INTERNATIONAL ORGANISATION FOR STANDARDISATION

ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/IEC JTC1/SC29/WG11

CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 N14849
October 2014, Strasbourg, France

Source	**Editor 23001-7**
Status	**DIS**
Title	**ISO/IEC 23001-7 3nd Edition - Common encryption in ISO base media file format files**
Authors	Kilroy Hughes, David Singer, Zubair Visharam

Error! Reference source not found.

Error! Reference source not found.

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 23001-7 was prepared by Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 29, Coding of audio, picture, multimedia and hypermedia information.

This second edition cancels and replaces the first edition.

ISO/IEC 23001 consists of the following parts, under the general title *Information technology — MPEG systems technologies*:

— *Part 1: Binary MPEG format for XML*

— *Part 2: Fragment Request Units*

— *Part 3: XML IPMP messages*

— *Part 4: Codec configuration representation*

— *Part 5: Bitstream Syntax Description Language (BSDL)*

— *Part 7: Common encryption in ISO base media file format files*

Introduction

Common Encryption specifies standard encryption and key mapping methods that can be utilized to enable decryption of the same file using different Digital Rights Management (DRM) and key management systems. It operates by defining encryption algorithms and encryption related metadata necessary to decrypt the protected streams, yet it leaves the details of rights mappings, key acquisition and storage, DRM content protection compliance rules, etc., up to the DRM system or systems. For instance, DRM systems must support identifying the decryption key via stored key identifiers (KIDs), but how each DRM system protects and locates the KID identified decryption key is left to a DRM-specific method.

DRM specific information such as licenses, rights, and license acquisition information can be stored in an ISO Base Media file using a Protection System Specific Header box (`'pssh'`). Each instance of this box stored in the file corresponds to one applicable DRM system identified by a well-known `SystemID`. DRM licenses or license acquisition information need not be stored in the file in order to look up a separately delivered key using a `KID` stored in the file, and decrypt media samples using the encryption parameters stored in each track.

The second edition of this standard added XML representations of Common Encryption parameters for delivery in XML documents, such as an MPEG DASH Media Presentation Description Documents (MPD). The second edition also defined the `'cbc1'` protection scheme using AES-CBC mode encryption.

The third edition added ‘`cbcs'` and `'cens'` protection schemes for pattern encryption, which encrypt only a fraction of the data Blocks within each video Subsample protected. Pattern encryption reduces the computational power required by devices to decrypt video tracks.

Error! Reference source not found.

1 Scope

Part 7 of ISO/IEC 23001 specifies common encryption formats for use in any file format based on ISO/IEC 14496-12, ISO Base Media File Format. File, track, and track fragment metadata is specified to enable multiple digital rights and key management systems (DRMs) to access the same common encrypted file or stream. This standard does not define a DRM system.

The AES-128 symmetric block cipher is incorporated by reference to encrypt elementary stream data contained in media samples. Both AES counter mode (CTR) and Cipher Block Chaining (CBC) are specified in separate protection schemes. Partial encryption using a pattern of encrypted and clear blocks is also specified in separate protection schemes. The identification of encryption keys, Initialization Vector storage and processing is specified for each scheme.

Subsample encryption is specified for NAL structured video, such as AVC and HEVC, to enable normal processing and editing of video elementary streams prior to decryption.

An XML representation is specified for important common encryption information so that it can be included in XML files as standard elements and attributes to enable interoperable license and key management prior to media file download.

2 Normative references

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

ITU-T Rec. X.667 (09/2004) | ISO/IEC 9834-8:2005, *Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components*

ITU-T Rec.H.264 | ISO/IEC 14496-10, Information technology — *Coding of audio-visual objects — Part 10: Advanced Video Coding*

ISO/IEC 14496-12, *Information technology — Coding of audio-visual objects — Part 12: ISO Base Media File Format*

ISO/IEC 14496-15, *Information technology — Coding of audio-visual objects — Part 15: Carriage of NAL unit structured video in the ISO Base Media File Format*

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

ISO/IEC 23008-2, Information technology – Coding of audio-visual objects – Part 2: High Efficiency Video Coding (HEVC)

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

ISO/IEC 18033-3:2010, *Information technology – Security techniques – Encryption algorithms – Part 3 : Block ciphers*

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

IETF RFC 3406, *Uniform Resource Names (URN) Namespace Definition Mechanisms*, October 2002

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

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace,* July 2005

3 Definitions

3.1 *Terms and definitions*

For the purposes of this document, the following terms and definitions apply. Words used as defined terms and normative terms (SHALL, SHOULD, MAY) are written in upper case to distinguish them from the same word intending its dictionary definition.

Constant IV	Initialization Vector specified in a sample entry or sample group description that applies to all samples and Subsamples under that sample entry or mapped to that sample group
Block	16-byte extent of sample data that may be encrypted or decrypted by the AES-128 block cipher, in which case, a cipher block.
Initialization Vector	8 or 16-byte value used in combination with a key and a 16 byte block of content to create the first cipher block in a chain, and derive subsequent cipher blocks in a cipher block chain.
ISO Base Media File	a file conforming to the file format described in ISO/IEC 14496-12 in which the techniques in ISO/IEC 23001-7 may be used
NAL Unit	syntax structure containing an indication of the type of data to follow and bytes containing that data in the form of an RBSP interspersed as necessary with emulation prevention bytes.
NAL Structured Video	video streams composed of NAL Units of which the carriage is specified by ISO/IEC 14496-15 – Carriage of NAL unit structured video in the ISO Base Media File Format
Protection Scheme	encryption algorithm and information defined in this specification and identified by a four character code in an ISO Media track's Scheme Type Box ('schm').
Subsample	byte range within a sample consisting of an unprotected byte range followed by a protected byte range

3.2 *Abbreviated terms*

For the purposes of this International Standard, the following abbreviated terms apply.

AES	Advanced Encryption Standard as specified in Federal Information Processing Standards Publication 197, FIPS-197
AES-CTR	AES Counter Mode as specified in *Recommendation of Block Cipher Modes of Operation*, NIST, NIST Special Publication 800-38A
AES-CBC	AES Cipher-Block Chaining Mode as specified in *Recommendation of Block Cipher Modes of Operation*, NIST, NIST Special Publication 800-38A
AVC	Advanced Video Coding as specified in ISO/IEC 14496-10
HEVC	High Efficiency Video Coding as specified in ISO/IEC 23008-2
IV	Initialization Vector
NAL	Network Abstraction Layer, as specified in ISO/IEC 14496-10 and ISO/IEC 23008-2
URN	Unique Resource Name
UUID	Universally Unique Identifier

4 Protection schemes

4.1 Scheme type signaling

Scheme signaling SHALL conform to ISO/IEC 14496-12. As defined in ISO/IEC 14496-12, the sample entry is transformed and a Protection Scheme Information Box (`'sinf'`) is added to the standard sample entry in the Sample Description Box to denote that a stream is protected. The Protection Scheme Information Box SHALL contain a Scheme Type Box (`'schm'`) so that the scheme is identifiable. The Scheme Type Box SHALL have the following additional constraints:

- The `scheme_type` field SHALL be set to a value equivalent to a four character code defined in section 10.
- The `scheme_version` field SHALL be set to 0x00010000 (Major version 1, Minor version 0).

The Protection Scheme Information Box SHALL also contain a Scheme Information Box (`'schi'`). The Scheme Information Box SHALL contain a Track Encryption Box (`'tenc'`), describing the default encryption parameters for the track.

