On the current state of interoperable content protection for internet video streaming

Stefan Pham, Franck Russel Kuipou, Stefan Arbanowski, Stephan Steglich
Future Applications and Media
Fraunhofer FOKUS
Berlin, Germany

***Abstract*— Over-the-top (OTT) media delivery or internet video streaming subscription services like Netflix and Amazon have become highly successful. Broadcasters and telecommunication companies are increasingly investing in creating video streaming platforms. This paper gives an overview of current media streaming and content protection standards. One of the main challenges for commercial content providers is to stream to as many devices (mobile, desktop, TV, etc.) as possible. Standards such as MPEG-DASH or HLS and related standards help to achieve this in an efficient and interoperable way. On the device side, different application platforms exist with different playback models. In order to distribute premium content, Digital Rights Management (DRM) systems are needed to protect the media streams. Using only one DRM system to protect the content is not enough to reach all relevant devices, because a platform or device is usually tied to the vendor's DRM. As a result, a multi-DRM ecosystem is needed for OTT delivery in order to protect content with more than one DRM system - the MPEG Common Encryption (CENC) standard enables this. In a multi-DRM backend, different entities exist that exchange sensitive metadata such as DRM-specific information and encryption keys. This communication can be interoperable following the Content Protection Information Exchange Format (CPIX) specification by the DASH-IF.**

Keywords—CPIX; CENC; DASH; HLS; CMAF

I. INTRODUCTION

Commercial video streaming services require content protection in order to meet requirements of copyright holders. Given that the Internet is used for the distribution of content and that it is used in many ways (such as file sharing) to get free access to copyrighted media content, the common solution to deal with content piracy is Digital Rights Management (DRM). DRM is an access-control technology used by manufacturers, publishers, and copyright holders to limit the usage of digital devices or information [1]. Using only one DRM to protect content is not enough, because this particular DRM might not be supported on another device or platform, because a platform or device is usually tied to the vendor's DRM. For OTT streaming, Microsoft PlayReady, Google Widevine and Apple FairPlay are the most relevant DRM systems. In order to create content that is protected with multiple DRM systems, multiple DRM license servers are needed. The entities involved in content creation have to exchange sensitive information. This can be accomplished using standardized interfaces following the Content Protection Information Exchange Format (CPIX) [2].

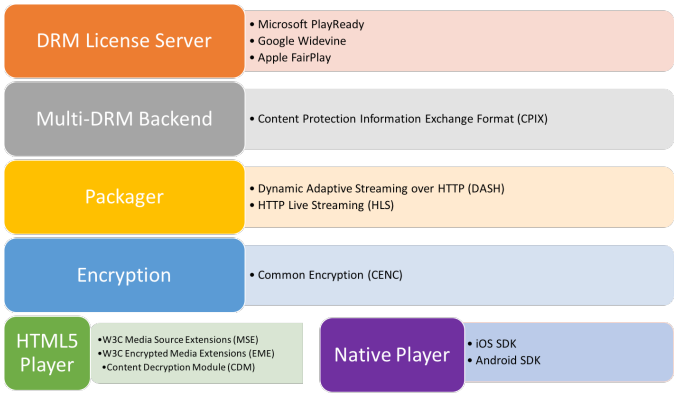

Figure 1 OTT DRM Video Streaming Stack

A packager, which creates the adaptive streaming manifest and segments should support MPEG-DASH or Dynamic Adaptive Streaming over HTTP (DASH), which was ratified as an international standard (ISO/IEC 23009-1) [3] in 2011. Previously, HLS (HTTP Live Streaming) was developed by Apple and initially released in 2009. Today, these two standards can be considered established in the OTT market, because they are supported on all major devices (iOS, Android), Web browsers, Smart TVs and streaming devices (Apple TV, Fire TV, Chromecast etc.). Moreover, with the introduction of MPEG CMAF (Common Media Application Format), DASH and HLS will support the same media format, namely ISOBMFF (ISO Base media file format). Common Encryption (CENC) [4], which is a standard for content encryption and key mapping, is compatible with DASH and HLS. CENC establishes an interoperable DRM ecosystem by enabling support for multiple DRM systems in the same streaming file format (e.g. ISOBMFF).

