TECHNICAL SPECIFICATION

Community Review

DASH-IF Interoperability Points;
Part 6: Content Protection

DASH Industry Forum

3855 SW 153rd Dr.

Beaverton, OR 97003 - USA

Email : admin@dashif.org

Community Review

Important notice

The present document can be downloaded from:
http://www.dashif.org/guidelines

Contents

Community Review

Intellectual Property Rights

Disclaimer

This is a document made available by DASH-IF. The technology embodied in this document may involve the use of intellectual property rights, including patents and patent applications owned or controlled by any of the authors or developers of this document. No patent license, either implied or express, is granted to you by this document. DASH-IF has made no search or investigation for such rights and DASH-IF disclaims any duty to do so. The rights and obligations which apply to DASH-IF documents, as such rights and obligations are set forth and defined in the DASH-IF Bylaws and IPR Policy including, but not limited to, patent and other intellectual property license rights and obligations. A copy of the DASH-IF Bylaws and IPR Policy can be obtained at http://dashif.org/.

The material contained herein is provided on an "AS IS" basis and to the maximum extent pe mitted by applicable law, this material is provided AS IS, and the authors and developers of this material and DASH-IF hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of workmanlike effort, and of lack of negligence.

In addition, this document may include references to documents and/or technologies controlled by third parties. Those third-party documents and technologies may be subject to third party rules and licensing terms. No intellectual property license, either implied or express, to any third-party material is granted to you by this document or DASH-IF. DASH-IF makes no warranty whatsoever for such third-party material.

Note that technologies included in this document and for which no test and conformance material is provided, are only published as candidate technologies, and may be removed if no test material is provided before releasing a new version of this guidelines document. For the availability of test material, please check https://www.dashif.org.

Community Review

Foreword

This Technical Specification (TS) has been produced by the DASH-IF Technical Working Group.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in deliverables except when used in direct citation.

Executive summary

In this document, content protection ande security guidelines for encrypted content are described. It is an update to DASH-IF IOP Guidelines version 4.3. It adds the support for the `cbcs` protection scheme. It clarifies the DASH-IF XML schema Finally it introduces a new option using `seig` but without key hierarchy for supporting key rotation and the Enhanced Clear Key Content Protection (ECCP), a content protection mechanism for DASH content which provides greater protection than TLS delivery, token authentication or Clear Key used individually.

Introduction

This document is Part 6 of a multipart set of documents, collectively called “IOP V5.0.0”. All the parts are:

1. Overview, architecture and interfaces
2. Core principles and CMAF mapping
3. DASH on-demand services
4. DASH live and low-latency live services
5. Ad insertion and content replacement
6. Content protection (this document)
7. Video
8. Audio
9. Text
10. Events
11. Additional functionalities
12. Conformance and reference tools

The guidelines defined in this document support the creation of protected interoperable services for video distribution based on MPEG-DASH and related standards. These guidelines are provided in order to address DASH-IF members' needs and industry best practices. The guidelines support the implementation of conforming service offerings as well as DASH client implementations.

While alternative interpretations may be equally valid in terms of standards conformance, services and clients created following the guidelines defined in this document can be expected to exhibit highly interoperable behavior between different implementations.

Community Review

1 Scope

This document is an update to the "Content Protection and Security" section of the DASH-IF IOP Guidelines version 4.3. The scope remains the same, giving guidelines for interoperable behaviors of clients in front of well formed encrypted content. Some updates have been made. This means:

- Updated encrypted content constraints for supporting CMAF. This includes the addition of the `cbcs` scheme support and recommendation for encrypting content when available using both `cbcs` and `cenc` protection schemes.

In addition, this document introduces:

- The DASH-IF XML schema where two elements are defined. These elements are namely the **`Laurl`** (License acquisition server URL) and **`Authzurl`** (Authorization server URL).
- A new option using `seig` but without key hierarchy for supporting key rotation.
- The Enhanced Clear Key Content Protection (ECCP), a content protection mechanism for DASH content which provides greater protection than TLS delivery, token authentication or Clear Key used individually.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] Information technology — MPEG systems technologies — Part 7: Common encryption in ISO base media file format files. February 2016. Published. URL: https://www.iso.org/standard/68042.html.

[2] Information technology — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media. March 2020. Published. URL: https://www.iso.org/standard/79106.html.

[3] Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats. December 2019. Published. URL: https://www.iso.org/standard/79329.html.

[4] DASH-IF Interoperability Points; Part 1: Overview, Architecture and Interfaces. DASH-IF IOP-1 V5.0.0 (2021-xx). URL: https://dashif.org/guidelines/

[5] DASH-IF Interoperability Points; Part 2: Core principles and CMAF mapping. DASH-IF IOP-1 V5.0.0 (2021-xx). URL: https://dashif.org/guidelines/

[6] David Dorwin; et al. "cenc" Initialization Data Format. 15 September 2016. NOTE. URL: https://www.w3.org/TR/eme-initdata-cenc.

[7] David Dorwin; et al. Encrypted Media Extensions. 18 September 2017. REC. URL: https://www.w3.org/TR/encrypted-media.

[8] Information technology — Coding of audio-visual objects — Part 12: ISO Base Media File Format. December 2015. International Standard. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/c068960_ISO_IEC_14496-12_2015.zip.

[9] M. Jones; J. Bradley; N. Sakimura. JSON Web Signature (JWS). May 2015. Proposed Standard. URL: https://datatracker.ietf.org/doc/html/rfc7515.

Community Review

[10] M. Jones; J. Bradley; N. Sakimura. JSON Web Token (JWT). May 2015. Proposed Standard. URL: https://datatracker.ietf.org/doc/html/rfc7519

[11] Mike West. Mixed Content. 2 August 2016. CR. URL: https://www.w3.org/TR/mixed-content/

[12] M. Nottingham; E. Wilde. Problem Details for HTTP APIs. March 2016. Proposed Standard. URL: https://datatracker.ietf.org/doc/html/rfc7807

[13] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. August 2018. Proposed Standard. URL: https://datatracker.ietf.org/doc/html/rfc8446

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[14] Enabling Low-Latency HLS. URL: https://developer.apple.com/documentation/http_live_streaming/protocol_extension_for_low-latency_hls_preliminary_specification.

