Evaluation of MPEG-4 IPMP Extension

HweeHwa Pang, Yongdong Wu
Institute for Infocomm Research(**I**^{2}**R**)
21, Heng Mui Keng Terrace, Singapore
{hhpang, wydong}@i2r.a-star.edu.sg

***Abstract*— MPEG-4 IPMPX (Intellectual Property Management and Protection eXtension) is the latest ISO standard which provides a flexible framework for protecting MPEG streams. The message mechanism of IPMPX enables inter-operability among IPMPX-compliant devices no matter which protection methods are embedded. This paper highlights several problems in the message syntax of IPMPX: the tool delivery message `IPMP_ToolES_AU` is vulnerable to network attack, the authentication message `IPMP_Mutual_Authentication` is incapable of defending against forgery attack, and the configuration message `IPMP_SelectiveDecrptionInit` is ambiguous and redundant. We propose a number of remedies to those problems, which can be incorporated into a corrigenda to improve the present ISO MPEG-4 IPMP standard.**

I. Introduction

Buying electronic contents typically entails browsing the seller's Web site (e.g. `http://www.amazon.com`), searching a database, and paying with a credit card before the content is rendered. Since digital content can be copied and disseminated easily and without any degradation in quality, the publishing industry stands to lose \$1.5 billion through piracy by 2005 [1]. Naturally, publishers, distributors, and Web retailers are looking for safe and effective ways to manage their intellectual property.

In response, many companies have developed rights management systems or solutions, such as Mediaplayer^{TM} and RealNetwork^{TM}. This multiplicity of solutions has proved more a curse than a blessing, as most of the systems are incompatible with each other [2]. That is to say, a piece of protected content can be consumed on some designated devices/systems only. To achieve economy of scale, this has to change; "legitimate" content distribution methods need to become interoperable. Thus, it is desirable to standardize the interface for the protection methods, particularly those for multimedia content that has high market value.

MPEG-4 [3] [4] is an excellent multimedia standard for digital television, interactive graphics applications, and interactive multimedia. To safeguard Intellectual Property in the form of MPEG-4, IPMP [5] defines the hooks for the protection methods after MPEG-4 became an International Standard in 1999. To enhance inter-operability, the newer MPEG-4 IPMPX (MPEG-4 part 13) [6] [7] was finalized recently. It not only enables IPMP-compliant devices to render content within an MPEG-4 terminal, but also provides a framework with normative messages to select and configure the most effective and appropriate tools. Following the principle of IPMP, several works (e.g. [8] - [11]) proposed flexible ways to protect MPEG-4 content in a controlled manner. However, although the IPMPX message mechanism [7] is sound, some message syntaxes are problematic. Specifically, the tool delivery message `IPMP_ToolES_AU` is vulnerable to network attack, the authentication message `IPMP_Mutual_Authentication` can be used for forging tools, and the configuration message `IPMP_SelectiveDecrptionInit` is ambiguous. We propose to patch those weaknesses in an upgrade of the standard.

The reminder of this paper is organized as follows. Section II gives an overview of the MPEG-4 IPMP extension; readers who are familiar with IPMPX can skip the section. In Section III, we focus on some clauses such as authentication protocol and Decryption Initialization, and elaborate on their weaknesses. For each weakness, we propose some remedies. Finally, Section IV summarizes our conclusion.

II. MPEG-4 IPMP Extension

To make the paper self-contained, we excerpt the following terminology from [7].

Terminal: An environment that consumes possibly protected content in compliance with the usage rules.

Tool: A module that perform (one or more) IPMP functions such as authentication, decryption, watermarking, etc. Conceptually, the use of one or more Tools is combined to perform the functionality of an IPMP system.

Tool Manager: A conceptual entity within a terminal that processes tools and retrieves the tools that are specified therein.

Tool Message: A message passed between any combination of tool or terminal.

ES: Elementary Stream is conceived as a flow of data that originates from a single source in the sender and terminates in a single sink at the receiver. An IPMP ES is used for delivering data related to IPMP tools.

A. IPMPX

The architecture of IPMPX is shown in Fig.1, where the terminal provides the user interface, manages the tools, and communicates with the content provider. Each tool is associated with a unique ID assigned by a registration authority (`www.ipmp-ra.org`), or a proprietary IPMP system.