4.2 Common encryption scheme types

Four protection schemes are specified in this edition of Common Encryption. Each scheme uses syntax and algorithms specified in Sections 5 through 9, as constrained in Section 10. They are:

1. **`'cenc'`** – AES-CTR mode full sample and video NAL Subsample encryption; see section 10.1

2. **'cbc1'** – AES-CBC mode full sample and video NAL Subsample encryption; see section 10.2

3. **'cens'** – AES-CTR mode partial video NAL pattern encryption; see section 10.3

4. **'cbcs'** – AES-CBC mode partial video NAL pattern encryption; see section 10.4

5 Overview of encryption metadata

The encryption metadata defined by Common Encryption can be categorized as follows:

- Protection System Specific Data – this data is opaque to Common Encryption. This gives protection systems (i.e. key and digital rights management "DRM" systems) a place to store their own data using a common mechanism. This data is contained in the `ProtectionSystemSpecificHeaderBox` described in section 8.1.
- Common encryption information for a media track – this includes default values for the key identifier (`KID`), Initialization Vector and vector size, protection pattern, and protection flag. This data is contained in the `TrackEncryptionBox` described in section 8.2.
- Common encryption information for groups of media samples – this includes overrides to the track level defaults defined above. This allows groups of samples within the track to use different keys, a mix of clear and protected content, share a Constant Initialization Vector (in `'cbcs'` scheme), etc. This data is contained in a `SampleGroupDescriptionBox` (`'sgpd'`) that is referenced by a `SampleToGroupBox` (`'sbgp'`). See section 6 for further details.
- Encryption information for individual media samples – this includes Initialization Vectors and Subsample encryption data. This data is sample auxiliary information, referenced by using a `SampleAuxiliaryInformationSizesBox` (`'saiz'`) and a `SampleAuxiliaryInformationOffsetsBox` (`'saio'`). See section 7 for further details.

6 Encryption parameters shared by groups of samples

Each sample in a protected track SHALL be associated with an `isProtected` flag, `Per_Sample_IV_Size`, `KID`, optional Block pattern information, and an optional `constant_IV`. This can be accomplished by relying on the default values in the `TrackEncryptionBox` (see section 8.2), and optionally specifying parameters by sample group. Encryption parameters specified in a sample group SHALL override the corresponding default parameter values for the samples in that group defined in the `TrackEncryptionBox`. Samples not mapped to any sample group SHALL use the defaults established in the `TrackEncryptionBox`.

When specifying the parameters by sample group, the `SampleToGroupBox` in the sample table or track fragment specifies which samples use which sample group description from the `SampleGroupDescriptionBox`. The format of the sample group description is based on the handler type for the track. For fragmented files, it may be necessary to store both the `SampleToGroupBox` and `SampleGroupDescriptionBox` in each track fragment to make them accessible for decryption of the samples they describe, e.g. when movie fragments are separately stored and delivered by streaming.

Tracks with a handler type of `'vide'` SHALL use the `CencSampleEncryptionInformationVideoGroupEntry` sample group description structure, which has the following syntax:

```
aligned(8) class CencSampleEncryptionInformationVideoGroupEntry
   extends VisualSampleGroupEntry( 'seig' )
{
   unsigned int(8)      reserved = 0;
   unsigned int(4)      crypt_byte_block;
   unsigned int(4)      skip_byte_block;
   unsigned int(8)      isProtected;
   unsigned int(8)      Per_Sample_IV_Size;
   unsigned int(8)[16]  KID;
   if (Per_Sample_IV_Size == 0) {
      unsigned int(8)   constant_IV_size;
      unsigned int(8)[constant_IV_size]   constant_IV;
   }
}
```

Similarly, tracks with other handler types SHALL use the `CencSampleEncryptionInformationGroupEntry` sample group description structure, which has the following syntax:

```
aligned(8) class CencSampleEncryptionInformationGroupEntry
   extends SampleGroupEntry( 'seig' )
{
   unsigned int(8)      reserved = 0;
   unsigned int(4)      crypt_byte_block = 0;
   unsigned int(4)      skip_byte_block = 0;
   unsigned int(8)      isProtected;
   unsigned int(8)      Per_Sample_IV_Size;
   unsigned int(8)[16]  KID;
   if (Per_Sample_IV_Size == 0) {
      unsigned int(8)   constant_IV_size;
      unsigned int(8)[constant_IV_size]   constant_IV;
   }
}
```

These structures use a common semantic for their fields as follows:

- `isProtected` is the flag which indicates the encryption state of the samples in the sample group. See the `isProtected` field in section 9.1 for further details.
- `Per_Sample_IV_Size` is the Initialization Vector size in bytes for samples in the sample group. See the `Per_Sample_IV_Size` field in section 9.1 for further details.
- `KID` is the key identifier used for samples in the sample group. See the `KID` field in section 9.1 for further details.
- `constant_IV_size` is the size of a possible Initialization Vector used for all samples associated with this group (when per-sample Initialization Vectors are not used).
- `constant_IV`, if present, is the Initialization Vector used for all samples associated with this group. See the `constant_IV` field in section 9.1 for further details.
- `crypt_byte_block` specifies the count of the encrypted Blocks in the protection pattern, where each Block is of size 16-bytes. See section 9.1 for further details.
- `skip_byte_block` specifies the count of the unencrypted Blocks in the protection pattern. See section 9.1 for further details.

In order to facilitate the addition of future optional fields, clients SHALL ignore additional bytes after the fields defined in the `CencSampleEncryption` group entry structures.

7 Common encryption sample auxiliary information

7.1 *Definition*

Each protected sample in a protected track SHALL have an Initialization Vector associated with it. Both Initialization Vectors and Subsample encryption information MAY be provided as Sample Auxiliary Information with `aux_info_type` equal to the scheme and `aux_info_type_parameter` equal to 0. For example, for tracks protected using the `'cenc'` scheme, the default value for `aux_info_type` is `'cenc'` and the default value for the `aux_info_type_parameter` is 0, so content SHOULD be created omitting these optional fields. Storage of sample auxiliary information SHALL conform to ISO/IEC 14496-12.

The format of the sample auxiliary information for samples with this type SHALL be:

```
aligned(8) class CencSampleAuxiliaryDataFormat
{
   unsigned int(Per_Sample_IV_Size*8) InitializationVector;
   if (sample_info_size > Per_Sample_IV_Size )
   {
      unsigned int(16) subsample_count;
      {
         unsigned int(16) BytesOfClearData;
         unsigned int(32) BytesOfProtectedData;
      } [subsample_count ]
   }
}
```

Where:

`sample_info_size` is the size of the sample auxiliary information for this sample from the Sample Auxiliary Information Size Box (`'saiz'`);

`InitializationVector` is the Initialization Vector for the sample. See the `InitializationVector` field in section 9.1 for further details.

`subsample_count` is the count of Subsamples for this sample. See the `subsample_count` field in 9.1 for further details.

`BytesOfClearData` is the number of bytes of clear data in this Subsample. See the `BytesOfClearData` field in section 9.1 for further details.

`BytesOfProtectedData` is the number of bytes of protected data in this Subsample. See the `BytesOfProtectedData` field in section 9.1 for further details.

If Subsample encryption is not used (the size of the sample auxiliary information equals Per_Sample_IV_Size), then the entire sample is protected (see section 9.4 for further details). In this case, all auxiliary information will have the same size and hence the `default_sample_info_size` of the SampleAuxiliaryInformationSizes box will be equal to the `Per_Sample_IV_Size` of the Initialization Vectors. If Per_Sample_IV_Size is also zero (Constant IVs are in use) then the sample auxiliary information would then be empty and SHALL be omitted.

Note: even if Subsample encryption is used, the size of the sample auxiliary information may be the same for all of the samples (if all of the samples have the same number of Subsamples) and the `default_sample_info_size` may be used.

7.2 *Sample Encryption Information box for storage of sample auxiliary information*

7.2.1 Sample Encryption Box (‘senc’)

Box Type: ‘senc’
Container: Track Fragment Box (‘traf’) or Track Box (‘trak’)
Mandatory: No
Quantity: Zero or one

An optional storage location for Sample Auxiliary Information is the Sample Encryption Information Box (‘senc’), specified here.

The Sample Encryption Box contains sample auxiliary information, and may contain a per sample Initialization Vector for each sample, and clear and protected byte ranges of partially protected video samples (“Subsample encryption”). It MAY be used when samples in a track or track fragment are protected. Storage of ‘senc’ in a Track Fragment Box makes the necessary Sample Auxiliary Information accessible within the movie fragment for all contained samples in order to make each track fragment independently decryptable; for instance, when movie fragments are delivered as DASH Media Segments.

