP_DescriptorFromBitstream (see definition in Section 4.2.1). IPMP Data carried in IPMP_Message class of IPMP Stream is delivered to the IPMP Tool declared in the IPMP_Descriptor whose IPMP_DescriptorID is indicated in the same IPMP_Message class. The IPMP Data is sent as a payload of the message IPMP_MessageFromBitstream (see definition in Section 4.2.1).

Physical routing of information and context resolution are handled by the Message Router. The Message Router abstracts all platform-dependent routing and delivery issues from the IPMP Tools.

4. Secure Messaging Framework

MPEG-4 IPMP Extension defines the following components of the IPMP Tool Interaction Framework: Interaction (or communication) between the Terminal and the IPMP Tool(s), realized via "messaging" between the Terminal and the IPMP Tools; the messages (syntax and semantics), and the process of message routing. As discussed in Section 2, this messaging framework allows different IPMP tools, potentially from different vendors, to be easily plugged into the terminal and interoperate with each other and with the terminal in a secure way. This is a critical step toward supporting interoperability in MPEG-4 IPMP.

All IPMP Tool interactions take place via the Terminal. IPMP Tools do not communicate directly with each other within the scope of the standard.

4.1 Flexible Messaging Infrastructure

All IPMP Tool Messages are routed through the Terminal. To represent this function, an entity called the Message Router (MR) is defined in the architecture. The MR connects and communicates with supported IPMP Tool(s). It thus abstracts the physical interface of one IPMP Tool from any other IPMP Tool that wishes to communicate with it. The interface between the Message Router and the Tools is *non-normative* and is not defined in the specification. Only messages derived from an expandable base message class called IPMP_ToolMessageBase [1] may cross the interface.

Message Routing is assumed to be instantaneous. In case of an MR error, an appropriate error status is returned by the MR. In all other cases, the MR is required to route, without a change in semantic meaning, information and responses as received.

4.2 Messages defined within MPEG-4 IPMP Extension

IPMP_ToolMessageBase is the expandable base class for all messages that may across the messaging interface within MPEG-4 IPMP Extension. It specifies the context ID (identifier of the logical instance of a tool, assigned by terminal) of the originator of the message, and the context ID of the intended recipient of the message.

4.2.1 IPMP Data Delivery Messages

There are currently three defined IPMP data delivery messages [1], i.e., IPMP_MessageFromBitstream, IPMP_DescriptorFromBitstream, and IPMP_MessageFromTool. Message IPMP_MessageFromBitstream is used to deliver IPMP_Messages received in the Content to the IPMP Tool context specified in the IPMP_Message. If an IPMP *Access Unit* delivered in the IPMP Elementary Stream contains more than one IPMP_Message for a specific IPMP Tool, all IPMP_Message for that tool will be included in a single IPMP_MessageFromBitstream message. Note that Access Unit is one individually accessible portion of data within an elementary stream. An access unit is the smallest data entity to which timing information can be attributed. Message IPMP_DescriptorFromBitstream is used to deliver an IPMP_Descriptor received in the bitstream to the IPMP Tool specified in the IPMP_Descriptor.

Message IPMP_MessageFromTool is used to deliver any IPMP Data from tool to tool. These IPMP Data can be categorized into Instantiation and Notification Messages, Event Notification Messages, IPMP

Processing messages, Authentication Messages, User Interaction Messages, Consumption Messages, and Inter Device Messages.

4.2.2 Instantiation and Notification Messages

These messages are used to instantiate and destroy logical instances of new Tools, to inform newly instantiated Tools of existing Tools, and to notify existing Tools of a new instantiation. Although they are primarily designed to be used by tools to request logical instances of other tools, these messages may also be used in the content stream when upstream capabilities exist, for example for mutual authentication between the server and the terminal.

4.2.3 Event Notification Messages

These messages provide the IPMP Tools the ability to request and get notified of events including connection, disconnection and watermark detection.