[15] Information technology — MPEG systems technologies — Part 12: Sample variants. December 2018. Published. URL: https://www.iso.org/standard/74431.html

[16] M. Jones; J. Hildebrand. JSON Web Encryption (JWE). May 2015. Proposed Standard. URL: https://datatracker.ietf.org/doc/html/rfc7516

[17] PlayReady Content Encryption Modes. URL: https://docs.microsoft.com/en-us/playready/packaging/content-encryption-modes.

[18] DASH-IF Implementation Guidelines:Token-based Access Control for DASH(TAC), February 2019, URL: https://dashif.org/guidelines/

[19] URI Signing for CDN Interconnection (CDNI),R. van Brandenburg,K. Leung,P. Sorber, February 11, 2021, https://tools.ietf.org/html/draft-ietf-cdni-uri-signing-21.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

content: one or more audio-visual elementary streams and the associated MPD if in DASH format.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
CDM	Content Decryption Module
CDN	Content Delivery NEtwork
CMAF	Common Media Application Format
CPU	Central Processing Unit
DASH	Dynamic Adaptive Streaming over HTTP

Community Review

DRM	Digital Rights Management
ECDSA	Elliptic Curve Digital Signature Algorithm
EME	Encrypted Media Extension
HDCP	High-bandwidth Digital Content Protection
HMAC	Hash-based Message Authentication Code
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ISOBMFF	ISO Base Media File Format
JSON	JavaScript Object Notation
MPD	Media Presentation Description
MSE	Media Source Extensions
KID	Key Identifier
SHA	Secure Hash Algorithm
TLS	Transport Layer Security
UHD	Ultra High Definition
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

4 Core concepts of content protection and security

4.1 Introduction

A DRM system cooperates with the device's media platform to enable playback of encrypted content while protecting the decrypted samples and content keys against potential attacks. This document focuses on the signaling in the DASH presentation and on the interactions of the DASH client with other components. This document does not define any DRM system. DASH-IF maintains a registry of DRM system identifiers at https://dashif.org/identifiers/content_protection/.

Figure 1 provides a high level view of the different elements involved in the rendering of encrypted content.

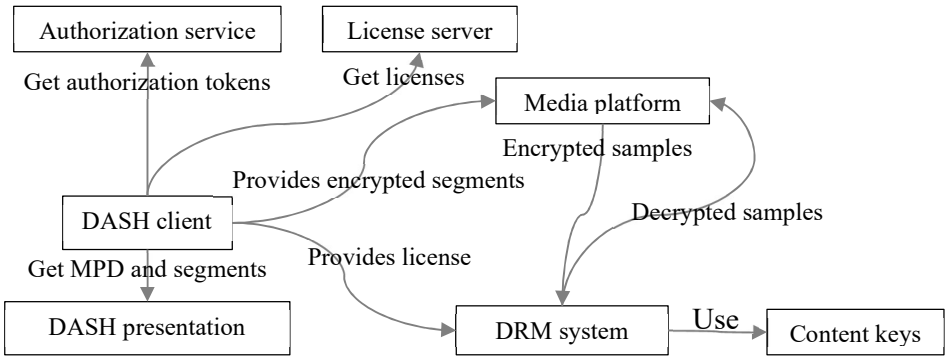


Figure 1: Core elements in content protection.

Community Review

Common Encryption [1] specifies several protection schemes which can be applied by a scrambling system and used by different DRM systems. The same encrypted DASH presentation can be decrypted by different DRM systems if a DASH client is provided the DRM system configuration for each DRM system, either in the MPD or at runtime.

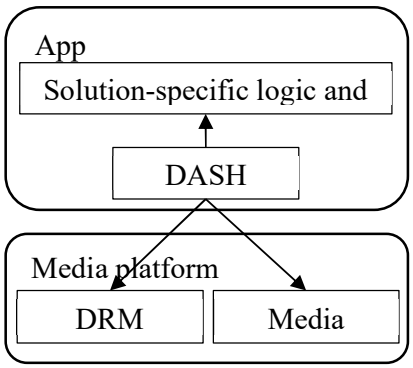
A content key is a key (usually 128-bit) used by a DRM system to make content available for playback. It is identified by a string called `default_KID` and/or `KID`. The format constraints of the string are defined in [1]. While the `default_KID` format visually resembles a UUID when in the MPD, it is not exactly the same. UUIDs have constraints on the byte values permitted at certain positions in the data structure, whereas [1] sets no constraints on the values in `default_KID`, it only defines the format of the string.

A content key and its identifier are shared between all DRM systems, whereas the mechanisms used for key acquisition and content protection are largely DRM system specific. DASH adaptation sets are often protected by different content keys.

A license is a data structure in a DRM system specific format that contains one or more content keys and associates them with a policy that governs the usage of the content keys (e.g. expiration time). The encapsulated content keys are typically encrypted and only readable by the DRM system.

4.2 Client reference architecture for content playback

Different software architectural components are involved in playback of encrypted content. The exact nature depends on the specific implementation. A high-level reference architecture is described in Figure 2.



Community Review

Figure 2: Reference architecture for encrypted content playback.

The media platform provides one or more APIs that allow the device's media playback and DRM capabilities to be used by a DASH client. The DASH client is typically a library included in an app. On some device types, the DASH client may be a part of the media platform.

This document assumes that the media platform exposes its encrypted content playback features via an API similar to W3C EME [7]. The technical nature of the API may be different but EME-equivalent functionality is expected.

The media platform often implements at least one DRM system. Additional DRM system implementations can be included as libraries in the app.

A DRM system is an implementation of content keys management. It is made of two main components: A license server for generating licenses and a DRM client for processing licenses and enforcing the associated policies. On some paltforms, the DRM client may handle the decryption of samples while on other platforms, decryption is handled by e.g. hardware elements the DRM client interacts with.

4.3 Robustness

DRM systems define rules, called compliance rules, that govern how they can be implemented. These rules can define different robustness levels which are typically used to differentiate implementations based on their resistance to attacks. The set of robustness levels, their names and the constraints that apply are all specific to each DRM system.

As an example, a hypothetical DRM system might define the following robustness levels:

- High: All cryptographic operations are performed on a separate CPU not accessible to the device's primary operating system (often called a trusted execution environment). Decrypted data only exists in a memory region not accessible to the device's primary operating system (often called a secure media path).
- Medium: All cryptographic operations are performed on a separate CPU not accessible to the device's primary operating system. Decrypted data may be passed to the primary operating system's media platform APIs.
- Low: All operations are performed in software that can be inspected and modified by the user. Software obfuscation techniques are to be used to protect against analysis.
- None: For development only. Implementation does not resist attacks.

