INTERNATIONAL STANDARD

ISO/IEC 23009-4

First edition
2013-07-01

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 4: Segment encryption and authentication

Technologies de l'information — Diffusion en flux adaptatif dynamique sur HTTP (DASH) —

Partie 4: Cryptage et authentification des segments

Reference number
ISO/IEC 23009-4:2013(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 23009-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23009 consists of the following parts, under the general title *Information technology — Dynamic adaptive streaming over HTTP (DASH)*:

— *Part 1: Media presentation description and segment formats*

— *Part 2: Conformance and reference software*[1]

— *Part 3:* [Technical Report][2]

— *Part 4: Segment encryption and authentication*

[1] To be published.

[2] To be published.

Introduction

Dynamic Adaptive Streaming over HTTP (DASH) enables media-streaming model for delivery of media content in which control lies exclusively with the client. Clients may request data using the HTTP protocol from standard web servers that have no DASH-specific capabilities. Consequently, ISO/IEC 23009 focuses not on client or server procedures but on the data formats used to provide a DASH Media Presentation.

This part of ISO/IEC 23009 provides methods and interfaces for segment encryption and verification of segment integrity and authenticity

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 4:
Segment encryption and authentication

1 Scope

This part of ISO/IEC 23009 specifies:

- format-independent segment encryption and signalling mechanisms for use with any media segment format used in DASH (ISO/IEC 23009-1:2012);
- mechanisms to ensure segment integrity and authenticity for use with any segment used in DASH (ISO/IEC 23009-1:2012).

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23009-1:2012, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

Advanced Encryption Standard, Federal Information Processing Standards Publication 197, FIPS-197, http://www.nist.gov/

Secure Hash Standard, Federal Information Processing Standards Publication 180, FIPS 180-3, http://www.nist.gov/

Recommendation of Block Cipher Modes of Operation, NIST, NIST Special Publication 800-38A, http://www.nist.gov/

Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST, NIST Special Publication 800-38D, http://www.nist.gov/

IETF RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*, H. Krawczyk, M. Bellare, R. Canetti, February 1997

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol*, T. Dierks et al, August 2008

IETF RFC 5652/STD 70, *Cryptographic Message Syntax (CMS)*, R. Housley, September 2009

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1.1
additional authenticated data
input data to the authenticated encryption function that is authenticated but not encrypted

3.1.2
authentication tag
cryptographic checksum on data that is designed to reveal both accidental errors and the intentional modification of the data

3.1.3
authenticated encryption
mode of operation in which the plaintext is encrypted into the ciphertext, and an authentication tag is generated on the AAD and the ciphertext

3.1.4
cryptoperiod
number of continuous segments for which the same encryption key and the same initialization vector are used

3.1.5
encryption system
system used for encryption of Media Segments using keys provided by the Key System

3.1.6
key system
system that provides keys necessary for decryption of Media Segments

3.1.7
segment number
unique positive integer associated with a Media Segment within a Representation

Note 1 to entry: The Media Segment presented (in presentation order) after Media Segment with Segment Number N has Segment Number N+1.

3.2 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

AAD	Additional Authentication Data
AES	Advanced Encryption Standard as specified in FIPS-197
AES-CBC	AES cipher in Cipher Block Chaining mode, as specified in NIST 800-38A
ECB	Electronic Code Book, as specified in NIST 800-38A
AES-GCM	AES cipher in Galois/Counter Mode, as specified in NIST 800-38D
HMAC	Hash-based Message Authentication Code, as specified in IETF RFC 2104
IV	Initialization Vector
MPD	Media Presentation Description, as specified in ISO/IEC 23009-1:2012

SHA	Secure Hash Algorithm, as specified in FIPS 180-3
SN	Segment Number
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name

3.3 Notation

Media Segment with Segment Number *i*: *S(i)*

Cryptoperiod starting with Segment Number *i* and having *d* Media Segments: *CP(i,d)*

Key and initialization vector in use during *CP(i,d)*: $K_{CP(i,d)}$, $IV_{CP(i,d)}$

4 Introduction

4.1 Segment Encryption

The content protection framework provided in this part of ISO/IEC 23009 is a framework for out-of-band derivation of parameters needed for successful decryption of media segments. The tools provided are MPD interfaces that allow derivation of key and initialization parameters, baseline encryption and key resolution methods, and, lastly, it provides extensibility points to accommodate different key resolution and encryption algorithms using the same interface.

Conceptually, the content protection framework provided in this part of the standard can be viewed as two entities, key system and encryption system. Key system derives keys associated with a segment given the information provided in the MPD, while the encryption system decrypts media segments given the information provided in the MPD and encryption keys provided by the key system.