4.2.4 IPMP Processing

These messages are defined to be used in the IPMP process. Although the exact functioning of the various IPMP tools are not specified, these messages support the *interoperable* use of common types of IPMP tools such as encryption/decryption, audio and video watermarking as well as rights management and governance. For example, the IPMP_SelectiveDecryptionInit message defined in Annex A of the MPEG-4 IPMP Extension [1][21][22] allows a terminal to configure a selective decryption tool (e.g., the ones proposed in [13][23]). It tells how the bitstream is encrypted, whether all bits are encrypted, or only portions of it, what portions of the received bitstream are encrypted [13] or shuffled [23], and therefore need to be decrypted or de-shuffled, etc. The IPMP_KeyData message allows carriage of a key, including timing information in order to synchronize the key with the media stream. These messages may be directly carried in the bitstream in the IPMP_Message and/or the IPMP_Descriptor messages or may be wrapped in the IPMP_MessageFromBitstream class or IPMP_DescriptorFromBitstream messages for passing between tools or between tools and the Terminal.

4.2.5 Authentication Messages

At any point in IPMP Information or Content processing, IPMP Tools may be required to communicate with one another or with the Terminal. The degree of security required for such communication is determined by a number of variables including information that may be included by the content provider in the Content and conditions of trust established between tool providers *a priori* and out of band. It is generally the case that a given ES is protected by multiple tools but that certain types of tools are complex (e.g. Rights Management tools) and others are utilities (e.g. Decryption engines). Complex tools may control the instantiation of other tools or make decisions about content use in response to usage queries from the terminal. Mutual authentication may occur between any pair of tools but the level of security required for this communication will in part be dictated by data contained in the bitstream in an opaque manner. The mechanism for making the determination of this security level is *non- normative*.

Mutual authentication is executed as follows:

1. The Tool that initiates mutual authentication with another tool determines the conditions of trust to be achieved by such authentication, i.e. the initiating tool determines whether it needs only integrity protected communication or full secure, authenticated communication. This level may or may not be dictated by IPMP Information in the Content.
2. The communicating tools then engage in a message exchange to determine which authentication protocol will be used. In some cases, this protocol may have been determined by an *a priori* out of band negotiation between the tool providers in their security audits of one another. The authentication messages are used to request a mutual authentication, or are generated by and exchanged between IPMP Tools and IPMP Tools and a Terminal for the purpose of mutual authentication.

4.2.6 User Interaction Messages

These messages allow information to be exchanged between the User and an entity requiring information from the User.

4.2.7 Consumption Permission

The IPMP_CanProcess message enables the notification of the Terminal, by IPMP tools, as to the tools ability to begin or discontinue, processing content.

4.2.8 Inter Device Messages

MPEG-4 IPMP Extension has also defined a set of inter-device messages in Annex D of [1]. These messages support the transfer of the content and IPMP Tools. Transfer of the content and tools can be made secure by putting them into secure message payload, using any established mechanisms. Section 5.2 makes use of these messages to provide a solution to the *Gobi Dessert Scenario*.

5. Two Sample Usage Scenarios

We illustrate two sample usage scenarios in this section where the second one is the usage scenario for the *Gobi Desert Scenario* we discussed in Section 1.

The first sample usage scenario illustrates a use case whereby an MPEG-4 system stream consists of one video object and one audio object. The video object is further composed of two elementary streams, one is video stream base layer (BL), while the other is video stream enhancement layer (EL). It is protected by MPEG-4 IPMP Extension.

5.1 A Simple MPEG-4 IPMP Extension Protected MPEG-4 Content

5.1.1 Content Authoring

At the content creation side, the content author creates a simple MPEG-4 system stream, which mainly consists of one single audio object with one audio elementary stream under it and one single video object with two video elementary streams (BL and EL) under it.

In order to protect the content, the content author uses AES [15] encryption tool to encrypt the video enhancement layer since it is of a higher value. The video base layer remains unprotected since it is not of a high commercial value. The author also embeds (using watermark encoding) some copyright information bits into the audio stream.