7.2.2 Syntax

```
aligned(8) class SampleEncryptionBox
   extends FullBox(‘senc’, version=0, flags)
{
   unsigned int(32)  sample_count;
   {
      unsigned int(Per_Sample_IV_Size*8)  InitializationVector;
      if (flags & 0x000002)
      {
         unsigned int(16)  subsample_count;
         {
            unsigned int(16)  BytesOfClearData;
            unsigned int(32)  BytesOfProtectedData;
         } [ subsample_count ]
      }
   }[ sample_count ]
}
```

7.2.3 Semantics

`flags` is inherited from the `FullBox` structure. The `SampleEncryptionBox` currently supports the following bit values:

- 0x2 – `UseSubSampleEncryption`
  - If the `UseSubSampleEncryption` flag is set, then the track fragment that contains this Sample Encryption Box SHALL use Subsample encryption as described in section 9.5. When this flag is set, Subsample mapping data follows each `InitializationVector`. The Subsample mapping data consists of the number of Subsamples for each sample, followed by an array of values describing the number of bytes of clear data and the number of bytes of encrypted data for each Subsample.

`sample_count` is the number of protected samples in the containing track or track fragment. This value SHALL be either zero (0) or the total number of samples in the track or track fragment.

`InitializationVector` SHALL conform to the definition specified in section 9.2. Only one `Per_Sample_IV_Size` SHALL be used within a file, or `Per_Sample_IV_Size` SHALL be zero when a sample is unencrypted or a Constant IV is in use. Selection of `InitializationVector` values SHOULD follow the recommendations of section 9.2.
`subsample_count` SHALL conform to the definition specified in section 9.1.
`BytesOfClearData` SHALL conform to the definition specified in section 9.1.
`BytesOfProtectedData` SHALL conform to the definition specified in section 9.1.

8 Box Definitions

8.1 *Protection System Specific Header box*

8.1.1 Definition

Box Type: `` `pssh' ``
Container: Movie (`moov`) or Movie Fragment (`moof`)
Mandatory: No
Quantity: Zero or more

This box contains information needed by a Content Protection System to play back the content. The data format is specified by the system identified by the `'pssh'` parameter `SystemID`, and is considered opaque for the purposes of this specification. The collection of Protection System Specific Header boxes from the initial movie box, together with those in a movie fragment, SHALL provide all the required Content Protection System information to decode that fragment.

The data encapsulated in the `Data` field MAY be read by the identified Content Protection System client to enable decryption key acquisition and decryption of media data. For license/rights-based systems, the header information MAY include data such as the URL of license server(s) or rights issuer(s) used, embedded licenses/rights, embedded keys(s), and/or other protection system specific metadata.

A single file MAY be constructed to be playable by multiple key and digital rights management (DRM) systems, by including Protection System Specific Header boxes for each system supported. In order to find all of the Protection System Specific data that is relevant to a sample in the presentation readers SHALL:

- Examine all Protection System Specific Header boxes in the Movie Box and in the Movie Fragment Box associated with the sample (but not those in other Movie Fragment Boxes).
- Match the `SystemID` field in this box to the `SystemID`(s) of the DRM System(s) they support
- Match the `KID` associated with the sample (either from the `default_KID` field of the Track Encryption Box or the `KID` field of the appropriate sample group description entry) with one of the `KID` values in the Protection System Specific Header Box. Boxes without a list of applicable `KID` values, or with an empty list, SHALL be considered to apply to all `KID`s in the file or movie fragment.

Protection System Specific Header data SHALL be associated with a sample based on a matching `KID` value in the `'pssh'` and sample group description or default `'tenc'` describing the sample. If a sample or set of samples is moved due to file defragmentation or refragmentation or removed by editing, then the associated Protection System Specific Header boxes for the remaining samples SHALL be stored following the above requirements.

8.1.2 Syntax

```
aligned(8) class ProtectionSystemSpecificHeaderBox extends FullBox('pssh',
version, flags=0)
{
   unsigned int(8)[16]     SystemID;
   if (version > 0)
   {
      unsigned int(32)     KID_count;
      {
      unsigned int(8)[16]  KID;
      } [KID_count];
   }
   unsigned int(32)           DataSize;
   unsigned int(8)[DataSize]  Data;
}
```

8.1.3 Semantics

`SystemID` specifies a UUID that uniquely identifies the content protection system that this header belongs to.
`KID_count` specifies the number of KID entries in the following table. The value MAY be zero.
`KID` identifies a key identifier that the Data field applies to. If not set, then the `Data` array SHALL apply to all KIDs in the movie or movie fragment containing this box.
`DataSize` specifies the size in bytes of the Data member.
`Data` holds the content protection system specific data.

8.2 *Track Encryption box*

8.2.1 Definition

Box Type: `` `tenc' ``
Container: Scheme Information Box (`schi`)
Mandatory: No (Yes, for protected tracks)
Quantity: Zero or one

The `TrackEncryptionBox` contains default values for the `isProtected` flag, `Per_Sample_IV_Size`, and `KID` for the entire track. In the case where pattern-based encryption is in effect, it supplies the pattern; and when Constant IVs are in use, it supplies the Constant IV. These values are used as the encryption parameters for the samples in this track unless over-ridden by the sample group description associated with a group of samples. For files with only one key per track, this box allows the basic encryption parameters to be specified once per track instead of being repeated per sample.

If the `Per_Sample_IV_Size` is indicated as zero, then the `default_constant_IV_size` for all samples that use these settings SHALL be present. A Constant IV SHALL NOT be used with counter-mode encryption. The version field of the `TrackEncryptionBox` SHALL be set to a value greater than zero when `Per_Sample_IV_Size` is zero, and to zero otherwise. A sample group description may supply keys and/or IVs for sample groups that override these default values for those samples mapped to the group.

8.2.2 Syntax

```
aligned(8) class TrackEncryptionBox extends FullBox('tenc', version, flags=0)
{
   unsigned int(8)      reserved = 0;
   if (version==0) {
      unsigned int(8)   reserved = 0;
   }
   else { // version is 1 or greater
      unsigned int(4)   default_crypt_byte_block;
      unsigned int(4)   default_skip_byte_block;
   }
   unsigned int(8)      default_isProtected;
   unsigned int(8)      default_Per_Sample_IV_Size;
   unsigned int(8)[16]  default_KID;
   if (default_Per_Sample_IV_Size == 0) {
      unsigned int(8)   default_constant_IV_size;
      unsigned int(8)[default_constant_IV_size] default_constant_IV;
   }
}
```

8.2.3 Semantics

`version` SHALL be zero unless pattern-based encryption is in use, whereupon it SHALL be 1.
`default_isProtected` is the protection flag which indicates the default protection state of the samples in the track. See the `isProtected` field in 9.1 for further details.
`default_Per_Sample_IV_Size` is the default Initialization Vector size in bytes. See the `Per_Sample_IV_Size` field in section 9.1 for further details.
`default_KID` is the default key identifier used for samples in this track. See the `KID` field in section 9.1 for further details.
`default_constant_IV_size` is the size of a possible default Initialization Vector for all samples.
`default_constant_IV`, if present, is the default Initialization Vector for all samples. See the `constant_IV` field in section 9.1 for further details.
`default_crypt_byte_block` specifies the count of the encrypted Blocks in the protection pattern, where each Block is of size 16-bytes. See section 9.1 for further details.
`default_skip_byte_block` specifies the count of the unencrypted Blocks in the protection pattern. See the `skip_byte_block` field in section 9.1 for further details.

9 Encryption of media data

9.1 *Field semantics*

Within the sample groups and sample auxiliary information used by the common encryption scheme, these fields have the following semantics:

`isProtected` is the identifier of the protection state of the samples in the track or group of samples. This flag takes the following values:
- 0x0: Not protected
- 0x1: protected (as signalled by the `scheme_type` field of the scheme type box `'schm'`, e.g. for `scheme_type` of `'cenc'`, the track default is AES-CTR encrypted using the `'cenc'` scheme)
- 0x02 – 0xFF: Reserved

`Per_Sample_IV_Size` is the size in bytes of the `InitializationVector` field. Supported values:
- 0 – If the `isProtected` flag is 0x0 (Not Protected) or Constant IVs are in use
- 8 – Specifies 64-bit Initialization Vectors.
- 16 – Specifies 128-bit Initialization Vectors.

`constant_IV_size` is the size in bytes of the `constant_IV` field. Supported values:
- 8 – Specifies 64-bit Initialization Vectors.