The baseline mandatory system applies AES-CBC encryption to a complete segment and uses HTTP(S) for key transport. In this baseline system the DASH client will be able to recognize uniquely for each segment which key and initialization vector were used for their encryption. The client will then issue a GET request for the key, and will either issue a GET request for the initialization vector or derive it locally. After receiving key and initialization vector, the DASH client can successfully decrypt the media segment and pass it to the media engine. In this description, AES-CBC full-segment encryption is the encryption system, and key retrieval using HTTP(S) is the key system.

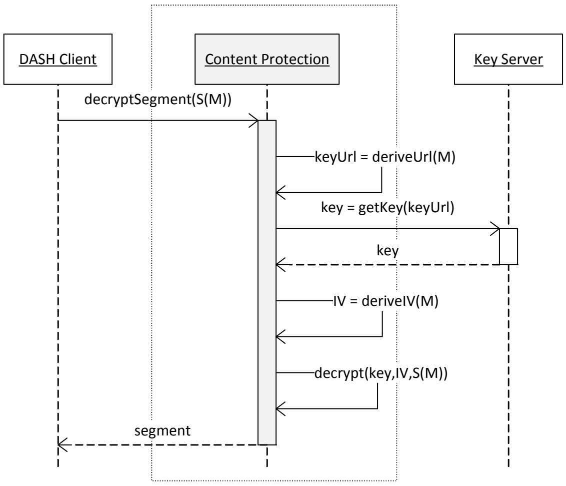

Figure 1 — Baseline Segment Encryption

As most DRM systems employ license-based systems to derive keys, license-based key systems are supported in this standard. In this case, a license is retrieved, and the key URIs are opaque key identifiers. The license-based key system will resolve these ID's into keys in an unspecified way, and pass the keys to an encryption system. The latter, having keys provided by the key system and the encryption information (e.g. algorithm specification and IV) provided by the MPD, decrypts the media segment.

Additional encryption methods can be signalled using URIs and (possibly) generic encryption-related parameters provided in this part of ISO/IEC 23009This part of ISO/IEC 23009 is format-independent: it does not apply specifically to any type of media segment, and its notion of cryptoperiods is completely divorced from any specific segment type. The baseline encryption system applies to a complete segment.

The normative part of this framework provides (a) the MPD interface, and (b) baseline key and encryption systems. These are shown in Figure1 — Baseline Segment Encryption. Note that the implementation shown in this figure is for illustration purposes, and many of the operations can be optimized e.g. by parallelization and pre-fetching.

The Segment Encryption scheme specifies standard encryption and key mapping methods that may be used when segment protection is needed. The scheme operates by applying encryption to segments, which are thus transmitted in a protected fashion. Definitions are provided to identify the segments as encrypted, and to identify the appropriate key(s) and IV(s) from a MPD.

4.2 Segment Authentication

The Segment Authentication framework is a framework allowing use of authenticity tags for all DASH segment types in order to verify the origin and content authenticity. This framework works by calculating a digest or a MAC of an unencrypted segment, and storing the value externally. The MPD interface provides URL templates to retrieve these, using HTTPS or HTTP. The client retrieves the digest/signature, then calculates them locally on the decrypted (sub)segment, and can reject the (sub)segment in case of a mismatch.

If used together with encryption, the mode of operation of this framework is "authenticate, then encrypt", rather than the more common "encrypt, then authenticate" mode. The former provides an important feature of encryption invariance: if no encryption, or different encryption algorithm or/and parameters were used for encryption of the same media segment for serving it to different clients, the authenticity tag will still stay the same as long as the content itself has not changed.

Segment Authentication is independent of any content protection scheme, and may be used on unencrypted segment, as well as on encrypted segments encrypted using any DRM system. Note that this implies that use of the Content Protection framework of this part of ISO/IEC 23009 is not required in order to use the Content Authentication framework.

The normative part of this framework provides (a) the MPD interface, and (b) a baseline authenticity algorithm.

4.3 MPD security

The frameworks provided in this part of ISO/IEC 23009 are as secure as the MPD is. Therefore it is extremely important to protect the MPD, e.g. by transmitting it over a secure connection, by verifying its integrity and authenticity.

Methods of MPD protection are out of scope of this standard.

5 Signalling encryption and authentication

5.1 Encryption declaration

5.1.1 ContentProtection element

5.1.1.1 Definition

The application of the encryption format defined in this part of the standard to segments shall be declared using the URN `urn:mpeg:dash:sea:enc:2013` as the value for `@schemeIdUri` in a **`ContentProtection`** descriptor applicable to the encrypted segments. The **`ContentProtection`** descriptor may contain zero or more **`CryptoPeriod`** and/or **`CryptoTimeline`** elements.

Note that ContentProtection descriptor is defined in the namespace `urn:mpeg:dash:schema:mpd:2011` defined in ISO/IEC 23009-1, while **`SegmentEncryption`**, **`CryptoPeriod`** and **`CryptoTimeline`** are defined in the namespace `urn:mpeg:dash:schema:sea:2013`, defined in Annex A. For illustration purposes, all elements of the above scheme are prefixed with `sea:` in the syntax below.