Suppose that the content author is aware that there are an IPMP tool X with tool_id 0xAAA that is capable to do AES decryption and an IPMP tool Y with tool_id 0xBBB that is able to detect the watermark from the audio elementary stream. The content author hence constructs the IPMP Tool List including the above-mentioned two tool_id's, to indicate to any terminal receiving the MPEG-4 content that these two tools are needed to play the content. The Tool List Descriptor is put under IOD. If necessary, the author can also put IPMP tool Y, binaries compiled for the desired platforms, as a tool elementary stream referenced in IOD, in case the terminal does not have tool Y.

The content author constructs the abstract IPMP Control Graph (described in Section 3.2.2) using IPMP_Descriptor and IPMP_DescriptorPointer to indicate to the terminal that tool X needs to be used for video EL stream, that tool X needs to sit at the control point of "before_decoder". The Control Graph also indicates that tool Y needs to be used for audio elementary stream, that tool Y needs to sit at the control point of "after_decoder". The IPMP Control Graph can be built by embedding IPMP Descriptor Pointers into their respective Elementary Stream Descriptor or Object Descriptor. The control point information, sequencing code, as well as any opaque data which may contain the tool initialization information is carried in each tool's specific IPMP Descriptor which is sent through object descriptor stream.
.
The AES encryption uses a time variant key stream to encrypt the above-mentioned video EL stream. Hence the content author constructs the IPMP stream that is a concatenation of IPMP_Message class, with each IPMP_Message specifying the destination (i.e., IPMP Tool X), and each IPMP_Message body containing IPMP_KeyData which carries the time variant key. The constructed IPMP stream is also multiplexed with other elementary stream under the video object (see OD A in Fig. 4). The content structure is shown below in Fig. 4. IOD, BIFS is omitted for brevity.

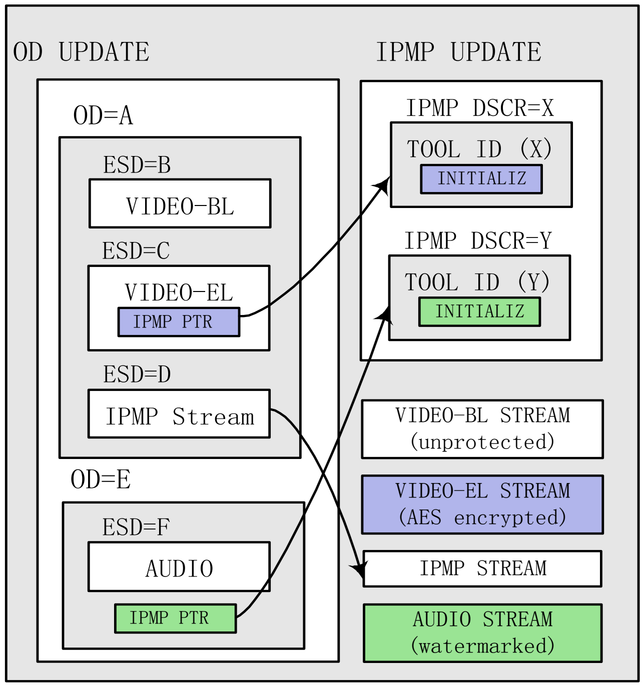

Figure 4: A Sample Content Structure

5.1.2 MPEG-4 IPMP Extension Terminal Behavior

The simplified architecture of MPEG-4 IPMP Extension terminal consisting of the two tools to handle the above authored content is demonstrated below in Figure 5.