On the client side, we differentiate between applications running in an HTML5 Web browser and those running in a native environment (e.g. iOS and Android). When targeting the HTML5 Web browser on a device, the W3C Media Source Extensions (MSE) and Encrypted Media Extensions (EME) become relevant. MSE enables Web apps to consume ISOBMFF-based content (DASH or HLS), while EME enables Web applications to acquire licenses from a DRM license server.

Fig. 1 summarizes the aforementioned standards, which are relevant in today's OTT video streaming stack with DRM. In this paper, we focus on the multi-DRM backend and the CPIX standard as an enabler for interoperable multi-DRM backend communication. In the following chapter, we discuss the need for a multi-DRM backend. In chapter III related work in this area is given. Chapter IV explains the CPIX standard and chapter V describes the implementation of a multi-DRM backend. We conclude the paper in chapter VI.

II. Current State of OTT Streaming with DRM

As can be seen in the Table 1, clients on various platforms support different DRM systems, hence the need for a multi-DRM backend in order to playback DRM-protected streams on all platforms. We will focus on HTML5 browser-based playback. Nevertheless, a multi-DRM backend is also needed for apps running in a native environment (e.g. iOS or Android).

The Netscape Plugin Application Programming Interface (NPAPI) is deprecated in most browsers. Plug-ins like Microsoft Silverlight used NPAPI to run within a Web application. Flash plug-ins, which were also used for streaming, are now also disabled by default in the most common browsers. As a result, Web apps need to use the HTML5 media extensions MSE and EME. The DRM client is typically implemented as a CDM (Content Decryption Module). As platforms implement different CDMs, a Web app has to query the available DRM system and communicate with the corresponding DRM license server through the EME API. On the one hand, media playback on Web platforms can be accomplished with the help of HTML5 media extensions MSE and EME (often referred to as Type 3 streaming [5]). On the other hand, if MSE and EME are not available, Type 1 streaming can be leveraged. Type 1 streaming refers to native support for a streaming format (e.g. DASH or HLS) on a client, which is usually implemented through HTML5 <video> or a proprietary <object> video element.

Client	Platform	MSE	EME	PR	WV	FP
Chrome	Desktop	✓	✓	×	✓	×
Firefox	Desktop	✓	✓	×	✓	×
Safari	Desktop	✓	✓	×	×	✓
Edge	Desktop	✓	✓	✓	×	×
Android	Mobile	✓	✓	×	✓	×
iOS	Mobile	×	✓	×	×	✓
Apple TV	STB	×	×	×	×	✓
Fire TV	STB	✓	✓	✓	✓	×
Chromecast	STB	✓	✓	✓	✓	×
Samsung	Smart TV	✓	✓	✓	✓	×
LG	Smart TV	✓	✓	✓	✓	×
Panasonic	Smart TV	✓	✓	✓	✓	×
Philips	Smart TV	✓	✓	✓	✓	×
Sony	Smart TV	✓	✓	✓	✓	×

Table 1 Support of MSE, EME and DRM systems across platforms as of 01-26-2017. PR = Playready, WV = Widevine, FP = Fairplay

In order to reach all platforms as listed in Table 1 a multi-DRM backend should include license servers from Microsoft PlayReady, Google Widevine and Apple FairPlay.

There is another level of fragmentation that needs to be considered: On the encryption level, CENC allows to encrypt the content once for multiple DRM systems. However, DRM systems such as PlayReady and Widevine use AES-CTR to encrypt the media content. Apple's Fairplay requires a different encryption mechanism (AES-CBC). The DRM industry is currently converging to a common AES encryption mode eventually (AES-CBC), which will lead to a true "common encryption". However, it will take time until updated PlayReady and Widevine DRM client implementations are integrated into devices. From content provider perspective, there will be legacy devices that do not get updated. As a result, the encrypted media samples have to be duplicated during content creation, in order to support DRM clients that support either AES-CTR or AES-CBC.

III. Related Work

CPIX is enabled by MPEG standards Common Encryption (CENC) [4] and MPEG-DASH [3]. Moreover, it is compatible with HLS (HTTP Live Streaming).