5.1.1.2 Semantics

Table 1 — Use of DASH Content Protection descriptor

Element or Attribute Name	Use	Description
`ContentProtection`		
`@schemeIdUri`	M	Shall be `urn:mpeg:dash:sea:enc:2013` for this part of the standard.
`sea:SegmentEncryption`	1	Specifies the encryption system used and its global properties. See 5.1.2 for the definition.
`sea:License`	0..N	Specifies the key system used and ways of getting license, if needed.
`sea:CryptoPeriod`	0..N	Specifies information needed for derivation of key and IV information for a single cryptoperiod. See 5.1.4.

Element or Attribute Name	Use	Description
sea:CryptoTimeline	0..N	Specifies information needed for derivation of key and IV information for several constant-length cryptoperiods. See 5.1.6.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are bold; attributes are non-bold and preceded with an @.		

5.1.2 SegmentEncryption element

The **SegmentEncryption** element describes the global properties of segment encryption as used in all cryptoperiods.

A single **SegmentEncryption** element shall always be present within the **ContentProtection** descriptor if the value of **ContentProtection**@schemeIdUri is urn:mpeg:dash:sea:enc:2013.

Table 2 — Semantics of SegmentEncryption element

Element or Attribute Name	Use	Description
SegmentEncryption		Specifies the properties of an encryption system
@schemeIdUri	M	Specifies the encryption system used for segment encryption. Possible encryption systems are specified in 6.3.2.
@keyLength	OD	Specifies the length (in bits) of a key used in cipher defined in @schemeIdUri. Default value is 128.
@ivLength	OD	Specifies the length (in bits) of an initialization vector used in cipher defined in @schemeIdUri. Default value is 128.
@authTagLength	OD	Specifies the length (in bits) of an authentication tag used, if authenticated encryption block operation mode (e.g. GCM) is used. Default value is 0 (i.e., no authentication is available).
@earlyAvailability	OD	Distance in seconds between the time a key and an IV can be resolved using the provided URIs and the availability time of the first segment encrypted using these keys. Default value is 1.0 seconds.
@ivEncryptionFlag	OD	When set to 'true', and Segment Number is used for IV derivation (as defined in 6.4.4), ECM IV encryption will be used. Default value is 'false'.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are bold; attributes are non-bold and preceded with an @.		

5.1.3 License element

The **License** element describes the global properties of a key system used in all cryptoperiods.

One or more License element shall be present within the **ContentProtection** descriptor if the value of **ContentProtection**@schemeIdUri is urn:mpeg:dash:sea:2013, and license-based key systems are used. If absent, URL's provided in the **CryptoPeriod** and **CryptoTimeline** elements shall be sufficient to retrieve the keys.

Table 3 — Semantics of License element

Element or Attribute Name	Use	Description
License		Specifies information needed to retrieve keys
@keySystemUri	M	Specifies the URN of the key system.
@keyLicenseUrlTemplate	O	Specifies template HTTP(S) URL used to retrieve license used by the key system to derive the encryption keys, using same syntax and variable substitution as defined in ISO/IEC 23009-1:2012, 5.3.9.4.4. Derivation of this template is specified in 6.2.3
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are bold; attributes are non-bold and preceded with an @.		

5.1.4 Common cryptoperiod properties

5.1.4.1 Definition

Cryptoperiods are characterized by the common encryption parameters, and duration. Hence, a separate element represents the archetypal cryptoperiod. Both **CryptoPeriod** and **CryptoTimeline** elements are built upon this foundation, the former representing a single cryptoperiod, the latter – multiple similar cryptoperiods.

5.1.4.2 Semantics

Table 4 — Common properties of a cryptoperiod

Element or Attribute Name	Use	Description
CryptoPeriodType		Specifies properties common to all cryptoperiods
@numSegments	O	Specifies the number of segments in a cryptoperiod. In case of CryptoTimeline, this is the number of segments in each cryptoperiod of this **CryptoTimeline**. @numSegments may be absent only if this is the last **CryptoPeriod** element of the Period. In this case, the cryptoperiod continues till the end of this Period. Note that @numSegments shall not be absent for any **CryptoTimeline** element.

Element or Attribute Name	Use	Description
@keyUriTemplate	M	Specifies the template for key URI generation, using same syntax and variable substitution as defined in ISO/IEC 23009-1:2012, 5.3.9.4.4. @keyUriTemplate is used once each cryptoperiod, thus for a cryptoperiod *CP(i,d)*, the @keyUriTemplate URI will be constructed with $Number$ = *i*. Same applies for $Time$: the value used is the $Time$ value of segment *S(i)* will be used. Note that the use of @keyUriTemplate does not imply use of @ivUrlTemplate or SegmentTemplate. Key derivation rules are specified in 6.4.3.
@ivUriTemplate	O	Specifies the template for IV URI generation using same syntax and variable substitution as defined in ISO/IEC 23009-1:2012, 5.3.9.4.4. @ivUriTemplate is used once each cryptoperiod, thus for a cryptoperiod *CP(i,d)*, the @ivUrlTemplate URI will be constructed with $Number$ = *i*. Same applies for $Time$: the value used is the $Time$ value of segment *S(i)* will be used. Use of @ivUrlTemplate does not imply use of either @keyUriTemplate or **SegmentTemplate**. For IV format definition see 6.4.4.2.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are bold; attributes are non-bold and preceded with an @.		