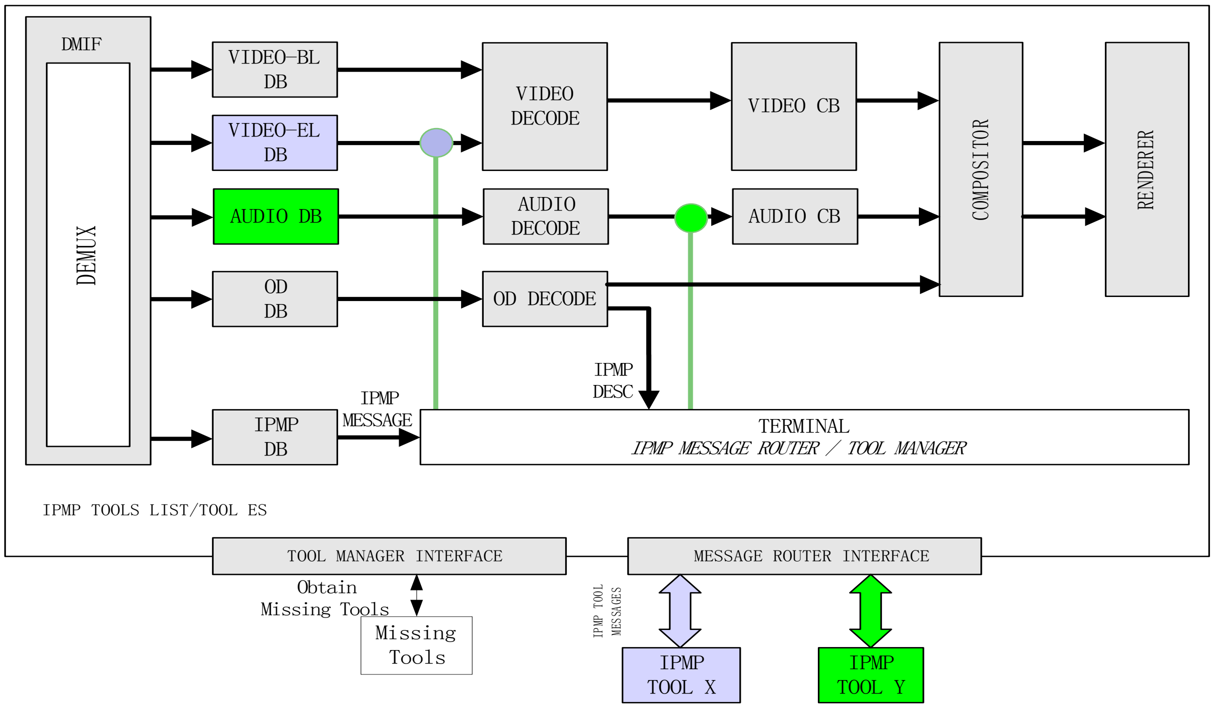

Figure 5: A Sample Terminal Architecture

The dataflow and how IPMP tools protect the content is based on Sections 3 and 4. The terminal, upon receiving the above constructed IPMP protected MPEG-4 stream, retrieves IPMP Tool List from IOD. According to the two tool_id's mentioned in the IPMP Tool List, the Tool Manager checks the presence of the two tools inside the terminal. If not present, the Tool Manger may retrieve them from a remote location which is also indicated in the Tool List, or it may attempt to get the missing tool from neighboring devices, or may retrieve the tool from the content (if the tool is carried in the content as a tool ES).

The terminal then checks the IPMP Control Graph by retrieving IPMP Descriptor Pointers from the Object Descriptor and/or Elementary Stream Descriptor. The IPMP Descriptors pointed by the two pointers are updated through OD stream. It now has the information that where and how tool X and tool Y should be used.

Tool X is instantiated at the before_decoder control point (between Video-EL DB and video decoder). Tool Y is instantiated at the control point that is after audio decoder. Both tools need to do a mutual authentication with the terminal using the mutual authentication messages to ensure both tools are trusted by the terminal. The mutual authentication could result in a secure communication channel between IPMP tools and the terminal.

The IPMP Descriptor containing the control point, sequence code and other IPMP data is sent to the tool indicated in the IPMP Descriptor through the "IPMP_DescriptorFromBitstream" message. The IPMP data embedded in the IPMP Descriptor may include the initialization information for that particular tool, for example, IPMP_AudioWatermarkingInit [1]. The IPMP tool receives this information and configures itself.

At the control point of video-EL decryption, the terminal routes demultiplexed video-EL bitstream to the IPMP Tool X running at that control point.