16 – Specifies 128-bit Initialization Vectors.

`KID` is a key identifier that uniquely identifies the key needed to decrypt the associated samples within the scope of an application so that KID is sufficient to identify a separately stored license containing the key that was used to encrypt the content. This allows the identification of multiple encryption keys per file or track. Unprotected samples in a protected track SHALL be identified by having an `isProtected` flag of 0x0, a `Per_Sample_IV_Size` of 0x0, and a `KID` value of 0x0. It is strongly recommended to use UUIDs [2] as `KIDs` in order to satisfy the uniqueness requirement across all applications.

`InitializationVector` specifies the Initialization Vector (IV) needed for decryption of a sample. For an `isProtected` flag of 0x0, no Initialization Vectors are needed and the auxiliary information SHOULD have a size of 0, i.e. not be present.

For an `isProtected` flag of 0x1:

- IVs shall be supplied using `Per_Sample` IVs or Constant IVs.
- If the `Per_Sample_IV_Size` field is 16 then `InitializationVector` specifies the entire 128-bit IV value
- If the `Per_Sample_IV_Size` field is 8, then its value is copied to bytes 0 to 7 of the Initialization Vector and bytes 8 to 15 of the Initialization Vector are set to zero.

`subsample_count` specifies the number of Subsample encryption entries present for this sample. If present this field SHALL be greater than 0.

`BytesOfClearData` specifies the number of bytes of clear data at the beginning of this Subsample encryption entry. (Note: this value may be zero if no clear bytes exist for this Subsample.)

`BytesOfProtectedData` specifies the number of bytes of protected data following the clear data. (Note: this value may be zero if no protected bytes exist for this Subsample.) The Subsample encryption entries SHALL NOT include an entry with a zero value in both the `BytesOfClearData` field and in the `BytesOfProtectedData` field. The total length of all `BytesOfClearData` and `BytesOfProtectedData` in a sample SHALL equal the length of the sample. Subsample encryption entries SHOULD be as compactly represented as possible. For example, instead of two entries with {15 clear, 0 protected}, {17 clear, 500 protected} use one entry of {32 clear, 500 protected}. If pattern-based encryption is used, then the pattern applies to the protected byte range, `BytesOfProtectedData`; otherwise all protected bytes are encrypted.

`crypt_byte_block` shall be zero unless pattern-based encryption is in effect. See section 9.6 for further details.

`skip_byte_block` shall be zero unless pattern-based encryption is in effect. See section 9.6 for further details.

9.2 *Initialization Vectors*

The Initialization Vector (IV) values for each sample SHALL be either a Constant IV, and located in the sample entry or a sample group description; or SHALL be signaled per sample, and be located in the Sample Auxiliary Information associated with each protected sample. See section 9.1 for additional details on how Initialization Vectors are formed and stored.

It is recommended that applications applying encryption generate a random number for the first Initialization Vector in a sequence.

- For 8-byte `Per_Sample_IV_Size`, Initialization Vectors for subsequent samples SHOULD be created by incrementing the 8-byte Initialization Vector and padding the least significant bits with zero to construct a 16-byte number. Using a random starting value for a track introduces entropy into the Initialization Vector values, and incrementing the most significant 8 bytes for each sample in sequence ensures that each 16-byte IV and CTR counter value combination is unique.

  The 8-byte Initialization Vector can roll over from the maximum value

(`0xFFFFFFFFFFFFFFFF`) to the minimum value (0x0) if the maximum 8-byte value is exceeded when incrementing the 8-byte value per sample from its random starting value. 8-byte IVs are recommended for per sample IVs to reduce storage size, and guarantee unique counter values for CTR mode counter blocks.

- For 16-byte `Per_Sample_IV_Size`, Initialization Vectors for subsequent samples using CTR mode MAY be created by adding the cipher block count of the previous sample to the Initialization Vector of the previous sample. Using a random starting value introduces entropy into the Initialization Vector values, and incrementing by cipher block count for each sample ensures that each CTR counter block is unique. Even though the least significant bytes of the IV (bytes 8 to 15) are incremented as an unsigned 64-bit counter in CTR mode, the Initialization Vector SHOULD be treated as a 128-bit number when calculating the next Initialization Vector from the previous 16-byte IV.

  CBC mode Initialization Vectors need not be unique per sample or Subsample, and may be generated randomly or sequentially, e.g. a per sample IV may be equal to the cipher text of the last encrypted cipher block (a continuous cipher block chain across samples), or generated by incrementing the previous IV by the number of cipher blocks in the last sample or by a fixed amount. Each `'cbc1'` IV is stored in sample auxiliary information, so its method of derivation is irrelevant for decryption.

Storing a unique IV per sample for both CTR and CBC mode increases cryptographic entropy, and provides random access, and error recovery for each sample (vs. continuous chaining of multiple samples). CTR mode requires a unique counter value for each cipher block sharing a key.

The `'cbcs'` scheme uses a Constant IV, either as a track default, or for a group of samples mapped to a sample group. It is assumed that compressed video data is random enough to allow the reuse of the same IV for each Subsample when a Constant IV is reused on multiple Subsamples and samples. A Constant IV reduces IV data size vs. a per-sample IV, which requires 8 or 16 bytes per sample. For a fragmented file, a Constant IV typically requires one sample group box per track fragment and a sample group description box containing the sample group's IV.

9.3 *AES-CTR mode counter operation*

Media data using the `'cenc'` or `'cens'` protection schemes SHALL use the Advanced Encryption Standard, specified in Federal Information Processing Standards Publication 197, FIPS-197 published by the United States National Institute of Standards and Technology (NIST) using 128-bit keys in Counter Mode (AES-CTR), as specified in Recommendation of Block Cipher Modes of Operation, NIST, NIST Special Publication 800-38A.

AES-128 CTR mode is a 16 byte block cipher that can encrypt an arbitrary sized byte stream without need for padding or leaving a clear remainder when the last Block of sample data is a partial Block (1 to 15 bytes in size). Counter mode (CTR) operates by encrypting a counter block using the AES block encryption algorithm using the key indicated by KID, and then XOR-ing the result with the data to be encrypted or decrypted.

The CTR mode counter block SHALL be constructed from a per sample IV, and incremented as described below and in section 9.2.

When an 8 byte `Per_Sample_IV_Size` is indicated, the least significant 8 bytes of the 16 byte IV (bytes 8 to 15) SHALL be set to zero and used as a 64 bit block counter that is incremented by one for each subsequent 16 byte cipher block of encrypted sample data.

When a 16 byte `Per_Sample_IV_Size` is indicated, and the least significant 8 bytes (64 bit counter) reaches the maximum value (0xFFFFFFFFFFFFFFFF), then incrementing it SHALL reset the 8 byte block counter to zero (bytes 8 to 15) without affecting the other 8 bytes of the counter (bytes 0 to 7).

Within each sample, encrypted data SHALL be a logically continuous byte sequence of 16 byte Blocks, regardless of physically interleaved clear data identified by Subsample encryption or pattern encryption. Only the last cipher block in a sample MAY be a partial cipher block (less than 16 bytes). The counter SHALL be incremented by one after each encrypted cipher block, and restarted on the next sample using the `InitializationVector` stored in sample auxiliary information.

9.4 *Full sample encryption*

9.4.1 *General*

Full sample encryption SHALL be used for all encrypted media types other than NAL Structured video, which SHALL use Subsample encryption.

CTR mode full sample encryption (`'cenc'` scheme) SHALL encrypt the entire sample.

CBC mode full sample encryption (`'cbc1'` scheme) SHALL encrypt the entire sample, except for any partial Block (less than 16 bytes in size) that may remain at the end of the sample.

9.4.2 *Full sample encryption using AES-CTR mode*

AES-CTR mode encryption SHALL use the `'cenc'` scheme with a unique IV per sample, and encrypt all bytes in the sample.