5.1.5 CryptoPeriod element

5.1.5.1 Definition

The **CryptoPeriod** element defines a single cryptoperiod – namely, it provides information allowing derivation of an encryption key and an initialization vector, as well as identifying segments which were encrypted using the former two elements. A **CryptoPeriod** element corresponds uniquely to a start segment. It may have explicitly specified duration (i.e., number of segments), or be unbounded (i.e., continue till the end of the current Period.

Segments are identified by the Segment Number, as defined in ISO/IEC 23009-1:2012 5.3.9.4.4.

An example of an MPD containing a **CryptoPeriod** element is given in C.1.

5.1.5.2 Semantics

Table 5 — Semantics of the `CryptoPeriod` element

Element or Attribute Name	Use	Description
`CryptoPeriod`		Specifies information and URIs needed for derivation of key information for a single cryptoperiod.
`@startOffset`	OD	Specifies the number of unencrypted segments after the end of the previous cryptoperiod and the first Media Segment to which the key/IV information applies. Default value is 0. Derivation rules specified in 6.4.2 apply
`@IV`	O	Specifies the initialization vector. It shall not be present if `@ivUriTemplate` is present. IV derivation rules are specified in 6.4.4.
`@aad`	O	Specifies the additional authentication data. AAD derivation rules are specified in 6.4.4.
`CryptoPeriodType`	-	specifies the common attributes and elements (attributes and elements from base type ***CryptoPeriodType***). For details see 5.1.4
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are `bold`; attributes are non-bold and preceded with an @.		

5.1.6 CryptoTimeline element

5.1.6.1 Definition

The **`CryptoTimeline`** element is used for derivation of multiple cryptoperiods of constant length. While a single **`CryptoPeriod`** corresponds to a single cryptoperiod, a single **`CryptoTimeline`** element corresponds to multiple cryptoperiods.

Use of the **`CryptoTimeline`** is encouraged when a highly regular pattern of cryptoperiods is used, e.g. when a key/IV pair is changed every 4 cryptoperiods. Each cryptoperiod generated from a **`CryptoTimeline`** contains the same number of segments (see example in C.2 below)

5.1.6.2 Semantics

Table 6 — Semantics of **CryptoTimeline** element

Element or Attribute Name	Use	Description
CryptoTimeline		Specifies a sequence of cryptoperiods, each containing same amount of segments
@numCryptoPeriods	O	Specifies number of constant-duration cryptoperiods within this timeline. If absent, the last cryptoperiod ends with the end of the Period this **ContentProtection** descriptor belongs to. Note that this implies that the amount of segments in the last cryptoperiod in this case can be smaller than specified in the @numSegments attribute.
@firstStartOffset	OD	Specifies the number of unencrypted segments between the end of the last cryptoperiod and the first segment of the first cryptoperiod in this **CryptoTimeline**. Default value is 0. Derivation rules specified in 6.4.2 apply.
@ivBase	OD	Specifies the IV base value for this cryptoperiod. When @ivBase is present, IV is a sum of @ivBase and Segment number, as specified in 6.4.4.2. If absent, the default value is 0. Shall not be present if @ivUriTemplate is present.
@aadBase	OD	Specifies the AAD base value for this cryptoperiod. AAD is the sum of @aadBase and the Segment Number. If absent, the default value is 0.
CryptoPeriodType	-	specifies the common attributes and elements (attributes and elements from base type ***CryptoPeriodType***). For details see 5.1.4
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are bold; attributes are non-bold and preceded with an @.		

Note: typically in a key rotation scenario @firstStartOffset and @numCryptoPeriods will not be specified, and the key/IV pair will change every @numSegments segments.

5.2 Authentication declaration

5.2.1 General

The **ContentAuthenticity** element, defined below, shall be used in either **EssentialProperty** or **SupplementalProperty** defined in ISO/IEC 23009-1:2012, depending on the application requirements.

The value of @schemeIdUri in either **EssentialProperty** or **SupplementalProperty** shall be urn:mpeg:dash:sea:auth:2013 if authentication framework is used.

Multiple content authenticity verification schemes can be defined. Two schemes, SHA-256 digest, identified by the URN urn:mpeg:dash:sea:sha256:2013 and HMAC-SHA1 MAC, identified by urn:mpeg:dash:sea:hmac-sha1 are specified in this part of the standard.