Policy associated with content can require a DRM system implementation to conform to a certain robustness level, thereby ensuring that valuable content does not get presented on potentially vulnerable implementations. This policy can be enforced on different levels, depending on the DRM system:

1. A license server may refuse to provide content keys to implementations with unacceptable robustness levels.
2. The DRM system may refuse to use content keys whose license requires a higher robustness level than the implementation provides.

Multiple implementations of a DRM system may be available to a DASH client, potentially at different robustness levels. The DASH client has to choose at media load time which DRM system implementation to use. However, the required robustness level may be different for different device types and is not expressed in the MPD, only the expected robustness level for playing back content can be exposed in the MPD. This decision is a matter of policy and is impossible for a DASH client to determine on its own. Therefore, solution-specific logic and configuration inform the DASH client of the correct choice.

A DASH client shall enable solution-specific logic and configuration to specify the robustness level of the DRM system implementation to be used. Depending on which DRM system is used, this can be implemented by, for example, changing the mapping of DRM system to key system in EME-based implementations (see clause 4.4).

4.4 W3C Encrypted Media Extensions

Whereas the DRM signaling in DASH deals with DRM systems, EME deals with key systems. While similar in concept, they are not always the same thing. A single DRM system may be implemented on a single device by multiple different key systems, with different codec compatibility and functionality, potentially at different robustness levels.

As an example, a device may implement the "ExampleDRM" DRM system as a number of EME key systems:

- The key system "ExampleDRMvariant1" may support playback of encrypted H.264 and H.265 content at up to 1080p resolution with "low" robustness level.
- The key system "ExampleDRMvariant2" may support playback of encrypted H.264 content at up to 4K resolution with "high" robustness level.

Comments Review

- The key system "ExampleDRMvariant3" may support playback of encrypted H.265 content at up to 4K resolution with "high" robustness level.

Even if multiple variants are available, a DASH client should map each DRM system to a single key system. The default key system should be the one the DASH client expects to offer greatest compatibility with content (potentially at a low robustness level). The DASH client should allow solution-specific logic and configuration to override the key system chosen by default (e.g. to force the use of a high-robustness variant).

5 DASH-IF XML schema

5.1 Introduction

This clause defines elements that can be added in the MPD allowing to provide information on license acquisition and authorisation servers that are available for every DRM system.

The full schema is defined in Annex A. The following is only describing the elements.

5.2 License acquisition URL

One or more **`Laurl`** elements may be added under the **`ContentProtection`** descriptor. It contains the URL for a license server allowing to retrieve a license in the format specific to the DRM system that is described by the **`ContentProtection`** descriptor. It has the optional attribute `@licenseType` that is a string describing the license type served by this license server. The meaning of this attribute is DRM specific.

Examples can be found in clause 8.

5.3 Authorization server URL

One or more **`Authzurl`** elements may be added under the **`ContentProtection`** descriptor. It contains the URL for an authorization service allowing to receive authorization tokens that may be required for requesting a license from a license server.

Examples can be found in clause 7.3.

6 Content protection constraints for CMAF

6.1 Introduction

The structure of content protection related information in the CMAF containers used by DASH is largely specified by [1] (in particular clause 8) and [2]. This clause outlines some additional requirements to ensure interoperable behavior of DASH clients and services.

Note: This document uses the cenc: prefix to reference the XML namespace `urn:mpeg:cenc:2013` [1].

6.2 Content protection data

This clause describes the structure of content protection data in CMAF containers used to provide encrypted content in a DASH presentation, summarizing the requirements defined by [1], [2], [8] and other parts of DASH-IF implementation guidelines.

DASH initialization segments contain:

- Zero or more `moov/pssh` "Protection System Specific Header" boxes ([1] clause 8.1) which provide DRM system initialization data in DRM system specific format.
- Exactly one `moov/trak/mdia/minf/stbl/stsd/sinf/schm` "Scheme Type" box ([8] clause 8.12.5) identifying the protection scheme ([1] clause 4).

Community Review

- Exactly one `moov/trak/mdia/minf/stbl/stsd/sinf/schi/tenc` "Track Encryption" box ([1] clause 8.2) which contains default encryption parameters for samples. These default parameters may be overridden in media segments.

DASH media segments are composed of a single CMAF fragment that contains:

- Exactly one `moof/traf/senc` "Sample Encryption" box ([1] clause 7.2) which stores initialization vectors (IVs) and, optionally, subsample encryption ranges for samples in the same CMAF fragment.
- Zero or one `moof/traf/saiz` "Sample Auxiliary Information Size" boxes ([8] clause 8.7.8) which references the sizes of the per-sample data stored in the `moof/traf/senc` box ([1] clause 7 and [2] clause 8.2.2). It is omitted if the parameters provided by the `senc` box are identical for all samples in the CMAF fragment.
- Zero or one `moof/traf/saio` "Sample Auxiliary Information Offset" boxes ([8] clause 8.7.9) which references the sizes of the per-sample data stored in the `moof/traf/senc` box ([1] clause 7 and [2] clause 8.2.2). It is omitted if the parameters provided by the `senc` box are identical for all samples in the CMAF fragment.
- Zero or more `moof/pssh` "Protection System Specific Header" boxes ([1] clause 8.1) which provide transparent updates to DRM system internal state.
- For each sample group, exactly one `moof/traf/sgpd` "Sample Group Description" box ([1] clause 6 and [8] clause 8.9.3) which contains overrides for encryption parameters defined in the `tenc` box. It is omitted if no parameters are overridden.
- For each sample grouping type (see [8], typically one), exactly one `moof/traf/sbgp` "Sample to Group" box ([1] clause 6 and [8] clause 8.9.2) which associates samples with sample groups. It is omitted if no parameters are overridden.

6.3 Content protection data constraints

Initialization segments should not contain any `pssh` box ([2], section 7.4.3) and DASH clients may ignore such boxes when encountered. Instead, `pssh` boxes should be placed in the MPD as **`cenc:pssh`** elements in DRM system specific **`ContentProtection`** descriptors.

Note: Placing the `pssh` boxes in the MPD has become common for purposes of operational agility, it is often easier to update MPD files than rewrite initialization segments when, fir example, a new DRM system needs to be supported. Furthermore, in some scenarios the appropriate set of `pssh` boxes is not known when the initialization segment is created.