There are many published scientific papers on interoperable DRM ecosystem such as [8], [7] or [8]. However, none of these papers focus on the entities of a multi-DRM backend and how they exchange information during the creation of content. This paper shows how this exchange can be accomplished with the help of a standardized approach like CPIX.

IV. CPIX

A typical OTT DRM architecture (see Fig. 2) has the following entities:

- *Packager*: an entity, which defines the structure of the media files.
- *Encryptor*: an entity, which encrypts the media files.
- *MPD Generator*: an entity, which creates the DASH MPD.
- *DRM Client*: gets information from media files or MPD and DRM license server to play the content.
- *DRM (License) Server*: deliver license to the DRM client.
- *Key Database*: stores encryption/content keys, referenced by Key IDs (KID)

During the creation of content, these entities have to exchange sensitive data such as encryption key and DRM-specific information. A standardized approach for a multi-DRM backend like CPIX, specifies the interfaces between the entities. This enables exchangeability of entities.

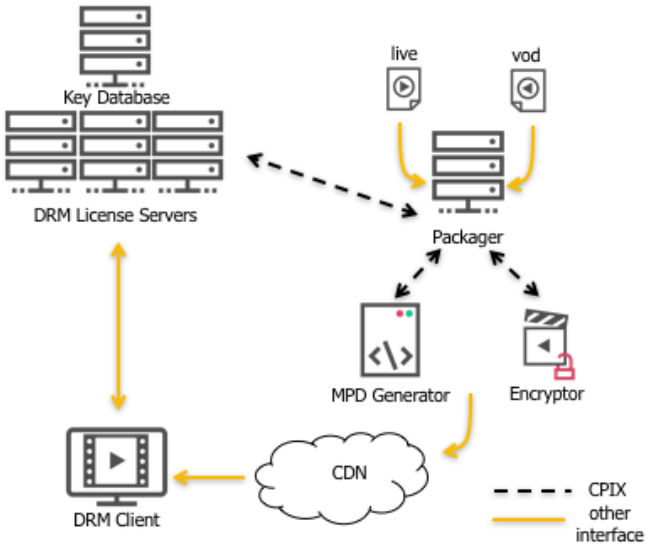

Figure 2 High-level Architecture

Content protection information consists of encryption keys and DRM-specific information. The format for DRM-specific information is specified in CENC as Protection System Specific Header (pssh). This information has to be exchanged between entities when an encrypted content is created. To facilitate this, the DASH Industry Forum has published the Content Protection Information Exchange Format (CPIX), which aims to standardize the way entities involved in the content creation workflow exchange protection information. The CPIX format is an XML file. The CPIX document can be used with any streaming format for on demand, as well as for live content. It includes the following elements:

- **CPIX:** This root element contains all necessary information to get the encryption key(s) and DRM-specific information. It has two optional attributes. An attribute "id" that specifies the identifier for this presentation and an attribute "name" that specifies the name of the presentation.
- **DeliveryDataList:** It is not mandatory and only required when the information in the CPIX file is encrypted. It contains DeliveryData elements. Each element specifies an entity authorized to decrypt the content keys in the CPIX document. It also contains all the information needed for the decryption of the content keys.
- **ContentKeyList:** It contains ContentKey elements. Each element has the key used to encrypt the content.
- **DRMSystemList:** It contains DRMSystem elements. There is a single DRMSystem element per DRM Server that is used to protect the media content. Information such as a pssh is saved here.
- **ContentKeyPeriodList:** It contains ContentKeyPeriod elements. Each element defines a period of time in which a Content Key is used.
- **ContentKeyUsageRuleList:** It contains ContentKeyUsageRule elements. Each element maps a Content Key to a specific context. For example, a specific content key should be used for UHD content.
- **UpdateHistoryItemList:** It contains UpdateHistoryItem elements. Each element contains information, such as which entity has made which changes and when.
- **Signature:** It contains the signature of the CPIX document or of the part that was signed.