5.2.2 ContentAuthenticity element

5.2.2.1 Definition

The **ContentAuthenticity** element provides a URL for key acquisition and a template for constructing a URL, which is further used for downloading the authenticity tag for a given (sub)segment. URL construction rules are defined in 5.2.3.

5.2.2.2 Semantics

Table 7 — Semantics of ContentAuthenticity element

Element or Attribute Name	Use	Description
ContentAuthenticity		Specifies information necessary to compute an authenticity tag for segment
@authSchemeIdUri	M	Specifies the algorithm used for computing the authenticity tag
@authUrlTemplate	M	Specifies the template for creating the URL used for retrieving the authenticity tag value. The rules for URL creation are specified in 5.2.3.
@authTagLength	O	Specifies the length of an authentication tag in bits. If absent, the tag length is same as in the algorithm identified by @authSchemeIdUri
@keyUrlTemplate	O	Specifies the template for key URI generation, using syntax and variable substitution as defined in ISO/IEC 23009-1:2012, 5.3.9.4.4.
Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: <minOccurs>...<maxOccurs> (N=unbounded) Elements are bold; attributes are non-bold and proceeded with an @.		

5.2.3 URL derivation

The authenticity tag URLs shall be constructed with the following mechanism:

1. A complete URL for a given media, initialization, index, or bitstream switching segment, or for a subsegment, is constructed.

2. The same substitution variables as in ISO/IEC 23009-1:2012 Annex E shall be used for constructing the digest or signature URL templates. If the request does not contain a byte range, the value of $first$ shall be "0", and the value of $last$ shall be "Inf".

The following restrictions are imposed on byte range requests:

1. Authenticity tags shall not be requested for byte range requests that do not correspond to segments or subsegments.

2. If subsegments are used, a separate authenticity tag per each subsegment can be retrieved using the byte range syntax above.

6 Segment encryption

6.1 Segment Format

Encrypted segments may not conform to any Media Segment format defined in ISO/IEC 23009-1:2012. All Media Segment format definitions and requirements of ISO/IEC 23009-1:2012 apply to unencrypted Media Segments, according to their MIME type as specified in the appropriate `@mimeType` parameter specified in the MPD.

6.2 Key systems

6.2.1 General

Keys and initialization vectors URIs are resolved using key systems identified by **`License`**`@keySystemUri`. If the **`License`** element is not present, all URL's in **`CryptoPeriod`** and **`CryptoTimeline`** elements shall be HTTP(S) URL's, and successful GET request with these URL's shall return keys and initialization vectors in a binary format specified in 6.4

6.2.2 License-based Key Systems

Proprietary key systems may be used to resolve arbitrary URN's. Some of these may need additional license information. Key systems requiring license information shall use **`License`**`@keyLicenseUrlTemplate` to retrieve the license. The license format is system-specific and is not defined in this part of the standard.

There can be different licenses corresponding to different key URI's, thus **`License`**`@keyLicenseUrlTemplate` may yield different results for different cryptoperiods. The substitution variables `$Number$` and `$Time$` used in this template are same as the ones in the corresponding cryptoperiod.

Additional substitution variable, `$KeyUri$`, can be used in this template. This variable has a format of a URI. Its value is defined as the value of key URI for a given cryptoperiod (i.e., `@keyUriTemplate` after expansion of the substitution variables.

Note: the definitions above imply that for each cryptoperiod `@keyUriTemplate` is derived first, and the derivation of `@keyLicenseUrlTemplate` is done after it.

6.3 Encryption systems

6.3.1 General

Media segments shall be encrypted using the encryption system identified by the **`SegmentEncryption`**`@schemeIdUri` attribute.

Initialization, Index, and Bitstream Switching segments shall not be encrypted.

Any concatenation involving encrypted segments shall apply after decryption.

Implementation of the appropriate encryption system is essential; hence a client that does not implement the algorithm specified in **`SegmentEncryption`**`@schemeIdUri` should not attempt to present any encrypted media segment.

A client shall implement the AES-128 CBC encryption scheme specified in 6.3.2 below.

6.3.2 AES-128 CBC Encryption System

The AES-128 CBC full-segment encryption system is identified by the URN `urn:mpeg:dash:sea:aes128-cbc:2013`. Support of this scheme is mandatory for clients implementing this part of ISO/IEC 23009.

In this algorithm, AES cipher with 128-bit keys used in CBC mode. Encryption shall be applied to complete segments. Segments shall be padded following the PKCS7 specification to be a multiple of 16 bytes, as described in RFC 5652. Segments start at the beginning of a 16-byte block. This means that if encrypted media segments are accessed through byte ranges, the segment boundaries shall be on 16-byte boundaries.

Cipher Block Chaining occurs only within a segment; at the beginning of each segment, encryption re-starts using the applicable key and initialization vector.

6.3.3 AES-128 GCM Encryption System

The AES-128 GCM full-segment encryption system is identified by the URN `urn:mpeg:dash:sea:aes128-gcm:2013`. Support for this scheme is optional for clients implementing this part of ISO/IEC 23009.