Fig.2 refines the MPEG-4 system architecture. In a typical application, a terminal requests digital content and downloads MPEG-4 content which includes an IOD (initial object descriptor). The IOD includes `IPMP_Descriptor` and `IPMP_ToolListDescriptor`. Each `IPMP_Descriptor` is identified with an `IPMP_DescriptorlID`. Possibly, it

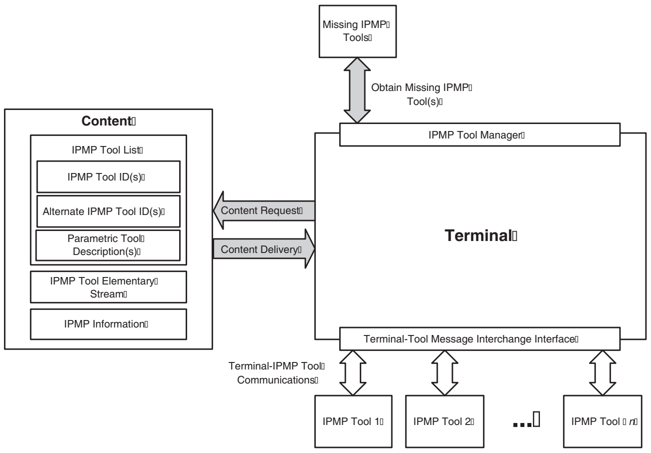
Missing IPMP Tools
Obtain Missing IPMP Tool(s)
IPMP Tool Manager
Content
IPMP Tool List
IPMP Tool ID(s)
Alternate IPMP Tool ID(s)
Parametric Tool Description(s)
IPMP Tool Elementary Stream
IPMP Information
Content Request
Content Delivery
Terminal
Terminal-Tool Message Interchange Interface
Terminal-IPMP Tool Communications
IPMP Tool 1
IPMP Tool 2
...
IPMP Tool n

Fig. 1. Architecture Diagram for Walkthrough Concept [7].

also includes `IPMP_ToolID`, the control point at which the tool resides (e.g., decryption is performed before decoding or after decoding), processing priority `sequenceCode`, and other tool-specific data. `IPMP_ToolListDescriptor` indicates the tools to be used for processing the protected content. The terminal examines the tool list carried in the content stream, and instantiates all the tools needed. If some tools are not available locally, the tool manager will attempt to download them according to `ToolURL` within `IPMP_ToolListDescriptor.IPMP_Tool`. Only after all the required tools have been initialized, can the terminal start to process the content. Simultaneously, the terminal may also serve as a bridge for messages between the tools.

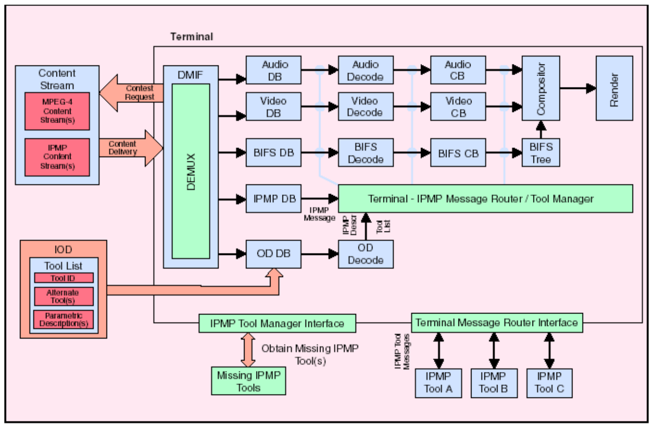

Fig. 2. Mapping of IPMP Extensions to the MPEG-4 system architecture [7]. The content stream contains protected content as well as tool data necessary for consuming the content. The terminal bridges the link between the tools and manages them, including downloading and instantiating the tools.

III. Evaluation of IPMPX

IPMPX defines security functionalities for tool delivery, initialization and message exchange. This section points out security weaknesses in the syntax definition of the trust model and some IPMPX messages. In this section, the syntax is described in SDL (Syntactic Description Language [3]). For simplicity, we ignore the syntax tag and the sender/recipient.