Protected content may be published without any `pssh` boxes in both the MPD and media segments. All DRM system configuration may be provided at runtime, including the `pssh` box data.

Media segments may contain `moof/pssh` boxes ([2] clause 7.4.3) to provide updates to DRM system internal state (e.g. to supply new leaf keys in a key hierarchy). See clause **Error! Reference source not found.** for an example. These state updates may be transparent to a DASH client on some media platforms that intercept the `moof/pssh` boxes and supply them directly to the active DRM system; on other media platforms, the DASH client may need to extract and forward the `moof/pssh` boxes to the DRM system. When using CMAF chunks for delivery, each CMAF fragment may be split into multiple CMAF chunks. If the CMAF fragment contained any `moof/pssh` boxes, copies of these boxes shall be present in each CMAF chunk that starts with an independent media sample.

Note: While DASH only requires the presence of `moof/pssh` in the first CMAF chunk, the requirement is more extensive in the interest of HLS interoperability [14].

6.4 Content encryption

A DASH presentation may provide some or all adaptation sets in encrypted form, requiring the use of a DRM system to decrypt the content for playback. The duty of a DRM system is to prevent

Community Review

disclosure of the content key and misuse of the decrypted content (e.g. recording via screen capture software) and may be to decrypt content.

In a DASH presentation, every representation in an adaptation set shall be protected using the same content key (identified by the same `default_KID` or `KID`). This means that if representations use different content keys, they shall be in different adaptation sets, even if they would otherwise (were they not encrypted) belong to the same adaptation set. A `urn:mpeg:dash:adaptation-set-switching:2016` supplemental property descriptor ([3] clause 5.3.3.5) shall be used to signal that such adaptation sets are suitable for switching.

Encrypted DASH content shall use either the `cenc` or the `cbcs` protection scheme defined in [1]. `cenc` and `cbcs` are two mutually exclusive protection schemes. DASH content encrypted according to the `cenc` protection scheme cannot be decrypted by a DRM system supporting only the `cbcs` protection scheme and vice versa.

Some DRM system implementations support both protection schemes. Even when this is the case, clients shall not concurrently consume encrypted content that uses different protection schemes.

Representations in the same adaptation set shall use the same protection scheme. Representations in different adaptation sets may use different protection schemes. If both protection schemes are used in the same period, all encrypted representations in that period shall be provided using both protection schemes. That is, the only permissible scenario for using both protection schemes together is to offer them as equal alternatives to target DASH clients with different capabilities.

Representations that contain the same media content using different protection schemes should use different content keys. This protects against some cryptographic attacks [17].

7 Content protection constraints for the MPD

7.1 Introduction

A DASH client needs to recognize encrypted content and activate a suitable DRM system, configuring it to decrypt content. The MPD informs a DASH client of the protection scheme used to protect content, identifies the content keys that are used and optionally provides the default DRM system configuration for a set of DRM systems.

The DRM system configuration is the complete data set required for a DASH client to activate a single DRM system and configure it to decrypt content using a single content key. It is supplied by a combination of XML elements in the MPD and/or solution-specific logic and configuration. The DRM system configuration often contains:

- DRM system initialization data in the form of a DRM system specific `pssh` box (as defined in [1]).
- DRM system initialization data in some other DRM system specific form (e.g. `keyids` JSON structure used by W3C Clear Key)
- The used protection scheme (`cenc` or `cbcs`)
- `default_KID` that identifies the content key
- License server URL
- Authorization service URL

7.2 Signaling encrypted content

The presence of a **`ContentProtection`** descriptor with `@schemeIdUri="urn:mpeg:dash:mp4protection:2011"` on an adaptation set informs a DASH client that all representations in the adaptation set are encrypted in conformance to Common Encryption ([1] clause 11 and [3] clauses 5.8.4.1 and 5.8.5.2) and require a DRM system to provide access.

Community Review

This descriptor is present for all encrypted content ([3] clause 5.8.4.1). It shall be defined on the adaptation set level. The value attribute indicates the used protection scheme ([3] clause 5.8.5.2). The `@cenc:default_KID` shall be present if the media segment do not carry `sgpd` boxes that contain `KID` overwritting the `default_KID` value and have a value matching the `default_KID` in the `tenc` box.

As an example, the following snippet of an MPD signals an adaptation set encrypted using the `cbcs` scheme and with a content key identified by `34e5db32-8625-47cd-ba06-68fca0655a72`.

```
<ContentProtection
    schemeIdUri="urn:mpeg:dash:mp4protection:2011"
    value="cbcs"
    cenc:default_KID="34e5db32-8625-47cd-ba06-68fca0655a72" />
```

The `tenc` box stores `default_KID` as a 16-byte array. The byte order shall be identical in the binary structure and the string-form `@cenc:default_KID`.

The following are two equivalent forms of representing the same `default_KID`:

- In string form: `00010203-0405-0607-0809-0a0b0c0d0e0f`
- In binary form: `0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f`

Some Windows-targeting software libraries implement "Microsoft style" UUID serialization that changes the order of bytes when transforming between string form and binary form. This is not appropriate when serializing/deserializing `default_KID` values. Linux-based tooling typically does not change the byte order.

7.3 Signaling DRM system information

A DASH service should supply DRM system configuration in the MPD for all supported DRM systems in all encrypted adaptation sets. This enables playback without the need for DASH client customization or additional client-side configuration. DRM system configuration may also be supplied by solution-specific logic and configuration, replacing or enhancing the defaults provided in the MPD.

Any number of **`ContentProtection`** descriptors ([3] clause 5.8.4.1) may be present in the MPD to provide DRM system configuration. These descriptors shall be defined on the adaptation set level. The contents may be ignored by the DASH client if overridden by solution-specific logic and configuration, the DRM system configuration in the MPD simply provides default values known at content authoring time.

A **`ContentProtection`** descriptor providing a default DRM system configuration shall use `@schemeIdUri="urn:uuid:<systemid>"` to identify the DRM system, with the `<systemid>` matching a value in the DASH-IF system-specific identifier registry. The `@value` attribute of the **`ContentProtection`** descriptor should contain the DRM system name and version number in a human readable form (for diagnostic purposes).

Each DRM system specific **`ContentProtection`** descriptor should contain a mix of XML elements and attributes defined by [1], the DRM system author, DASH-IF (see clause 5) or any other party.