In this algorithm, AES cipher is used in GCM mode with 96-bit initialization vectors and 128-bit authentication tags. Encryption shall be applied to complete segments.

A single combination of key and initialization vector shall be used only once during the whole Period. As a consequence, a cryptoperiod in this encryption system shall only consist of a single segment, and there shall be no identical key/IV combinations within the Period.

Authentication tag is appended to the last byte of the segment (i.e., encrypted segment is `@authTagLength` bytes longer than the unencrypted one).

6.4 Cryptoperiods

6.4.1 General

Each Media Segment is associated with zero or one cryptoperiod; segments that have no cryptoperiod associated with them shall not be encrypted. In a cryptoperiod, segments are encrypted with the same key/IV pair. The properties of a cryptoperiod are a key, an initialization vector, first segment number, and last segment number.

Note: cryptoperiod duration is measured in segments, not time units. Thus, thus there is no requirement for the *segments* to have constant duration.

6.4.2 Assigning segments to cryptoperiods

A single **`CryptoPeriod`** element corresponds to a single cryptoperiod containing `@numSegments` segments with and starting `@startOffset` segments from the end of the previous cryptoperiod. If this cryptoperiod is the first during this Period, `@startOffset` is relative to the start of the Period. A **`CryptoPeriod`** element with `@numSegments` = *D* and first Segment Number *M* corresponds to a cryptoperiod *CP(M,D)*.

For cryptoperiod *CP(M,D)*, segments *S(M), S(M+1), S(M+1), ... S(M+D-1)* are encrypted with the same key / IV combination, $K_{CP(M,D)}$ and $IV_{CP(M,D)}$.

If these are not signalled explicitly, the key and IV derivation rules below apply.

A single **CryptoTimeline** is used for derivation of **CryptoTimeline**@numCryptoPeriods cryptoperiods, each containing **CryptoTimeline**@numSegments segments. The first cryptoperiod in a **CryptoTimeline** is @firstStartOffset segments after the end of the previous cryptoperiod. If this first cryptoperiod is the first during this Period, @firstStartOffset is relative to the start of the Period.

For a **CryptoTimeline** element, with first Segment Number *M*, @numCryptoPeriods = *N*, and @numSegments = *D*, for $0 \le k \le N$, the k^{th} cryptoperiod generated using this **CryptoTimeline** element is $CP(M + k\times D, D)$.

If the **CryptoPeriod** or **CryptoTimeline** are the last element in this Period, and cryptoperiod duration is not explicitly stated by **CryptoTimeline**@numSegments or **CryptoPeriod**@numSegments, it is assumed that the current cryptoperiod continues till the end of the Period. Note that in case of **CryptoTimeline** this implies that there is only one cryptoperiod within such a **CryptoTimeline**.

If neither **CryptoPeriod** nor **CryptoTimeline** are present, all segments shall be unencrypted.

Any segments that are not associated with a cryptoperiod using the rules in this subclause shall be unencrypted.

6.4.3 Key derivation

6.4.3.1 General

A key URI is used in order to retrieve a key resource. There must be one URI associated with a given cryptoperiod.

A URI identifying the location of a key shall be derived once for each cryptoperiod.

The key and initialization vector shall both be available at least @earlyAvailability seconds before the start of the availability window of the first segment of the cryptoperiod and till the end of the availability window of the last segment. This implies that in case of live broadcast the key and initialization vector combination is guaranteed to be available at least @earlyAvailability seconds ahead of time.

Key URI is constructed from the @keyUriTemplate attribute using the using syntax and variable substitution as defined in ISO/IEC 23009-1:2012, 5.3.9.4.4. Use of substitution variables is not required in a template, hence simple URIs can be specified in @keyUriTemplate.

Note: when template variable substitution is used to construct the key URI for cryptoperiod *CP(i,d),* the value of $Number$ is *i*, and the value of $Time$ is the value of **SegmentTimeline**@numSegments corresponding to *S(i).*

HTTPS, rather than HTTP should be used in key URI's. Use of HTTP for this purpose is strongly discouraged.

6.4.3.2 Key format

A key is always in binary format, i.e. a key is represented by sequence of bytes with length given by **SegmentEncryption**@keyLength.

If a key URI is an HTTP(S) URL, the content of the message body of the HTTP response shall only contain **SegmentEncryption**@keyLength bytes and have MIME type application/octet-stream.

6.4.4 IV derivation

6.4.4.1 General

The IV value of a cryptoperiod defined by a **`CryptoPeriod`** element shall be derived using the following mechanism:

1. If **`CryptoPeriod`**`@IV` is present, its value is the IV, in the format defined in 6.4.4.3.
2. If **`CryptoPeriod`**`@ivUriTemplate` is present, this URI is used to derive the IV.
3. If neither **`CryptoPeriod`**`@IV` nor **`CryptoPeriod`**`@ivUriTemplate`, implementations shall derive the IV from the Segment Number, as specified in 6.4.4.2.