A. Trust Model

In an IPMP system, the MPEG-4 terminal, as well as the pre-installed tools are trusted. Additionally, IPMPX defines the attacker profile in the data structure `IPMP_TrustSecurityMetadata` based on the attacker's resource as class I (little resource), II (some resource), and III (unlimited resource) [2]. However, such a definition based on resource capability is not well recognized in the security community. Rather, the commonly accepted classification is based on whether a system is secure against ciphertext-only attack, known plaintext attack, etc [12].

In the key delivery semantics `IPMP_KeyData`, the `keyBody` is transmitted along with other data such as expiry date. We think the secret `keyBody` should be transmitted in a protected form instead.

B. IPMP_ToolES_AU

IPMPX has a major advantage over IPMP, in providing for dynamic insertion and replacement of tools. A new tool may be acquired from a specific site within an IPMP ES Access Unit. [1] For transmitting a missing tool, `IPMP_ToolES_AU` is used to actually carry the required tool, similar to the downloading of JAVA applet in a web application. The syntax of `IPMP_ToolES_AU` is shown in Fig.3.

```
class IPMP_ToolES_AU
{
    bit(1) isSigned;
    **if (isSigned)** {
        ByteArray IPMP_Tool_Signature;
        ··· certificates ···
        bit(128) Verifying_Tool_Id;
    }
    bit(32) sizeOfTool;
    bit(8) Tool_Body[sizeOfTool];
}
```

Fig. 3. `IPMP_ToolES_AU` syntax [7]. `IPMP_Tool_Signature` is the digital signature generated by a trusted authority, `Verifying_Tool_Id` indicates the tool for verifying the requesting tool, and `Tool_Body` is the executable such as JAVA bytecode. "bit(n) X" means X is of n bits.

Although `IPMP_ToolES_AU` delivers the tool correctly, the syntax has the following weaknesses:

1) Efficiency: To provide an authenticated tool, a transmitter sends the tool and overhead including `IPMP_Tool_Signature` and certificates, in addition to signatures on the individual AUs. If a tool is delivered via multiple AUs, the overhead is incurred repeatedly, thus wasting network resource. Instead, IPMPX should define

[1] In an IPMP stream, discrete portions of data are associated with specific points in time. Generally, these potions of data are called AU (Access Unit), and each ES is modelled as a sequence of AUs [4].

explicitly that each IPMP_ToolES_AU encapsulates only one tool.

2) Security: In order to be compatible to terminals which do not care about security, IPMP_ToolES_AU enables a terminal to install unauthorized tools with isSigned=False. However, when a terminal installs an unauthorized tool, it opens a door to system crackers, and further threatens network security. For example, if the IPMP-enabled devices from a manufacturer accept the tools with isSigned=False, they may be used to launch DDOS (Distributed Denial-of-Service) like those that crashed the servers of Yahoo!, Buy.com and eBay.com in 2000 [13]. As noted in [14]: (To start DDOS,) " The requests were sent to Yahoo! indirectly. The attacker or attackers actually sent a flood of requests through networks that fraudulently listed Yahoo! as return address". It is not acceptable for ISO security standard compliant devices to bring in such serious security threats to the whole networks.

Indeed, although signed applets can access system resources (e.g. read the disk), they cannot remain on the host. However, IPMPX not only enables to install/execute unauthorized tools, it even allows the unauthorized tools to reside on the terminal stealthily. We propose to disable installation of unauthorized tools into terminals. At a minimum, a warning alert MUST be given when an unauthorized tool is to be installed.

C. IPMP_Mutual_Authentication

Besides the extension to tool refreshment with IPMP_ToolES_AU, IPMPX provides an authenticated channel for tool communication. To start an authentication process between tools and/or the MPEG-4 terminal, one tool (initiator) sends a message IPMP_initAuthentication to the intended recipient. Following that, the two participants execute an authentication protocol with the message format as shown in Fig.4.

```
class IPMP_Mutual_Authentication
{
      ......
   bit (1) inclAuthCodes;
      AlgorithmDescriptor agreedAlgorithms[];
   if (inclAuthCodes){
      unsigned int type;
      if (type == 1) {
      // ··· certificates ···
      } elseif (type == 2) {
         KeyDescriptor publicKey;
      }
      ......
      ByteArray authCodes; // signature
   }
}
```

Fig. 4. IPMP_Mutual_Authentication syntax [7]. The authentication algorithm agreedAlgorithm is negotiated by the participants, and authCode is the signature of this message excluding the signature itself.