.
The IPMP Stream is received by the terminal. According to the destination address (IPMP Descriptor ID) contained within each IPMP_Message(), the message is routed to the specific tool at the time indicated by the timing information associated with the access unit which carries the IPMP_Message().

The delivery is done using IPMP_MessageFromBitstream message. For the IPMP Tool X (AES decryption tool), the message contains the time variant key in the form of IPMP_KeyData, which is used by Tool X to do its decryption job.

After receiving and decrypting the video-EL access units, the IPMP Tool X returns the decrypted video access units to the terminal through the *non-normative* messaging interface.

At the control point of the audio watermark retrieval, the terminal routes every decoded audio packet to the IPMP Tool Y. Tool Y retrieves the watermark from the received audio packets, and the watermark retrieval result is notified to the terminal in the form of IPMP_SendAudioWatermark message [1]. Tool Y may also verify the copyright information bits in the audio stream, and if necessary, Tool Y can control the rendering process by sending the IPMP_CanProcess message to the terminal.

5.2 A Note on the *Gobi Desert Scenario*

In the *Gobi Desert Scenario*, it is assumed that two different devices (owned by Alice and Bob) want to share content and that they can communicate with one another via IR, Firewire, etc. Alice's device supports IPMP-A tool, Bob's supports IPMP-B tool. The following steps show how this is accomplished within the MPEG-4 IPMP Extension framework.

1. Bob wants to listen to the content that is packaged for IPMP-A
2. He connects his device to Alice's
3. He locates the content that he wants and requests a download through the MPEG-4 IPMPX defined inter device messages.
4. Alice and Bob's devices do a mutual authentication using IPMP Extension's inter-device messages, and establish a secure authentication channel (SAC).
5. Alice's device transfers the content to Bob's device using the secure messages over the SAC between the two devices.
6. By checking the IPMP tools list in the requested content, Bob's device determines that IPMP-A tool is required and that IPMP-A tool is not available in the terminal nor is it conveyed in the IPMP tool ES in the content stream.
7. Bob's device connects to Alice's device to request the missing IPMP-A tool.
8. Again, mutual authentication is done between Alice and Bob's devices, a SAC is established.
9. The IPMP-A tool is securely transferred to Bob's device using IPMPX's inter-device tool transfer messages.
10. Bob can now play the content locally by using the IPMP-A tool.

11. Note that the trust relationship between the two devices should have been established between the device manufacturers. If the two devices do not trust each other, this copy procedure cannot occur.

6. Conclusion

This paper introduces MPEG-4 IPMP Extension, the break-through technology standardized by MPEG for interoperable Digital Rights Management. MPEG-4 IPMP Extension offers flexibility, robustness and interoperability, which promotes secure content delivery around the globe. MPEG-4 IPMP Extension can be used in combination with proprietary tools, which enables the implementation of various degree of security for different business models, while maintaining the interoperability. Some implementation issues, such as messaging interfaces, registration authorities, and profiling for different industrial domains, are considered out of the scope of MPEG, and are left unspecified. They are left for further specification by the industrial body for a specific application.

MPEG-4 IPMP Extension has been finalized, and the industry is beginning to accept it. MOSES [16], a consortium of more than 7 worldwide companies has just launched a *music-4-you* service based on MPEG-4 IPMP Extension for secure music distribution. Internet Streaming Media Alliance (ISMA) [17] has adopted MPEG-4 IPMP Extension's protection signaling method in its ISMACryp specification. The MPEG-4 IPMP Extension framework has also been successfully mapped to MPEG-2 system, resulting in MPEG-2 IPMP [6][7], which has drawn substantial interest from the broadcasting industry as well as broadband applications.

7. References

[1] ISO/IEC JTC1/SC29/WG11 MPEG, N5284, "Information technology - coding of audio-visual objects — Part 13: Intellectual Property Management and Protection (IPMP) extension", Shanghai, Oct 2002

[2] ISO/IEC JTC1/SC29/WG11 MPEG, Text of ISO/IEC 14496-1/PDAM3, N5282, "Amendment to MPEG-4 system on IPMP Extension", Shanghai, Oct 2002.