Although CPIX allows exchanging data without additional encryption, it is recommended to protect them, even if the file is sent over a secure protocol like HTTPS. Therefore, the following key hierarchy is used to protect the sensitive data (encryption key, pssh information) within the document itself. For each CPIX document a document key is used to encrypt each (CENC) content key using the AES256-CBC PKCS #7 padding algorithm. The document key is a 256-bit key and is part of each DeliveryData element. It is itself encrypted with the Delivery Key of each receiving entity using the RSA-OAEP-MGF1-SHA1 algorithm. The Delivery Key of a receiving entity is the public key of its key pair. The Delivery Key is saved in the CPIX document in the DeliveryData element as the X509 certificate of the recipient.

A CPIX document can be created from any entity. Therefore, different workflows are possible depending on the existing architecture and entities of a content provider.

In a simple workflow, only one entity produces the CPIX file. For example, the *Encryptor* produces the CPIX file. Prior to that public keys are exchanged between the *Encryptor* and the *DRM servers*. Once this is done, it can then create and protect the CPIX document using the above key hierarchy. On receiving the file, the *DRM servers* can decrypt the content keys using their private key, write their DRM-specific information into the file, and send it back if the *Encryptor* needs such information, or they can just save the content keys. Fig. 3 illustrates this workflow.

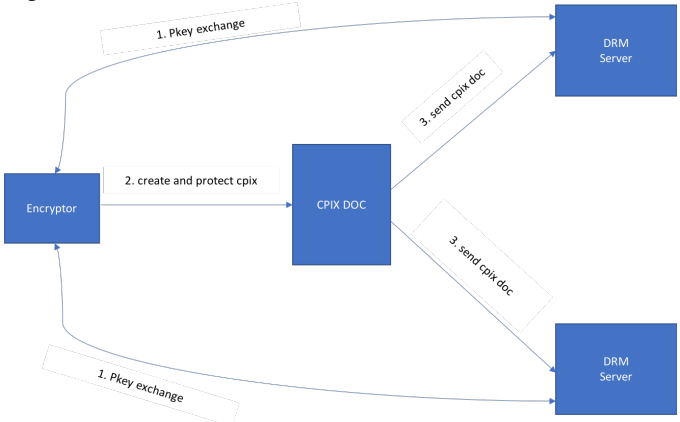

Figure 3 Simple workflow: Encryptor as a producer. Public Keys need to be exchanged prior before the first CPIX Document is Sent. Based on [2]

In a more complex workflow, more than one entity consumes the CPIX file and more than one produces it. In an example with DASH content, the *Packager*, *Encryptor* and the *DRM*

server exchange public keys. The *Packager* defines the structure of the media file, creates a CPIX document (CPIX v1), and forwards the document to the *Encryptor*. The *Encryptor* generates and encrypts the content key with the *DRM server's* public key, adds it to the CPIX document (CPIX v2), and forwards it to the *DRM Server*. The *DRM Server* decrypts the content key with its private key, saves the content key, creates a "DRMSystem" element containing DRM-specific information, protects these information with the *Encryptor's* and *MPD Generator*'s public key, adds the "DRMSystem" element to the CPIX file (CPIX v3), and sends it to the *Encryptor* and the *MPD Generator*. Each entity records its changes in the UpdateHistoryItem elements. The *Encryptor* uses information in the "DRMSystem" element to write a "pssh" box in the media file. The *MPD Generator* uses the same information to create a "ContentProtection" element in the MPD. In this example workflow, the *Packager* and the *DRM Server* are producers, *Encryptor* and *MPD Generator* are consumers. Fig. 4 illustrates the workflow.

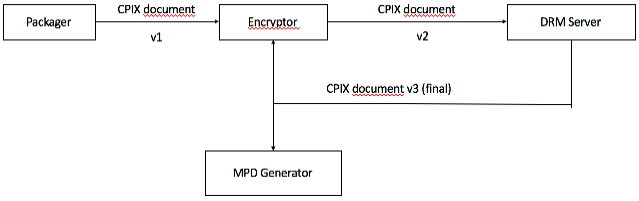

Figure 4 Workflow with multiple producers and consumers workflow. Based on [2]

Each workflow has its advantages and disadvantages. The one that is chosen depends on the requirements of the existing architecture. For example, if the *Encryptor* knows the format of the DRM-specific pssh, it is more efficient to let it create the CPIX document and it thus becomes the producer.