DRM systems generally use the concept of license requests as the mechanism for obtaining content keys and associated usage constraints. For DRM systems that use this concept, one or more **`dashif:Laurl`** elements should be present under the ContentProtection descriptor, with the value of the element being the URL to send license requests to. This URL may contain content identifiers, see clause 7.4 for more details.

For DRM systems that require proof of authorization to be attached to the license request, one or more **`dashif:Authzurl`** elements should be present under the **`ContentProtection`** descriptor, containing the default URL to send authorization requests to.

Community review

Multiple **dashif:Laurl** or **dashif:Authzurl** elements under the same **ContentProtection** descriptor define sets of equivalent alternatives for the DASH client to choose from. A DASH client should select a random item from the set every time the value of such an element is used if the attributes of the elements have the same values.

The following is an example of a **ContentProtection** descriptor that provides default DRM system configuration for a fictional DRM system.

```
<ContentProtection
    schemeIdUri="urn:uuid:d0ee2730-09b5-459f-8452-200e52b37567"
    value="FirstDRM 2.0">
    <cenc:pssh>
        YmFzZTY0IGVuY29kZWQgY29udGVudHMgb2YgkXBzc2iSIGJveCB3aXRoIHRoaXMgU3lzdGVtSUQ=
    </cenc:pssh>
    <dashif:Authzurl>https://example.com/tenants/5341/authorize</dashif:Authzurl>
    <dashif:Laurl>https://example.com/AcquireLicense</dashif:Laurl>
</ContentProtection>
```

The presence of a DRM system specific **ContentProtection** descriptor is not required in order to activate the DRM system; these descriptors are used merely to provide the default DRM system configuration.

The contents of DRM system specific **ContentProtection** descriptors with the same `schemeIdUri` shall be identical in all adaptation sets with the same `default_KID`. This means that a DRM system will treat equally all adaptation sets that use the same content key.

Note: For changing the default DRM system configuration associated with a content key, all the instances where the data is present in the MPD have to be updated. For live services, this can mean updating the data in multiple periods.

7.4 Using a content ID

The concept of a content ID is sometimes used to identify groups of content keys based on solution-specific associations. This document supports it if the solution architecture demands it.

In order to make use of a content ID in DRM workflows, the content ID should be embedded into authorization service URLs and/or license server URLs (depending on which components are used and require the use of the content ID). This may be done either directly at MPD authoring time (if the URLs and content ID are known at such time) or by solution-specific logic and configuration at runtime.

Having embedded the content ID in the URL, all DRM workflows continue to operate the same as they normally would, except now they also include knowledge of the content ID in each request to the authorization service and/or license server. The content ID is an addition to the license request workflows and does not replace any existing data.

Embedding a content ID allows the service handling the request to use the content ID in its business logic. However, the presence of a content ID in the URL does not invalidate any requirements related to the processing of the `default_KID` and `KID` values of content keys.

No generic URL template for embedding the content ID is defined, as the content ID is always a proprietary concept. Recommended options include:

- Query string parameters:
  `https://example.com/tenants/5341/authorize?contentId=movie865343651`
- Path segments: `https://example.com/moviecatalog-license-api/movie865343651/AcquireLicense`

The following is an example of a DRM system configuration with the content ID embedded in the authorization service and license server URLs. As shown, each service may use a different implementation-defined URL structure for carrying the content ID.

```
<ContentProtection
```

Community Review

```
    schemeIdUri="urn:uuid:d0ee2730-09b5-459f-8452-200e52b37567"
    value="AcmeDRM 2.0">
    <cenc:pssh>
        YmFzZTY0IGVuY29kZWQgY29udGVudHMgb2YgkXBzc2iSIGJveCB3aXRoIHRoaXMgU3lzdGVtSUQ=
    </cenc:pssh>
    <dashif:Authzurl>
        https://example.com/tenants/5341/authorize?contentId=movie865343651
    </dashif:Authzurl>
    <dashif:Laurl>
        https://example.com/moviecatalog-license-api/movie865343651/AcquireLicense
</dashif:Laurl>
</ContentProtection>
```

The content ID should not be embedded in DRM system specific data structures such as `pssh` boxes, as logic that depends on DRM system specific data structures is not interoperable and often leads to increased development and maintenance costs.

8 Use of W3C Clear Key with DASH

Clear Key is a DRM system defined by W3C in [7]. It is intended primarily for client and media platform development/test purposes and does not perform the content protection and content key protection duties ordinarily expected from a DRM system.

A DRM system specific **`ContentProtection`** descriptor for Clear Key shall use the `systemID=e2719d58-a985-b3c9-781a-b030af78d30e` and the attribute `@value` shall be equal to `"ClearKey1.0"`.

The **`Laurl`** element shall be used to indicate the license server URL. The attribute `@licenseType` describes the type of the license served by this license server. The value is `"EME-1.0"` when the license is in the format defined in [7] clause 9.1.4.

W3C describes the optional use of `systemID=1077efec-c0b2-4d02-ace3-3c1e52e2fb4b` in [6] clause 4 to indicate that tracks are encrypted with Common Encryption. DASH clients shall not interpret a `pssh` box with `systemID=1077efec-c0b2-4d02-ace3-3c1e52e2fb4b` as an indication that Clear Key is supported on this content.

Note: This common `pssh` box is used in some deployments as a source for creating, on the fly, in the DASH client, `pssh` boxes for other DRM systems. In a lot of cases the `KIDs` are already enough information for creating init data. Even if there is some additional information required, i.e. a contentID or something similar, this is side-loaded. This means that the common `pssh` box is enough to successfully request a DRM license. That, in turn, means that the `MPD` can be kept a little bit smaller and potentially permits the application to add new DRMs without repackaging and without reaching out to some additional service.