The authentication protocol comprises three steps: algorithm agreement, signature generation and verification. Unfortunately, all the steps in the present protocol require refinement:

1) Algorithm agreement: With this message, both tools agree on a group of authentication algorithms agreedAlgorithms[]. However, there is only one signature authCodes, and which algorithm should be applied is undefined. Furthermore, in practice one algorithm is all that is needed.

2) Signature generation: A digital signature is used to prove that one participant (i.e., a tool) owns a private key corresponding an authentic public key. In other words, the prover must know a private key in order to generate the digital signature authCodes. Usually, a prover tool has two ways to obtain the private key:

- The private key is given to the prover by the host terminal. In this case, the prover tool is merely a signature generation algorithm and hence can not prove its identity.
- The private key is hardcoded in a genuine tool with code obfuscation technology [15]. If so, an attacker may retrieve the private key through reverse-engineering so as to construct a malicious tool, or the attacker may wrap a malicious tool around the genuine tool. Consequentially, a so-called "authentic" tool may in fact be malicious and the authentication protocol fails.

3) Verification: In case of type=2, a publicKey will be used for verifying the authCodes (i.e., digital signature). However, this authentication method is incorrect if publicKey is not authenticated because an attacker can forge tools easily. With regard to the attack diagram in Fig.5, a malicious tool generates a pair of forged private/public key, and sends to the verifier tool a bogus message IPMP_Mutual_Authentication with the following parameters: type=2, forged publicKey e, and forged authenCodes. Clearly, the verifier will be deceived.

In general, a verifier must have some trusted public keys to authenticate an identity with a digital signature scheme. For example, Microsoft's Internet Explorer embeds the public key of a Certificate Authority (e.g. *Verisign.com* which issues certificates for web sites to perform secure transactions). An unauthorized public key is useless for authentication purposes.

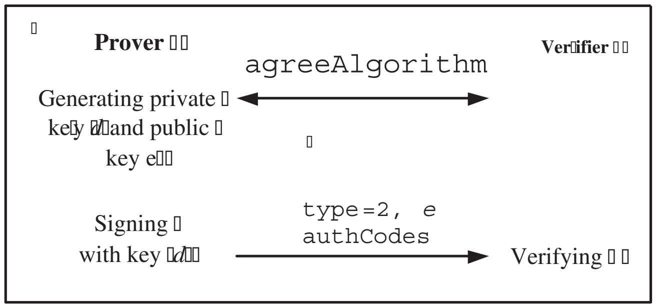

Fig. 5. Forgery method when type=2.

Indeed, it is unnecessary to authenticate the tools within one terminal since the tool is authenticated before it is installed.

D. *IPMP_SelectiveDecryptionInit*

For the sake of efficiency, IPMPX enables the protection of only portions of a digital content with the message `IPMP_SelectiveDecryptionInit`. As shown in Fig.6, the selective decryption syntax, including cipher segment and field segment, indicates how to decrypt the protected content. Unfortunately, this syntax has two major weaknesses.

```
class IPMP_SelectiveDecryptionInit
{
      ......
   bit(8) numBufs;
   for(i=0, i<numBufs; i++){
      Struct bufInfoStruct {
         bit(128) cipher_Id;
         //· · · cipherparameters · · ·
      }
   }
      ......
   bit(8) numFields;
   for(i=0, i< numFields; i++) {
      Struct fieldStruct {
         bit(8) field_Id;
         ......
            bit(1) sendMapTable;
            bit(1) isShuffled;
            if(sendMapTable){
               bit(16) sizeMapTable;
               bit(16) mappingTable[sizeMapTable]
            }
            if(isShuffled){
               ByteArray shuffleSpecificInfo;
            }
      }
   }
      ......
}
```

Fig. 6. `IPMP_SelectiveDecryptionInit` syntax [7]. The data structure `bufInfoStruct` is used to store the cipher parameters, and `fieldStructure` indicates the protected fields such as DC (Direct Coefficients), MV (Moving Vector), or AC (Alternative Coefficient)

1) Ambiguity: The scope of each cipher is unclear. For example, assume two ciphers (`numBufs`=2) DES and AES are employed, and three fields DC, MV and AC are protected (`numFields`=3). The simplified data structure is

DES, AES, DC, MV, AC.