On the other hand, if it is expected that a new DRM system will be added, it is better that the *Encryptor* just produces the content keys and receives the pssh information from the *DRM Servers*. A new *DRM Server* can be added without the need for the *Encryptor* to communicate with the new *DRM Server*.

In a complex workflow where many producers and consumers exist, CPIX offers the most advantages. Likewise, all new information can be easily recorded and read from the entities that need them. The number of messages sent between the entities is reduced because a lot of information can be sent at once.

V. Multi-DRM Backend Implementation

Based on our implementation experience of an interoperable multi-DRM backend, creating multi-DRM content with CENC means encrypting the media sample, writing CENC directives into the MPD and adding each DRM-specific pssh into the media file.

The protection systems used were: Microsoft PlayReady, Google Widevine and W3C ClearKey. Note that ClearKey should not be used in a productive environment, since the content key is not embedded in a license but directly delivered in the clear to the client. ClearKey was used for testing purposes, in order to interact with multiple DRMs and because the EME specification mandates all browsers supporting EME to implement ClearKey.

Once content with a single DRM can be created, adding multiple DRM systems into a media file becomes trivial. One of the challenges in implementing a multi-DRM backend is how to deliver encryption keys and KIDs across all DRM servers, and to get the DRM-specific protection information from the DRM server securely.

For the exchange of sensitive information between entities, HTTPS in combination with a RESTful API, can be used to secure messages in the transport layer. Moreover, a trust relationship using public/private keys between entities in the multi-DRM backend is recommended, in order to encrypt the messages that are exchanged. For details see chapter 4 Key Management of the CPIX specification [2].

A simple algorithm, which avoids the need to send the encryption key is defined by Microsoft. It can be used to generate a content key based on a unique key seed (a secret value in bytes) and the KID defined by the content user or implementer. It works by hashing the KID and the key seed for three times. After that, the resulting values are XORed to create the content key. A detailed description can be found in [9]. This algorithm can be used to solve the key sharing problem. In fact, the entities involved in the content creation like *Encryptor*, and the *DRM Servers* exchange a unique key seed and implement the hashing algorithm. When the encryption key is needed, they can use this algorithm to generate it. This makes sure that they always have the same content key and therefore do not need to send the clear content key over the network.

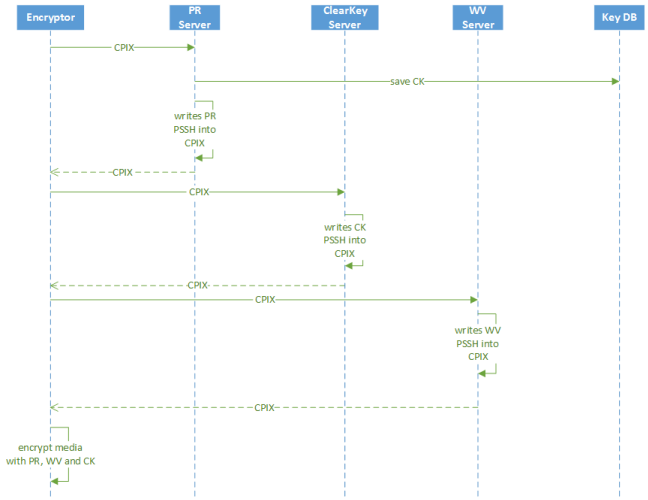

Figure 5 Protection of DASH content with PlayReady, Widevine and ClearKey using CPIX. CK = Content Key

An example workflow where clear content is protected with PlayReady, Widevine and ClearKey is shown in Fig. 5. Note, that in this example the encryption/content key is not additionally secured via a public/private key mechanism or the aforementioned hash algorithm. In the example the *Encryptor*,

after creating the CPIX document and writing the content key into it, sends a POST request to the PlayReady server containing the CPIX file as body. The PlayReady server parses it, saves the content key it gets from parsing the file into the *Key Database* if the key was not saved by another server, creates the PSSH, adds the PSSH into CPIX file, and sends it back as a response to the *Encryptor*. The preceding process is repeated for the Widevine and ClearKey server. After receiving the file from the Widevine server, the *Encryptor* can create content protected with PlayReady and Widevine.