An example of a snippet with Clear Key **`ContentProtection`** descriptor using **`Laurl`** is as follows.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:dashif="https://dashif.org/">
    <Period>
        <AdaptationSet>
            <ContentProtection
            schemeIdUri="urn:uuid:e2719d58-a985-b3c9-781a-b030af78d30e" value="ClearKey1.0">
                <dashif:Laurl licenseType="EME-1.0">
                    https://clearKeyServer.foocompany.com
                </dashif:Laurl>
                <dashif:Laurl licenseType="EME-1.0">
                    file://cache/licenseInfo.txt
                </dashif:Laurl>
            </ContentProtection>
        </AdaptationSet>
    </Period>
</MPD>
```

W3C specifies that in order to activate Clear Key, the client must provide Clear Key initialization data to the browser. The Clear Key initialization data consists of a listing of the `KIDs` required to decrypt the content. The **`ContentProtection`** descriptor for Clear Key shall not contain Clear Key

Community review

initialization data. Instead, clients shall construct Clear Key initialization data at runtime, based on the `default KIDs` signaled in the MPD using **`ContentProtection`** descriptors with the `urn:mpeg:dash:mp4protection:2011` scheme.

Note: W3C specifies in [6] clause 3 that the source for the `KIDs` shall be the optional Common `PSSH` box, but because it is optional and not in the MDP, this document prefers using the **`ContentProtection`** descriptors with the `urn:mpeg:dash:mp4protection:2011` scheme.

When requesting a Clear Key license to the license server, it is recommended to use a secure connection as described in clause 10.

When used with a license type equal to "`EME-1.0`":

- The GET request for the license includes in the body the JSON license request format defined in [7] clause 9.1.3. The license request may also include additional authentication elements such as token.
- The response from the license server includes in the body the Clear Key license in the format defined in [7] clause 9.1.4 if the device is entitled to receive the content keys.

Clear Key licenses shall not be used to manage a key and `KID` that is also used by a DRM system. The use of an unprotected key risks the security of DRM systems using that key, and violates the terms of use of most DRM systems.

9 Key rotation

9.1 Introduction

This clause presents different alternatives for implementing key rotation. While key rotation is a tool allowing to update content keys, the aim of the rotation may be different, from a periodic re-authorization of the client to the management of complex use cases with programs of unknown duration.

9.2 Manifest based key rotation signalling

In a live DASH presentation the rights of the user can be different for different programs included in the presentation. This clause describes recommended mechanisms for forcing rights to be re-evaluated at program boundaries, hence creating a periodic re-authorization.

Note: Changing the content keys does not increase the cryptographic security of content protection. The term periodic re-authorization is therefore used here instead of key rotation, to maintain focus on the goal and not the mechanism.

The user's level of access to content is governed by the issuance (or not) of licenses with content keys and the policy configuration carried by the licenses. The authorization server is the authority on what rights are assigned to the user and this is enforced by the license server. To force re-evaluation of rights, a service shall force a new license request to be made. This can be accomplished by changing the content key to one that is not yet available to DASH clients, thereby triggering DRM system activation for the new content key.

Changing the content key is possible on DASH period boundaries as the initialisation segment is updated (and thereofre the `default_KID`), live DASH presentations should create a new period in which content is encrypted with new content keys to force re-evaluation of user's access rights.

9.3 In-band key rotation signalling

An alternative mechanism is to update the `KID` that is in the `moof/traf/sgpd` box in every DASH media segment. This overrides the `default_KID` value that is in the `moov/tenc` box. This allows not generating a new DASH period everytime the content key is changed. Periods make MPD verbose

Community Review

and as key rotation can theoritically happen on every segment in an extreme case, the MPD would then become too big.

For supporting this mechanism:

- The MPD has no `default_KID` attribute in the `ContentProtection` element defined at the `AdaptationSet` level and it has no `ContentProtection` child elements such as `PSSH` elements.
- The initialisation segment has one `PSSH` box per DRM allowing to access content and a `default_KID` value in the tenc box.
- The media segments have one `PSSH` box per DRM allowing to access content and a `KID` value in a `sgpd` box with grouping_type `'seig'`.

Doing so, when the player parses the `sgpd` box, it extracts a `KID` for which the DRM client has no associated content key, it therefore makes a license request for acquiring the new content key.

The following is an example of a snippet of a MPD.