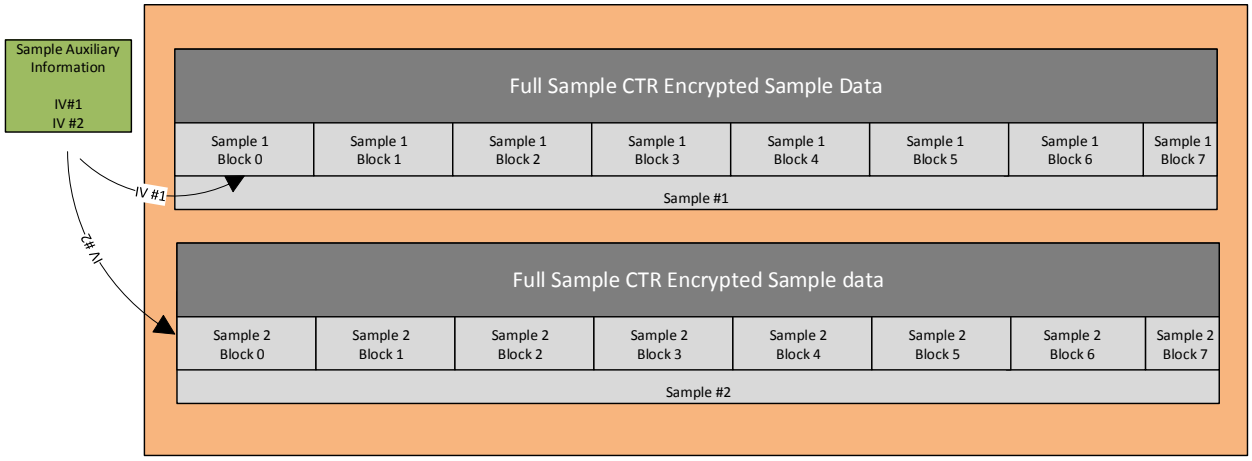

Figure 1— Full sample encryption using AES-CTR mode

Note: AES-CTR mode is a block cipher that can encrypt complete samples that are not a multiple of 16 bytes in size. Cipher blocks are shown to illustrate the underlying Blocks used to encrypt the samples. Block 7 is shown as smaller than 16 bytes to illustrate that CTR mode can encrypt partial cipher blocks, i.e. smaller than 16 bytes. Each sample starts with a unique IV.

9.4.3 *Full sample encryption using AES-CBC mode*

The full sample CBC mode encryption scheme (`'cbc1'`) SHALL use the Advanced Encryption Standard specified by AES [FIPS197] using 128-bit keys in Cipher Block Chaining mode (AES-CBC-128), as specified in Block Cipher Modes [NIST 800-38A]. Per sample IVs SHALL be stored in sample auxiliary information as defined in section 7.

Encrypted NAL Structured Video tracks SHALL use Subsample protection as defined in section 9.5. All other types of encrypted tracks SHALL use full sample encryption using either `'cenc'` or `'cbc1'` protection schemes.

AES-CBC mode full sample encryption SHALL leave any partial Block at the end of a sample unencrypted (i.e. when the last Block in the sample is less than 16 bytes). Each sample SHALL be encrypted as a continuous cipher block chain starting with an Initialization Vector, which MAY be specified per sample by Sample Auxiliary Information, or constant over multiple samples specified by a sample group and sample group description. The `'cbc1'` scheme SHALL use a per-sample IV.

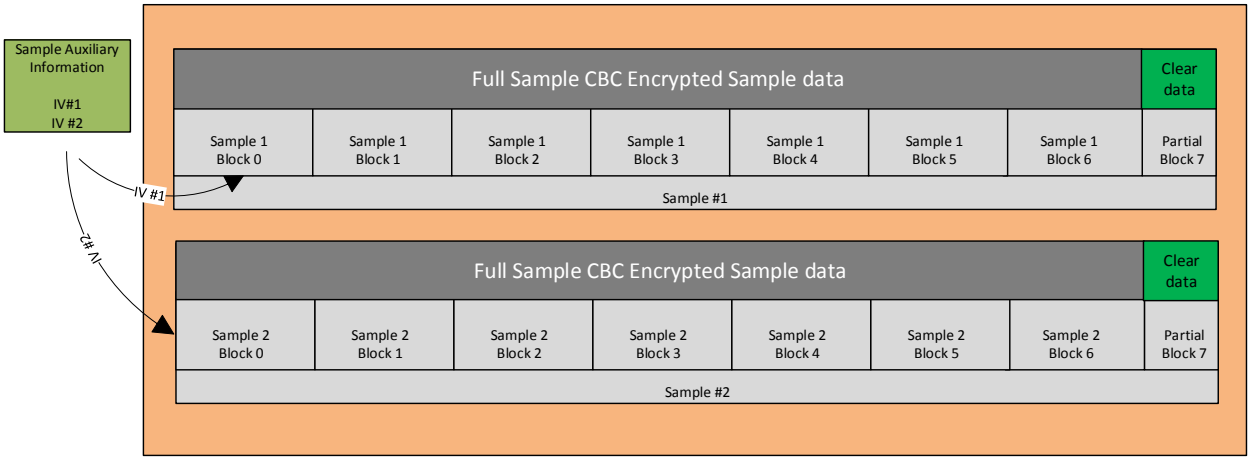

Figure 2 – Full Sample encryption using AES-CBC mode

Note: AES-CBC mode requires all encrypted cipher blocks to be 16 bytes, and the schemes defined in this specification leave partial Blocks unencrypted. (Block 7 is shown as smaller than 16 bytes to illustrate that CBC mode does not encrypt Blocks smaller than 16 bytes to avoid adding padding that would change the file size). Per sample IVs are applied at the start of each sample with `'cbc1'` full sample encryption.

9.5 *Subsample encryption*

9.5.1 Definition (Normative)

Subsample encryption SHALL divide each sample into one or more contiguous Subsamples. Each Subsample SHALL have an unprotected part followed by a protected part, only one of which MAY be zero bytes in length. (Note: usually both are non-zero values). The total length of all of the Subsamples (`BytesOfClearData + BytesOfProtectedData` for all Subsamples that make up a sample) SHALL be equal to the size of the sample itself, and they SHALL not overlap.

For all schemes except the '`cbcs'` scheme, the protected byte sequences of a sample SHALL be treated as a logically continuous chain of 16 byte cipher blocks, even when they are separated by Subsample `BytesOfClearData,` or a `skip_byte_block`.

The `'cbcs'` scheme SHALL treat each Subsample as a separate chain of cipher blocks, starting with the Initialization Vector associated with the sample.

The CTR mode counter SHALL be incremented after each complete encrypted cipher block, ignoring Subsample boundaries. CBC mode cipher block chaining for the `'cbc1'` scheme SHALL be continuous per sample after the IV is applied to the first cipher block in the sample.

All cipher blocks except possibly the last cipher block in a sample when using CTR mode SHALL be 16 bytes. A partial CTR cipher block MAY be encrypted as the last Block of a sample when terminated by a Subsample `BytesOfProtectedData` range.

For `'cenc'` and `'cens'` protection schemes, `BytesOfProtectedData` SHOULD be adjusted to a multiple of 16 bytes to avoid partial Blocks at the end of Subsamples. Application specifications may prohibit partial CTR cipher blocks and can require Subsample Block end alignment to reduce the complexity of decryption.

For `'cbc1'` protection scheme, `BytesOfProtectedData` size SHALL be adjusted to a multiple of 16 bytes to avoid partial Blocks at the end of Subsamples.

For `'cbcs'` protection scheme a partial Block at the end of a Subsample SHALL remain unencrypted. CBC mode cipher block chaining for the `'cbcs'` scheme SHALL be continuous per Subsample, and the IV applied to the first encrypted cipher block of each Subsample. Application specifications can require `BytesOfProtectedData` to start on the first complete byte of video slice data so that a size that is a 16 byte multiple may not be possible, making partial Blocks in Subsamples unavoidable.

Figure 3 – CTR Subsample encryption using one IV per sampleis a Subsample encryption example showing two samples, each containing two Subsamples, each with a per-sample Initialization Vector and a logically continuous sequence of 16-byte cipher blocks interspersed with unencrypted byte ranges.

Note that Block 2 of sample 1 is continued in the second Subsample, which is possible, but not recommended, with the scheme `'cenc'`, but not `'cens'`, `'cbc1'` and `'cbcs'`.