Another example where CPIX offers advantages is when a new protection scheme has to be added to an already encrypted content.

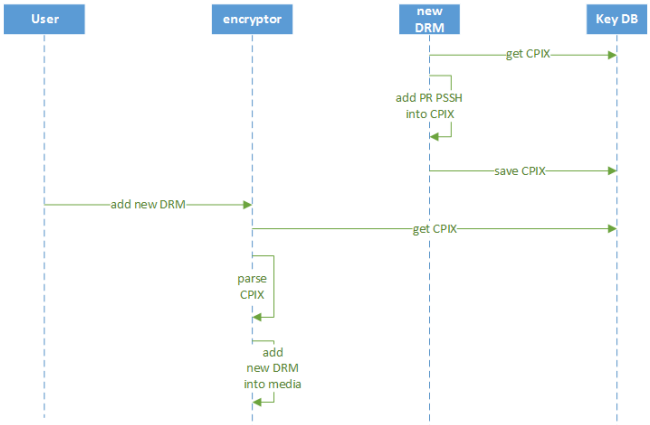

Figure 6 Adding a new DRM system to already encrypted content by using CPIX

In the example in Fig. 6, the new DRM system just gets the CPIX file from the *Key Database* and adds its DRM-specific information into the CPIX document. When the *Encryptor* is notified to add the new DRM, it just connects to the *Key Database*, parses the CPIX document, and gets the new DRM-specific information. The *Encryptor* can then write this new information into the media content or in case of DASH into the MPD.

With this approach, more than one bit of information, like a complete PSSH box and "ContentProtection" element for the MPD can be sent at once. New information can be easily tracked. For example, if PSSH information of a DRM has changed (e.g. security restrictions have changed), the DRM signals this by updating the CPIX file. This update can be directly seen from other entities by reading the "UpdateHistoryItem" element that saves when and which entity made a change in the CPIX file.

VI. Conclusion

With the deprecation of NPAPI and Flash, HTML5 will soon completely replace Adobe Flash and Microsoft Silverlight in all browsers. HTML5 playback refers to usage of MSE and EME to play encrypted content. Due to the fact that each major browser vendor implements a different CDM and therefore, a different DRM system, content providers cannot avoid to implement a multi-DRM system to reach a majority of platforms. To deploy such a system, entities involved in the content creation have to exchange sensitive data like content key and DRM-specific information. By using a standardized approach like CPIX, transparency, scalability, interoperability, and easy exchangeability of entities can be achieved.

References

[1] EC-Council, Computer Forensics Investigating Network Instructions and Cyber Crime, 1st ed., Cengage Learning, 2009.

[2] Content Protection Information Exchange Format (CPIX), DASH-IF Impelementation Guidelines v2.0, 2016.

[3] Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media Presentation description and segment formats. International Standard, ISO/IEC 23009-1:2014, 2014.

[4] Information technology – MPEG systems technologies – Part 7: Common Encryption in ISO base media file format files, ISO/IEC 23001-7:2014, 2014.

[5] D. Mebane and J. Smith. (2015, January 29). Simplified Adaptive Video Streaming: Announcing support for HLS and DASH in Windows 10 [Online]. Available: https://blogs.msdn.microsoft.com/ie/2015/01/29/simplified-adaptive-video-streaming-announcing-support-for-hls-and-dash-in-windows-10/

[6] S. Kaiser and S. Pham, "DRM-interoperable MPEG-DASH end-to-end architecture," in Multimedia and Expo Workshops (ICMEW), Chengdu, China, 2014.

[7] G. L. Heileman and P.A. Jamkhedkar, "DRM interoperability analysis from the perspective of a layered framework", in Proceedings of the Fifth ACM Workshop on Digital Rights Management, Alexandria, VA, USA, 2005.

[8] A. Mikityuk et al., "Content Protection in HTML5 TV Platforms: Towards Browser-agnostic DRM and Cloud UI Environments", in TrustED '15 Proceedings of the 5th International Workshop on Trustworthy Embedded Devices, Denver, CO, 2015, pp. 43-52.

[9] Microsoft Corporation. (2015, August). PlayReady Header Object [Online]. Available: https://www.microsoft.com/playready/documents/