```
<AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1"
subsegmentAlignment="true" subsegmentStartsWithSAP="1" bitstreamSwitching="true">
    <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc"/>
    <ContentProtection schemeIdUri="urn:uuid:edef8ba9-79d6-4ace-a3c8-27dcd51d21ed"/>
    <ContentProtection schemeIdUri="urn:uuid:9a04f079-9840-4286-ab92-e65be0885f95"
        value="MSPR 2.0"/>
    <SegmentTemplate timescale="60000"
        media="index_video_$RepresentationID$_0_$Number$.mp4?m=1618853312"
        initialization="index_video_$RepresentationID$_0_init.mp4?m=1618853312"
        startNumber="12625002" presentationTimeOffset="403937359120">
        <SegmentTimeline>
            <S t="403938079120" d="120000" r="5"/>
            <S t="403938799120" d="124000"/>
        </SegmentTimeline>
    </SegmentTemplate>
    <Representation id="10" width="1920" height="1080" frameRate="30/1" bandwidth="5000000"
        codecs="avc1.4D4028"/>
    <Representation id="7" width="1280" height="720" frameRate="30/1" bandwidth="3000000"
        codecs="avc1.4D401F"/>
    <Representation id="9" width="640" height="360" frameRate="30/1" bandwidth="1499968"
        codecs="avc1.4D401E"/>
</AdaptationSet>
```

Note: As for the mechanism presented in clause 9.2, all players will perform license requests at the same time, hence potentially, creating delays in the response time of the license server.

9.4 In-band key hierarchy

Using a key hierarchy allows a single content key to selectively unlock only a subset of a DASH presentation and apply license policy updates without the need to perform license requests at every program boundary. Unlike the mechanisms presented in the previous clauses, license requests at when content keys change are not necessary.

A key hierarchy defines a multi-level structure of cryptographic keys, instead of a single content key:

- Root keys take the place of content keys in DASH client workflows.
- Leaf keys are used to encrypt the media samples.

Note: A root key might not be an actual cryptographic key. Rather, it acts as a reference to identify the set of leaf keys that protect content. A DASH client requesting a license for a specific root key will be interpreted as requesting a license that makes available all the leaf keys associated with that root key.

Intermediate layers of cryptographic keys may also exist between root keys and leaf keys but such layers are DRM system specific and only processed by the DRM system, being transparent to the DASH client and the media platform. To a DASH client, only the root keys have meaning. To the media platform, only the leaf keys have meaning.

Community Review

This layering enables the user's rights to content to be evaluated in two ways:

1. Changing the root key invokes the full re-evaluation workflow as a new license request shall be made by the DASH client.
2. Changing the leaf key invokes an evaluation of the rights granted by the license for the root key and processing of any additional policy attached to the leaf key. If result of this evaluation indicates the leaf key cannot be used, the DRM system will signal playback failure to the DASH client.

Changing the root key is equivalent to changing the content key in terms of content and MPD signaling, requiring a new period to be started. The leaf key can be changed in any media segment and does not require modification of the MPD. Leaf keys should not be changed within the same program. Changing leaf keys on a regular basis does not increase cryptographic security.

Note: A DASH service with a key hierarchy is sometimes referred to as using "internal key rotation".

The mechanism by which a set of leaf keys is made available based on a request for a root key is DRM system specific. Nevertheless, different DRM systems may be interoperable as long as they can each make available the required set of leaf keys using their system-specific mechanisms, using the same root key as the identifier for the same set of leaf keys.

A key hierarchy is implemented by listing the `default_KID` in the `tenc` box of the initialization segment (identifying the root key) and then overriding the key identifier in the `sgpd` boxes of media segments (identifying the leaf keys that apply to each media segment). The `moof/pssh` box is used to deliver/unlock new leaf keys and may provide the associated license policy. When using a key hierarchy, the leaf keys are typically delivered in-band in the media segments, using `moof/pssh` boxes, together with additional/updated license policy constraints. The exact implementation is DRM system specific and transparent to a DASH client.

As an example, in Figure 3, different rows indicate root key changes. Color alternations indicate leaf key changes. A key hierarchy enables per-program access control even in scenarios where a license request is only performed once per day. The single license request makes available all the leaf keys that the user is authorized to use during the next epoch.

nunity Review

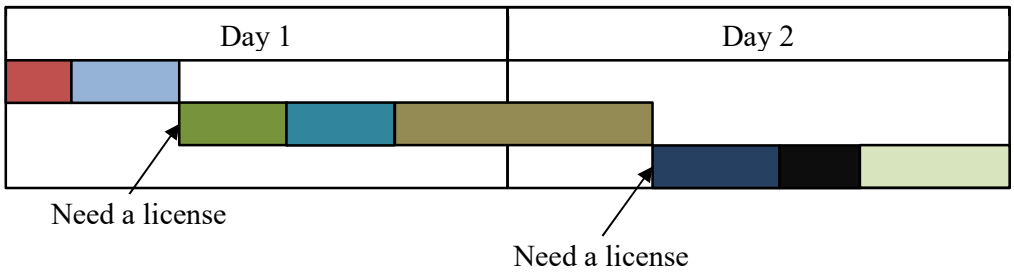


Figure 3: Example of key hierarchy on different programs.

A key hierarchy is useful for broadcast scenarios where license requests are not possible at arbitrary times (e.g. when the system operates by performing nightly license updates). In such a scenario, this mechanism enables user access rights to be cryptographically enforced at program boundaries, defined on the fly by the service provider, while re-evaluating the access rights during moments when license requests are possible. At the same time, it enables the service provider to supply in-band updates to license policy (when supported by the DRM system).

Similar functionality could be implemented without a key hierarchy by using a separate content key for each program and acquiring all relevant licenses in advance. The advantages of a key hierarchy are:

- Greatly reduced license acquisition traffic and required license storage size, as DRM systems are optimized for efficient handling of large numbers of leaf keys.
- Ability for the service provider to adjust license policy at any time, not only during license request processing (if in-band policy updates are supported by the DRM system).

10 HTTPS and DASH

Transport security in HTTP-based delivery may be achieved by using HTTP over TLS (HTTPS) as specified in [13]. HTTPS is a protocol for secure communication which is widely used on the Internet and also increasingly used for content streaming, mainly for protecting:

- The privacy of the exchanged data from eavesdropping by providing encryption of bidirectional communications between a client and a server, and
- The integrity of the exchanged data against forgery and tampering.

As an MPD carries links to media resources, web browsers follow the W3C recommendation [11]. To ensure that HTTPS benefits are maintained once the MPD is delivered, it is recommended that if the MPD is delivered with HTTPS, then the media also be delivered with HTTPS.

DASH also explicitly permits the use of HTTPS as a URI scheme and hence, HTTP over TLS as a transport protocol. When using HTTPS in an MPD, one can for instance specify that all media segments are delivered over HTTPS, by declaring that all the BaseURL's are HTTPS based, as follow:

```
<BaseURL>https://cdn1.example.com/</BaseURL>
<BaseURL>https://cdn2.example.com/</BaseURL>
```

One can also use HTTPS for retrieving other types of data carried with a MPD that are HTTP-URL based, such as, for example, DRM licenses specified within the ContentProtection descriptor:

```
<ContentProtection
    schemeIdUri="urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    value="DRMNAME version"
    <dashif:Laurl>https://MoviesSP.example.com/protect?license=kljklsdfiowek</dashif:Laurl>
</ContentProtection>
```

It is recommended that HTTPS be adopted for delivering DASH content. It should be noted nevertheless, that HTTPS does interfere with proxies that attempt to intercept, cache and/or modify content between the client and the TLS termination point within the CDN. Since the HTTPS traffic is opaque to these intermediate nodes, they can lose much of their intended functionality when faced with HTTPS traffic.

While using HTTPS in DASH provides good protection for data exchanged between DASH servers and clients, HTTPS only protects the transport link, but does not by itself provide an enforcement mechanism for access control and usage policies on the streamed content. HTTPS itself does not imply user authentication and content authorization (or access control). This is especially the case that HTTPS provides no protection to any streamed content cached in a local buffer at a client for playback. HTTPS does not replace a DRM.

11 Enhanced Clear Key Content Protection (ECCP)

11.1 Background

The purpose of Enhanced Clear Key Content Protection (ECCP) is to define a content protection mechanism for DASH content which provides greater protection than Transport Layer Security (TLS) [13] delivery, token authentication or Clear Key used individually. TLS protects content in transit from client to server but the media content is stored and processed in the clear. Token authentication schemes will authenticate a client, but the media content again is processed in the clear. Clear Key protects the content at rest and while it is being processed in an inaccessible

Community Review

pipeline within the client, however it provides no authentication as the key is given to any client that requests it. By combining TLS delivery with token authentication and Clear Key, we can ensure cryptographic enforcement of that token authorization, leading to a class of access control which provides stronger protection than the contributing schemes applied individually.

ECCP is therefore a collective set of restrictions placed on content preparation, manifest preparation, license server behavior and segment authentication. These restrictions are defined below.

11.2 Constraints on DASH content generation

Media segments shall be packaged in CMAF containers per DASH-IF IOP part 1 [4] and part 2 [5]. Additional contraints for encryption are defined in clause 6. Content protection constraints for the MPD are defined in clause 7.

11.3 Constraints on content protection

Implementation of W3C Clear Key content protection shall follow the requirements defined in clause 8.

Note: The support of ECCP is not announced in the MPD with an additional **`ContentProtection`** descriptor with its own system ID. Having a **`ContentProtection`** description with the Clear Key system ID is enough. The additional constraints for ECCP are enforced when interacting with media servers and content key servers.

11.4 Constraints on transport

All URLs referencing manifests, media objects and license servers shall use the scheme “https” as defined in clause 10, with the additional constraint that the TLS version shall be 1.2 or higher.

11.5 Constraints on access control

Access control to the manifest, the license key URL and all media segments containing encrypted content shall be present. No object described in the manifest, including the manifest itself, shall be openly available to an unauthenticated and/or unauthorized client. This access control may invoke both authorization and/or authentication processes and may be implemented through an enforcement scheme such as:

- Tokens. There is no single standard for the token format. Examples are available in [18] and [19].
- Client certificates
- Proxy solutions. A common implementation for transparent authorization is to use a “license proxy” that sits between the client and the real license server. It acts as a license server but instead forwards the license request after authorization checks have passed.

If used, tokens shall be transmitted either as part of the URL path, or as a query argument. This ensures that native implementations will transfer the token information automatically while making the GET or POST request.

11.6 Examples

An example of a compliant manifest for content protected by ECCP is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:dashif="https://dashif.org/CPS" xmlns="urn:mpeg:dash:schema:mpd:2011"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:cenc="urn:mpeg:cenc:2013"
    xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd urn:mpeg:cenc:2013 CENC.xsd
https://dashif.org/CPS DASH-IF-CPS.xsd"
    profiles="urn:mpeg:dash:profile:isoff-live:2011" minBufferTime="PT2S" type="static"
    mediaPresentationDuration="PT32.08333206176758S">
    <Period id="0">
```

Community Review

```
        <AdaptationSet id="0" contentType="video" maxWidth="1920" maxHeight="1080"
            frameRate="12288/512" segmentAlignment="true" par="16:9">
            <ContentProtection value="cbcs" schemeIdUri="urn:mpeg:dash:mp4protection:2011"
                cenc:default_KID="9eb4050d-e44b-4802-932e-27d75083e266"/>
            <ContentProtection value="ClearKey1.0"
                schemeIdUri="urn:uuid:e2719d58-a985-b3c9-781a-b030af78d30e">
                <dashif:Laurl>https://example-license-server.com/license</dashif:Laurl>
            </ContentProtection>
            <Representation id="0" bandwidth="1350152" codecs="avc1.64081f" mimeType="video/mp4"
                sar="1:1" width="768" height="432">
                <SegmentTemplate timescale="12288" initialization="768x432/init.mp4"
                    media="768x432/$Number%04d$.m4s" startNumber="1">
                    <SegmentTimeline>
                        <S t="0" d="49152" r="7"/>
                        <S t="393216" d="1024"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
        <AdaptationSet id="1" contentType="audio" segmentAlignment="true">
            <ContentProtection value="cbcs" schemeIdUri="urn:mpeg:dash:mp4protection:2011"
                cenc:default_KID="9eb4050d-e44b-4802-932e-27d75083e266"/>
            <ContentProtection value="ClearKey1.0"
                schemeIdUri="urn:uuid:e2719d58-a985-b3c9-781a-b030af78d30e">
                <dashif:Laurl>https://example-license-server.com/license</dashif:Laurl>
            </ContentProtection>
            <Representation id="5" bandwidth="131598" codecs="mp4a.40.2" mimeType="audio/mp4"
                audioSamplingRate="48000">
                <AudioChannelConfiguration
                    schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
                <SegmentTemplate timescale="48000" initialization="audio_en/init.m4a"
                    media="audio_en/$Number%04d$.m4a" startNumber="1">
                    <SegmentTimeline>
                        <S t="0" d="192512"/>
                        <S t="192512" d="191488"/>
                        <S t="384000" d="192512"/>
                        <S t="576512" d="191488"/>
                        <S t="768000" d="192512"/>
                        <S t="960512" d="191488"/>
                        <S t="1152000" d="192512"/>
                        <S t="1344512" d="191488"/>
                        <S t="1536000" d="1024"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
</MPD>
```

Community Review

As defined in [7] clause 9.1.3 and clause 9.1.4, upon receiving this manifest, the player would remove the '-' from the the KID, convert the Hex to base64, strip out any '==' padding and then call the License server specified in the MPD `Laurl` with a HTTPS POST-based JSON request which would look like:

{"kids":["nrQFDeRLSAKTLifXUIPiZg"],"type":"temporary"}

The license server would respond with a JSON response similar to:

{"keys":[{"kty":"oct","k":"FmY0xnWCPCNaSpRG-tUuTQ","kid":"nrQFDeRLSAKTLifXUIPiZg"}],"type":"temporary"}

The player would parse the key "k" which it would use to proceed with decryption via the CENC (Common Encryption) component available within the application environment.

Annex A (normative): XML schema for DASH-IF MPD extensions

The namespace for the DASH-IF MPD extensions defined in this document is https://dashif.org/CPS. This document refers to this namespace using the dashif prefix. The schema of the DASH-IF MPD extensions is provided on the dashif.org site. It is reproduced below for convenience, the electronic file is the authoritative source.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:dashif="https://dashif.org/CPS"
targetNamespace="https://dashif.org/CPS" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Laurl" type="dashif:LaurlType"/>
    <xs:element name="Authzurl" type="dashif:AuthzurlType"/>
    <xs:complexType name="LaurlType">
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute name="licenseType" type="xs:string" use="optional"/>
                <xs:anyAttribute namespace="##other" processContents="strict"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="AuthzurlType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:anyAttribute namespace="##other" processContents="strict"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

Community Review

Annex B (informative): Change History

Date	Version	Information about changes
<Month year>	<#>	<Changes made are listed in this cell>

Community Review

History

Document history		
<Version>	<Date>	<Milestone>

Latest changes made on 2019-01-29

Community Review