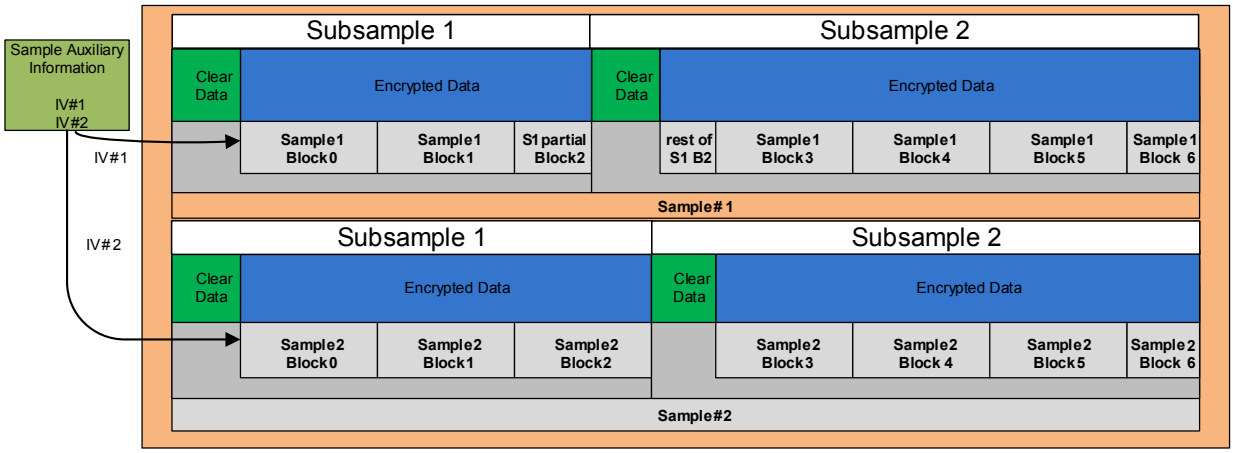

Figure 3 – CTR Subsample encryption using one IV per sample

Note: Cipher block and counter chaining is continuous from Subsample 1 to Subsample 2 in the first sample so that all cipher blocks are 16 bytes, except possibly the last cipher block in each sample. Block 6 is shown as smaller than 16 bytes to illustrate that CTR mode can encrypt cipher blocks smaller than 16 bytes without adding padding that changes the file size. Block 6 would be unencrypted if CBC mode were used.

9.5.2 Subsample encryption of NAL Structured Video tracks

9.5.2.1 Structure of NAL video samples and the use of Subsamples (Informative)

This subsection describes the methods and reasons for using Subsample encryption on NAL Structured Video samples.

NAL (Network Abstraction Layer) structured video specifications define NAL unit syntax elements that can be sequenced to form elementary streams, and access units that can be decoded to images. ISO/IEC 14496-15 specifies how NAL Structured Video is stored in ISO Base Media files, and how each access unit is stored as a sample in a track. Each sample is composed of multiple NAL units, and each NAL unit is separated by a Length field stating the length of the NAL unit. Each NAL unit contains a NAL type header, and video NALs contain a slice header.

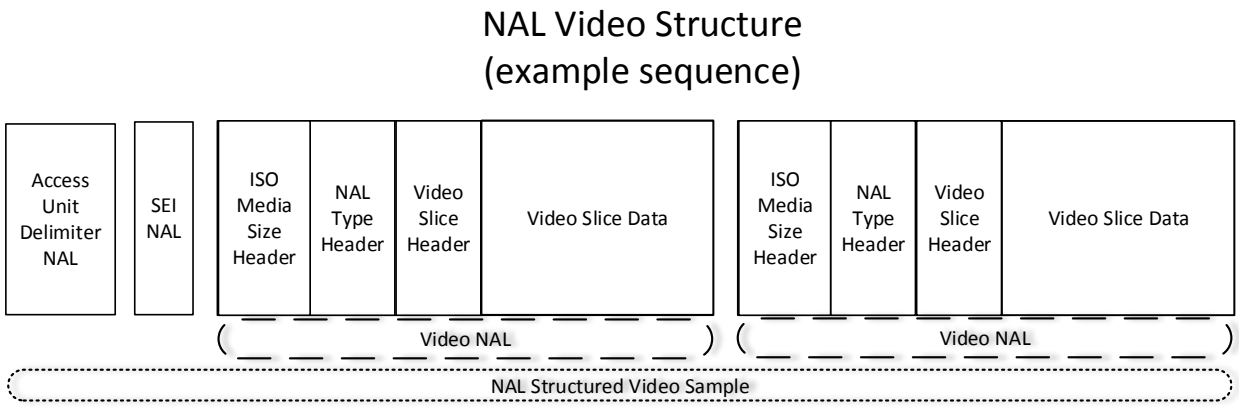

Figure 4 — NAL Structured Video ISO Media sample containing multiple NAL Units

Secure video processors typically do not make data from the video stream that has been decrypted available to applications in order to protect decrypted video, so display applications that need to access information stored in video slice headers or SEI NAL units, such as caption and framing information, will not be able to access that data if it is protected. To keep video encryption keys secure, the same key should not be used to encrypt audio tracks, which typically do not have the same level of key protection as video. Some of the video slice data may remain unencrypted in order to align encrypted bytes, or to align cipher blocks to eliminate the need for partial cipher block decryption in devices. Because NAL Structured Video is usually compressed by spatial and temporal prediction, and the result entropy coded (e.g. CABAC), the loss of portions of a sample will still make it nearly impossible to reconstruct a picture and pictures that predict from it.

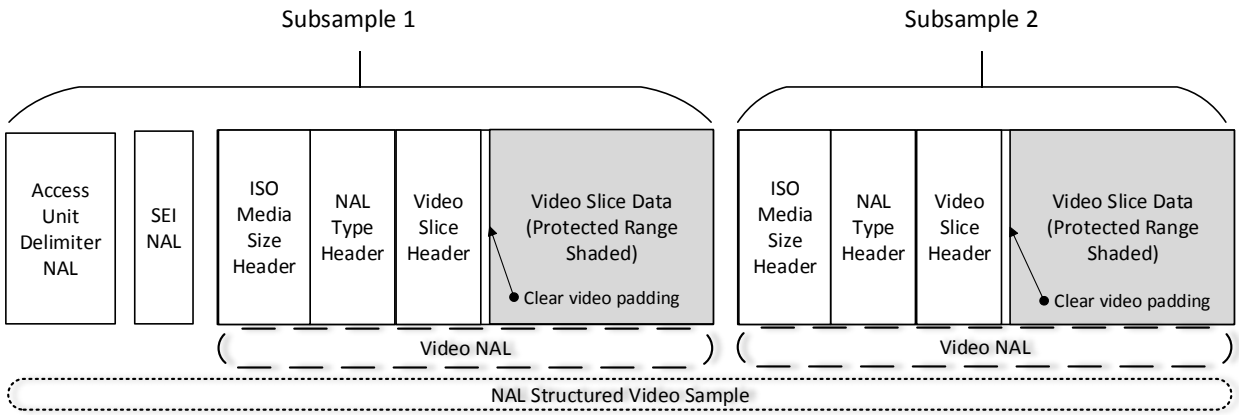

Figure 5 - Subsample encryption applied to NAL Structured Video

Note: the Protected Range of a Subsample may not encrypt all video data. The start of the range may leave some video unencrypted to accomplish byte or 16 byte Block alignment of the Protected Range. The Protected Range may also be partially encrypted by the `'cens'` and `'cbcs'` schemes, which apply a pattern of encrypted and clear Blocks in the Protected Range.

Not all decoders are designed to decode ISO Media format streams that include size headers and exclude decoding parameter NALs, such as Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) NALs. Some decoders are designed to decode video elementary streams in ISO/IEC 14496-10 Annex B byte stream format with startcode delimited NAL Units and SPS/PPS parameter NALs following each access point in the stream. It may be necessary to reformat Common Encrypted ISO Media elementary streams to byte stream format prior to decoding. It may also be necessary to reformat Common Encrypted elementary streams in order to transmit the data using a network protocol like RTP that packetizes NAL Units, or repackage Common Encrypted elementary streams between ISO Media and MPEG-2 Transport Stream containers. Leaving non-video NAL units and all NAL size and type headers unencrypted allows reformatting the elementary stream without decrypting.