The IV value of a cryptoperiod derived from a **`CryptoTimeline`** shall be derived as follows:

1. If **`CryptoTimeline`**`@ivUriTemplate` is used, this URI is used to derive the IV.
2. Otherwise, implementations shall derive the IV from the Segment Number, as specified 6.4.4.2.

6.4.4.2 IV derivation from Segment Number

If `SegmentEncryption@ivEncryptionFlag` value is 'false', and **`CryptoPeriod`** element is used, Segment Number shall be used as the IV value, i.e. $IV_{CP(M,D)}=SN$.

If **`SegmentEncryption`**`@ivEncryptionFlag` value is 'false', and **`CryptoTimeline`** element is used a sum of Segment Number and `@ivBase` shall be used as the IV value, i.e. $IV_{CP(M,D)}=SN + ivBase$. Note that the default value of `@ivBase` is 0, hence if `@ivBase` is absent, $IV_{CP(M,D)}=SN$.

If **`SegmentEncryption`**`@ivEncryptionFlag` value is 'true', ECB-encrypted IV's will be used. This method is described in Appendix C of NIST 800-38A, and its application to this part of ISO/IEC 23009 is defined below.

If **`SegmentEncryption`**`@ivEncryptionFlag` value is 'true', and **`CryptoPeriod`** element is used, IV shall be be an ECB-encrypted value of Segment Number. For example, when AES-128 encryption is used (in any mode), $IV_{CP(M,D)}= AES(SN, K_{CP(M,D)})$.

If **`SegmentEncryption`**`@ivEncryptionFlag` value is 'true', and **`CryptoTimeline`** element is used, the IV is the ECB-encrypted sum of Segment Number and `@ivBase`. For example, when AES-128 encryption is used (in any mode), $IV_{CP(M,D)}= AES(SN + ivBase, K_{CP(M,D)})$.

If **`SegmentEncryption`**`@ivEncryptionFlag` value is 'true', and **`SegmentEncryption`**`@ivLength` is smaller than the output block size of the ECB output (e.g. when 96-bit IV's are used), then the first **`SegmentEncryption`**`@ivLength` most-significant bits from ECB output shall be used as an initialization vector.

6.4.4.3 IV format

The IV is a number in a hexadecimal format. The big-endian binary representation of this number shall be placed in a buffer of **`SegmentEncryption`**`@ivLength` bytes and padded (on the left) with zeros (i.e., bytes with hexadecimal value 0x00). If the IV is derived from the Segment Number is used as an IV, it shall be placed in such a buffer and padded (on the left) with zeros.

When `@ivUriTemplate` is used, the content of the HTTP response to HTTP GET with the IV URL shall only contain **`SegmentEncryption`**`@ivLength` bytes and have MIME type `application/octet-stream`.

6.4.5 AAD derivation

For **CryptoPeriod** element, the AAD is given by the value of **CryptoPeriod**@aad.

For a **CryptoTimeline** element, Segment Number and @aadBase are used for AAD derivation, i.e. $AAD_{CP(M,D)}$=*SN* + *aadBase*.

6.5 Adding new encryption and key systems

This part of the standard defines (a) signalling cryptoperiod properties, (b) signalling encryption and key system properties, and (c) mandatory encryption and key systems. Mandatory encryption and key systems, defined, respectively, in 6.3.2 and 6.2.2 provide a baseline. This baseline is guaranteed to be interoperable.

User-defined and, thus, optional encryption systems and/or key systems can be added using different values of URIs in @encryptionSystemUrn and @keySystemUrn. Example of such system is provided in C.4. A client that does not implement these can distinguish between encrypted and non-encrypted segments, but will be unable to present the encrypted media segments.

Two optional extension mechanisms provided by this part of the specification are licenses and XML extensibility. @keyLicenseUrlTemplate can be used to retrieve information necessary to instantiate the key system, while use of elements from different namespaces in the **SegmentEncryption**, **CryptoPeriod**, and **CryptoTimeline** elements allows addition of user-defined information.

7 Segment authentication

7.1 General

Authenticity tag URLs are provided via the MPD, using the **ContentAuthenticity** element. Authenticity tags may be provided for Media (sub)segments, as well as for Initialization, Index, and Bitstream Switching segments.

If content protection is used, authenticity tags shall be calculated on the unencrypted segment. Segment authentication is optional if used with **SupplementaryProperty** descriptor, and mandatory if used with **EssentialProperty** descriptor.

If the HTTP response to the authentication tag request returns an error, the client may still continue the presentation as usual, as long as the (sub)segments themselves are successfully retrieved.

7.2 Algorithms

7.2.1 SHA-256

The SHA-256 digest algorithm is defined in FIPS 180-3. Its use is indicated by the **ContentAuthenticity**@authSchemeIdUri value of urn:mpeg:dash:sea:sha256:2013. The digest format is a big-endian number in a hexadecimal format.