[3] ISO/IEC JTC1/SC29/WG11 MPEG, "MPEG-4 Intellectual Property Management & Protection (IPMP) Overview & Applications", N2614, Rome, Dec. 1998.

[4] ISO/IEC JTC1/SC29/WG11 MPEG, International Standard ISO/IEC 14496-1. "Information Technology – Generic Coding of Moving Pictures and Associated Audio, Part 1: Systems," 2001

[5] ISO/IEC JTC1/SC29/WG11 MPEG, N2195, "MPEG-4 applications", Tokyo, March 1998.

[6] ISO/IEC JTC1/SC29/WG11 MPEG, "Study Text of ISO/IEC 13818-11/FCD (MPEG-2 IPMP)," N5469, Awaji, Dec. 2002.

[7] ISO/IEC JTC1/SC29/WG11 MPEG, "Study Text of ISO/IEC 13818-1:2000/FPDAM2 (MPEG-2 IPMP Amendment)," N5465, Awaji, Dec. 2002.

[8] Ahmet M. Eskicioglu, John Town, and Edward J. Delp, "Security of Digital Entertainment Content from Creation to Consumption", *Proceedings of SPIE Applications of Digital Image Processing XXIV*, San Diego, CA, July 31-August 3, 2001.

[9] B.J. van Rijnsoever, Peter Lenoir, J.P.M.G. Linnartz, "Interoperable protection for digital multimedia content", invited paper in *Journal of VLSI Signal Processing - Systems for Signal, Image and Video Technology" special issue on Multimedia Communications*, 2001

[10] L. Chiariglione, "Intellectual Property in the Multimedia Framework", *Management of Digital Rights*, Berlin, 00/10/20-21, http://leonardo.telecomitalialab.com/paper/berlin_drm/drmoo.htm.

[11] ISO/IEC JTC1/SC29/WG11 MPEG, "Call of Proposals for IPMP solutions", N3543, Beijing, July 2000.

[12] Ming Ji, SM Shen, ISO/IEC JTC1/SC29/WG11 MPEG, "Refined Tool List Syntax and Tool Download Ability", M7530, Sydney, July 2001.

[13] J. Wen, M. Severa, W. Zeng, M. Luttrell and W. Jin, "A format compliant configurable encryption framework for access control of video", *IEEE Tran. Circuits & Systems for Video Technology, Special Issue on Wireless Video,* pp. 545-557, June 2002.

[14] Margarita A. Kousseva, "Digital Rights Management Literature Review", http://www.nova.edu/~kousseva/literature_review.html, Dec, 2002.

[15] Advanced Encryption Standard, NIST US FIPS PUB 197, http://csrc.nist.gov/encryption/aes/

[16] MPEG Open Security for Embedded Systems (MOSES), http://www.crl.co.uk/projects/moses/

[17] Internet Streaming Media Alliance (ISMA), http://www.isma.tv

[18] *Data Encryption Standard (DES),* FIPS PUB 46-3, Reaffirmed Oct. 25, 1999. Available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[19] DVD Forum, website: http://www.dvdforum.com/forum.shtml

[20] Secure Digital Music Initiative (SDMI), website: http://www.sdmi.org/

[21] W. Zeng, J. Wen, and M. Severa, "Editorial changes and extension of Annex C on selective decryption configuration message," *ISO/IEC JTC 1/SC 29/WG 11/IPMP* M7989, Jeju, March 2002.

[22] J. Wen, W. Zeng, D. Kosiba and M. Severa, "A format-compliant configurable encryption framework for access control of multimedia," *ISO/IEC JTC 1/SC 29/WG 11/IPMP* M7213, Paris, June 2001.

[23] W. Zeng, J. Wen and M. Severa, "Fast self-synchronous content scrambling by spatially shuffling codewords of compressed bitstreams," *Proc. IEEE Inter. Conf. Image Proc.*, Sept. 2002.