It is not clear which of the two ciphers should be used to decrypt each of the protected fields.

In view of the fact that the *de facto* standard SSL (Secure Socket Layer) employs only one cipher for data delivery in secure network applications such as Internet banking and government services, it is doubtful that there is really a necessity for more than one cipher for protecting a stream. Even if there are extraordinary circumstances that warrant two or more ciphers, the syntax should be modified to allow multiple pairs of <protected field, cipher> to clarify the association.

2) Redundancy: The rationale for defining mapping table is not well founded. If mapping tables are sent on an unsecured channel, they should be part of the decoder other than the decrypter; otherwise, the table should be protected by a secret key. The same comment applies to the `shuffle` parameter.

Furthermore, just one of `shuffle` or `mapping table` needs to be provided for, since shuffle is just a special mapping table. Therefore, if both features are needed, shuffle can be merged into `mapping table`.

IV. Conclusion

The IPMP framework allows the coexistence of different vendors' tools, and the secure communication among the tools, via a messaging interface. The framework also provides the ability to download and integrate the tools into an MPEG-4 terminal. This paper highlights several weaknesses in the IPMPX syntax (most of those weaknesses exist in MPEG-2 IPMP [16] too), and proposes suitable remedies.

References

[1] Forrester Research, "Content Out Of Control", Sept. 2000

[2] N. Rump, "Can digital rights management be standardized?" IEEE Signal Processing Magazine, 21(2):63-70, 2004

[3] ISO/IEC 14496-1:2001(E), Information technology - Coding of audio-visual objects - Part 1: System, ISO/IEC JTC 1/SC 29/WG 11 N3850, 2000-10-19

[4] Fernando Pereira, Touradj Ebrahimi (ed.), The MPEG-4 Book, ISBN: 0130616214, Pearson Education, 2002

[5] ISO/IEC 14496-1:2001/FDAM 3:2003(E), Information technology - Coding of audio-visual objects - Part 1: Systems, AMENDMENT 3: Intellectual Property Management and Protection (IPMP) extensions, ISO/IEC JTC 1/SC 29/WG 11, 2002-12-4

[6] James King and Panos Kudumakis, "MPEG-4 IPMP Extension," DRM 2001, LNCS 2320, pp.126-140, 2002

[7] ISO/IEC 14496-13:2004(E), Information technology - Coding of audio-visual objects - Part 13: Intellectual Property Management and Protection (IPMP), extensions, SC 29/WG 11 N 5284, 2004-05-21

[8] T. Senoh, T. Ueno, T. Kogure, Shengmei Shen, Nfing Ji, Jing Liu, Zhongyang Huang, C. A. Schultz, "DRM renewability & interoperability," First IEEE Consumer Communications and Networking Conference (CCNC), pp.424-429, 2004

[9] MIRADOR: MPEG 4 Intellectual Property Rights by Adducing and Ordering, `http://www.cordis.lu/infowin/acts/analysys/products/thematic/mpeg4/mirador/mirador.htm`

[10] J. Lacy, N. Rump, T. Shamoon, and P. Kudumakis, "MPEG-4 Intellectual Property Management & Protection," 17th Conf. Audio Engineering Society, 1999.

[11] Kwang Yong Kim, JinWoo Hong, "MPEG4 IPMP authoring system for protection of object based contents," The 6th International Conference on Advanced Communication Technology, Vol.1, pp.499 - 503, 2004

[12] A. Menezes, P. Van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, pp.41, CRC Press, 1996

[13] David Anderson, "Distributed Denial Of Service Attacks," `http://wind.lcs.mit.edu/~dga/ddos.txt`

[14] Michael, Brick, and Kevin Max, "In the Wake of Web-Site Hacking, No Easy Answers, or Solutions," NYTimes.com/TheStreet.com, Feb. 9, 2000 `http://www.nytimes.com/library/tech/00/02/biztech/articles/10attack.html`

[15] C. S. Collberg , C. Thomborson ,and D. Low, "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs," ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1998

[16] ISO/IEC FDIS 13818-11:2003(E), Information technology - Generic coding of moving pictures and associated audio information - Part 11: IPMP on MPEG-2 systems, ISO/IEC JTC 1/SC 29/WG 11, 2003-04-17.