Full sample encryption prevents video stream reformatting and information access prior to decrypting the samples. But, if NAL headers and complete NALs other than video types are left unencrypted, an application can convert ISO Media video samples, such as `'avc1', 'avc3', 'hev1'`, etc., to Annex B byte streams by replacing unencrypted NAL size headers with start codes matching the NAL type indicated in the NAL type header, and if necessary, inserting PPS/SPS NAL units following each Access Unit Delimiter NAL that starts a random access point (usually an IDR picture).

Common Encryption specifies Subsample encryption for NAL Structured Video that only encrypts video data, and leaves other NAL types, all NAL size and type headers, and video slice headers unencrypted. Encryptors must be aware of the NAL structure, but decryptors may be video format agnostic and simply decrypt the byte ranges indicated by Subsample information stored in Sample Auxiliary Information. Encryption of only the video slice data allows applications to access information in SEI NALs as well as picture information in video slice headers. Access to video NAL slice header information may be necessary for presentation applications to manage picture buffers, layers, tiles, parallel slice decoding, etc. by reading slice header information prior to secure video decryption.

9.5.2.2 Subsample encryption applied to NAL Structured Video (Normative)

NAL Structured Video samples SHALL be exactly spanned by one or more contiguous Subsamples. The slice data in a video NAL MAY be spanned by multiple Subsamples to create multiple clear and protected ranges, or to span protected slice data that is larger than the maximum size of a single `BytesOfProtectedData` field, with `BytesOfClearData` size equal to zero in each Subsample. Multiple unprotected NALs SHOULD be spanned by a single Subsample clear range, but a large clear range MAY be spanned by multiple Subsamples with zero size `BytesOfProtectedData.`

- For AVC video using `'avc1'` sample description stream format, the NAL `lengthSizeMinusOne` field and the `nal_unit_type` field (the first byte after the length) of each NAL unit SHALL be unencrypted, and only video data in slice NALs SHOULD be encrypted.

  Note 1: Encrypted slice headers were not prohibited in the first edition of this standard but were prohibited by application specifications. A “SHOULD” requirement to leave slice headers unencrypted for ‘avc1’ allows possible legacy content with encrypted slice headers to remain conformant to this new edition. But, new content should not encrypt slice headers, or it may not decode properly in secure video decoders.

Note 2: The size of the length field is variable length. It can be 1, 2, or 4 bytes long and is specified in the Sample Entry for the track as the `lengthSizeMinusOne` field in the `AVCDecoderConfigurationRecord`

- For other NAL Structured Video sample description stream formats (e.g. `'avc3'`, `'hvc1'`, `'hev1'`, etc.), only video slice data SHALL be protected. For avoidance of doubt: Video NAL slice, size and type headers SHALL be unencrypted and other NAL types SHALL be unencrypted.

- There MAY be multiple Subsamples per NAL, and MAY be multiple NALs per Subsample, e.g. when multiple unencrypted NALs are included in one clear byte range for efficient representation.

- Partial video encryption MAY be implemented using multiple Subsamples per video NAL that indicate multiple clear and protected byte ranges per video slice, however pattern encryption (using `'cens'` and `'cbcs'` schemes) SHOULD be used for more efficient representation of partial encryption.

9.5.2.3 Subsample encryption using AES-CTR mode applied to video NALs

Figure 6 details the IVs used, the areas of clear data, the areas of protected data, as well as the NAL unit and sample boundaries. The diagram applies to `'cenc'` and `'cens'` protection schemes.

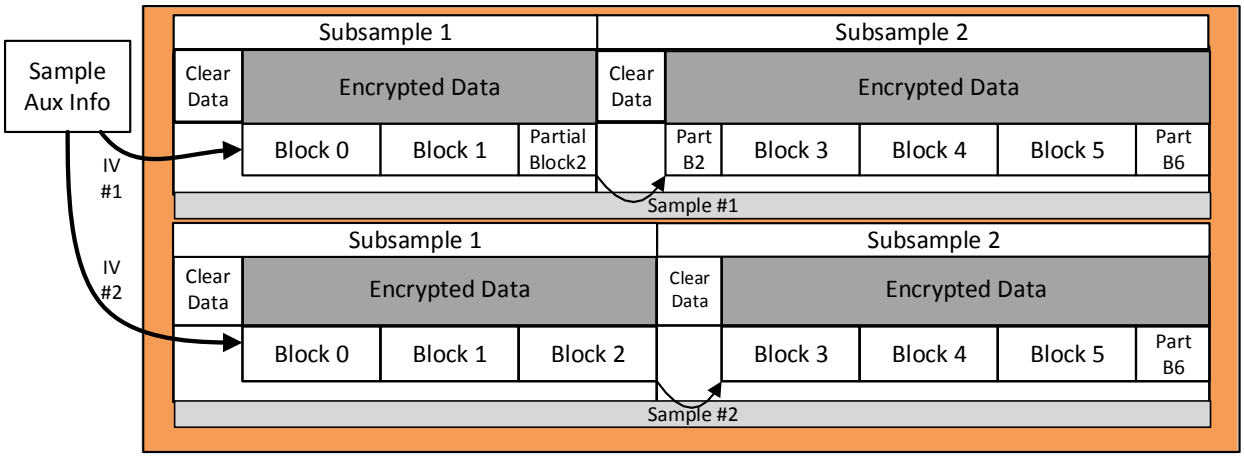

Figure 6 — Subsample encryption of video NALs using AES-CTR

Note: AES-CTR mode is a block cipher that can encrypt partial cipher blocks. Cipher blocks are shown to illustrate the underlying cipher block chain that spans each sample. The last Blocks (Block 6) in both Sample #1 and Sample #2 are less than 16 bytes to illustrate that CTR mode allows encryption of partial cipher blocks. Also note that cipher block 2 of Sample #1 is continued in the next Subsample to form a 16 byte cipher block with one counter value. This example shows Subsamples that match the size of each video NAL unit, but that is not a general constraint of this specification. The protection scheme `'cens'` may apply a pattern of encrypted and clear Blocks to the range labelled “Encrypted Data”.

9.5.2.4 Subsample encryption using 'cbc1' AES-CBC mode applied to video NALs

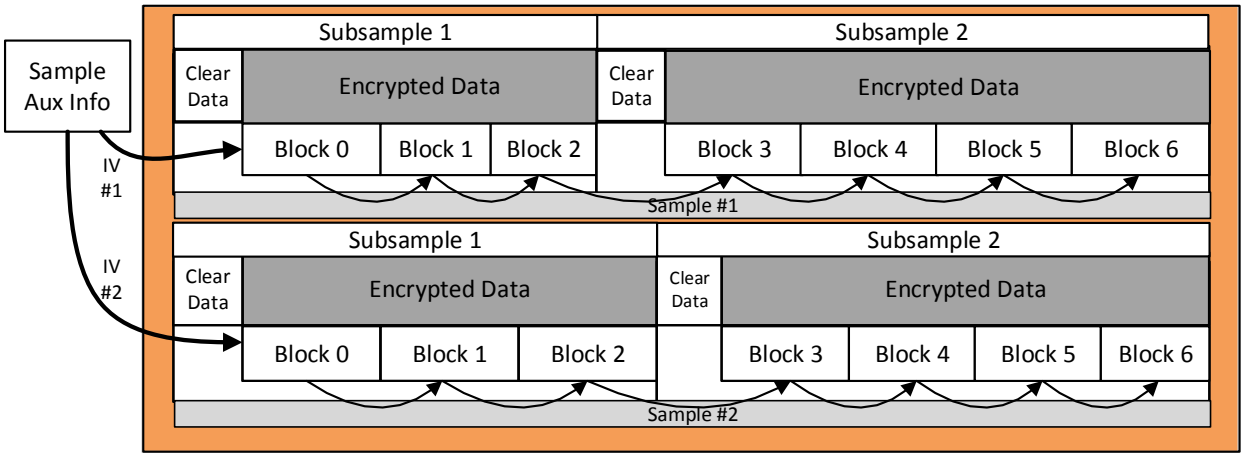

Figure 7 — Subsample encryption of video NALs using AES-CBC mode and 'cbc1' scheme

Note: AES-CBC mode 'cbc1' scheme starts each sample with a per sample IV, then forms 16 byte cipher blocks regardless of spanning Subsample `BytesOfClearData`. Clear data is sized appropriately so that the last Block in each Subsample is 16 bytes (Blocks 2 and 6 in this example).