Note: IETF RFC 6234 provides a reference implementation of SHA-256.

7.2.2 HMAC-SHA1

The HMAC-SHA1 message authentication algorithm is defined in IETF RFC 2104. Its use is indicated by the @schemeIdUri value of urn:mpeg:dash:sea:hmac-sha1:2013. The signature format is a big-endian number in a hexadecimal format.

Annex A
(normative)

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:mpeg:dash:schema:sea:2013"
    attributeFormDefault="unqualified"
    elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:mpeg:dash:schema:sea:2013" xmlns:dash="urn:mpeg:dash:schema:mpd:2011">

    <!-- Global encryption properties -->

    <xs:complexType name="SegmentEncryption">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>

        <xs:attribute name="encryptionSystemUrn" type="xs:anyURI" use="required"/>
        <xs:attribute name="keyLength" type="xs:unsignedInt" default="128"/>
        <xs:attribute name="ivLength" type="xs:unsignedInt" default="128"/>
        <xs:attribute name="authTagLength" type="xs:unsignedInt" default="0"/>
        <xs:attribute name="earlyAvailability" type="xs:double" default="1.0"/>
        <xs:attribute name="ivEncryptionFlag" type="xs:boolean" default="false"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>

    </xs:complexType>

    <xs:complexType name="License">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="keySystemUrn" type="xs:anyURI" use="required"/>
        <xs:attribute name="keyLicenseUrlTemplate" type="xs:anyURI"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <!-- Cryptoperiod signaling -->

    <xs:complexType name="CryptoPeriodType">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="numSegments" type="xs:unsignedLong" default="1"/>
        <xs:attribute name="keyUriTemplate" type="xs:anyURI" use="required"/>
        <xs:attribute name="ivUriTemplate" type="xs:anyURI"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="CryptoPeriod">
        <xs:complexContent>
            <xs:extension base="CryptoPeriodType">
                <xs:attribute name="startOffset" type="xs:unsignedLong" default="0"/>
                <xs:attribute name="IV" type="xs:hexBinary"/>
                <xs:attribute name="aad" type="xs:hexBinary"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="CryptoTimeline">
        <xs:complexContent>
            <xs:extension base="CryptoPeriodType">
                <xs:attribute name="firstStartOffset" type="xs:unsignedLong" default="0"/>
                <xs:attribute name="numCryptoPeriods" type="xs:unsignedLong"
                              use="required"/>
                <xs:attribute name="ivBase" type="xs:hexBinary" default="00"/>
                <xs:attribute name="aadBase" type="xs:hexBinary" default="00"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Authenticity signaling -->

    <xs:complexType name="ContentAuthenticity">
        <xs:attribute name="keyUriTemplate" type="xs:anyURI" use="required"/>
        <xs:attribute name="authSchemeIdUri" type="xs:anyURI" use="required"/>
        <xs:attribute name="authUrlTemplate" type="xs:anyURI" use="required"/>
        <xs:attribute name="authTagLength" type="xs:unsignedInt"/>
    </xs:complexType>
</xs:schema>
```

Annex B
(informative)

Implementation Guidelines

B.1 Key Delivery

When the `urn:mpeg:dash:sea:keysys:http:2013` key system is used, key delivery should be done over a secure channel (e.g. HTTP over TLS). In case of a live stream, keys should be available several seconds before the segments.

For the `urn:mpeg:dash:sea:aes128-cbc:2013` encryption scheme, initialization vectors can be delivered in the clear.

If short cryptoperiods are used, use of persistent HTTP connections is recommended to avoid the overhead of connection establishment for each key request. Same applies to IV requests, if IV is retrieved via HTTP URL's.

B.2 Encryption

Cryptoperiods should be kept short. Cryptoperiods of 2-10 seconds are a reasonable setting.

Use of unpredictable initialization vectors is highly recommended, especially given highly predictable beginning of media segments, both for MPEG-2 TS and ISO-BMFF. When randomly generating various cryptographic values, cryptographically secure random or pseudo-random number generator should be used, following the industry best practices. One is encouraged to consult NIST special publication 800-90A on recommendations regarding secure random number generation.

Unique IV values per segment should be used with encryption systems based on a cipher operated in stream mode (e.g. GCM). When using AES128-GCM encryption system, one is encouraged to consult with NIST special publication 800-38D for recommendations on safe usage of GCM mode.

For short cryptoperiods, it is recommended to generate IV's locally, as opposed to using HTTP to request them.

B.3 Content Authenticity

When authenticity tags are requested frequently enough, the overhead of connection establishment may be avoided using persistent HTTP connections.

It is recommended that requests for authenticity tags for byte ranges that do not represent a segment or a subsegment will be ignored by the HTTP server, and a 4xx error wil be returned.

It may be advantageous to use HTTPS for requesting an authenticity tag, especially when digests (e.g. SHA) are used.