9.5.2.5 Subsample encryption using 'cbcs' AES-CBC applied to video NALs

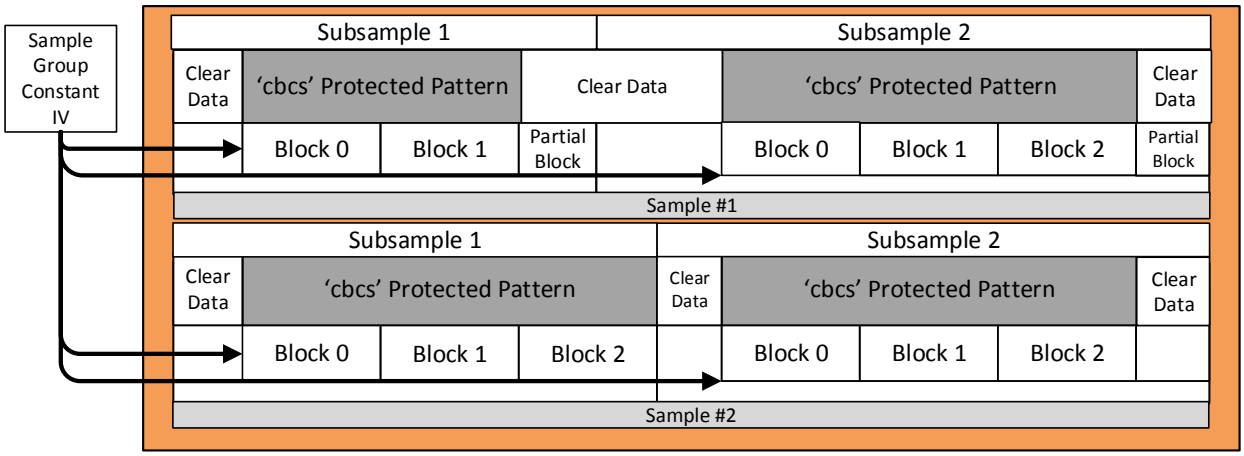

Figure 8 - Subsample encryption using AES-CBC mode and 'cbcs' scheme

Note: AES-CBC mode 'cbcs' scheme starts each Subsample with a Constant IV, then encrypts complete 16 byte cipher blocks leaving any partial Blocks unencrypted at the end of the Subsample's `BytesOfProtectedData`. The protection pattern consists of a sequence of `crypt_byte_block` encrypted cipher blocks followed by `skip_byte_block` clear Blocks, terminated by the end of the `BytesOfProtectedData` range. If the last Block in the range is partial, it is unencrypted.

9.6 *Pattern encryption*

9.6.1 *Definition*

Pattern encryption utilizes a pattern of encrypted and clear ("skipped") 16 byte Blocks over the protected range of a Subsample.

Note: Subsamples are defined to protect only video slice data, leaving NAL size, NAL type, video slice headers, and other NAL types in the clear.

When the fields `default_crypt_byte_block` and `default_skip_byte_block` in a version 1 Track Encryption Box (‘tenc’) are non-zero numbers, pattern encryption SHALL be applied.

The pattern SHALL consist of the number of encrypted cipher blocks indicated by the field `default_crypt_byte_block` or `crypt_byte_block` (if present in a sample group description) followed by the number of unencrypted sample data Blocks indicated by the field `default_skip_byte_block` or `skip_byte_block` (if present in a sample group description).

If the last Block pattern in a Subsample is incomplete, the partial pattern SHALL be followed until truncated by the `BytesOfProtectedData` size, and any partial `crypt_byte_block` SHALL remain unencrypted.

The ‘cens’ scheme SHALL apply an IV to the first encrypted cipher block of each sample (same as ‘cenc’ and ‘cbc1’ schemes without pattern encryption).

The ‘cbcs’ scheme SHALL apply an IV to the first encrypted cipher block of each Subsample.

9.6.2 *Example of pattern encryption applied to a video NAL unit*

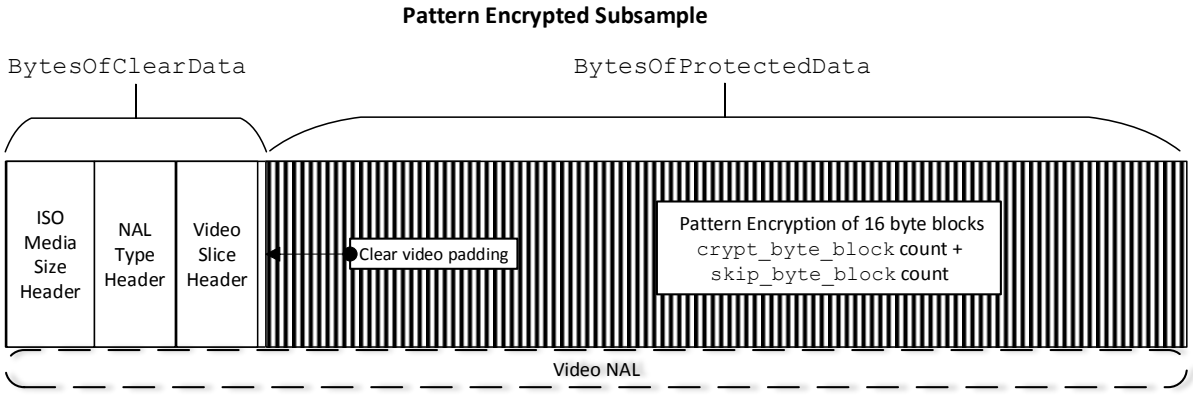

Figure 9 - Pattern encryption of a Subsample aligned with a video NAL Unit

Note: Pattern encryption is represented by vertical black and white lines representing a pattern of encrypted cipher blocks followed by clear Blocks. The pattern spans the protected range of a Subsample specified by `BytesOfProtectedData`, and approximately spans the video data following the slice header. Byte or Block alignment may require that the start of `BytesOfProtectedData` is not at the first bit of slice data, but some number of bits or bytes following that. Multiple Subsamples may be mapped to a single NAL and multiple clear NALs to a single Subsample in the `'cens'` scheme, but a single Subsample per VCL NAL may be required for the `'cbcs'` scheme.

10 Protection scheme definitions

10.1 *'cenc' AES-CTR scheme*

Support for the ‘cenc’ scheme is mandatory.

The `scheme_type` field of the scheme Type Box (`schm') SHALL be set to the four character code `'cenc'`.

Encrypted video tracks using NAL unit structured video conforming to ISO/IEC 14496-15 SHALL be protected using Subsample encryption specified in section 9.5, and SHALL NOT use

pattern encryption. As a result, the fields `crypt_byte_block` and `skip_byte_block` SHALL be 0.

Non-video encrypted tracks SHALL be protected using full-sample encryption as specified in section 9.4.

The version of the `TrackEncryptionBox ('tenc')` SHALL be 0.

Constant IVs SHALL NOT be used; `Per_Sample_IV_Size` SHALL NOT be 0, except for unencrypted sample groups.

For an `isProtected` flag of 0x1 where the `scheme_type` field of the scheme type box is `'cenc'` (i.e. AES-CTR), counter values SHALL be unique per `KID`.

`default_Per_Sample_IV_Size` and `Per_Sample_IV_Size` SHOULD be 8-bytes, and SHALL be a single value per track, or zero for unencrypted samples.

> Note1: If a `Per_Sample_IV_Size` of 8 is used, then the `InitializationVector` values for a given `KID` SHALL be unique for each sample if samples are less than 2^{64} cipher blocks in length and there are less than 2^{64} samples with unique 8-byte IVs in all tracks sharing the same `KID`.

The `BytesOfProtectedData` size SHOULD be a multiple of 16 bytes to avoid partial cipher blocks in Subsamples.

> Note 2: Support for `'cenc'` scheme is mandatory in the common encryption standard, and all Common Encryption implementations are required to decrypt the `'cenc'` scheme so that files using the `'cenc'` scheme can be processed by all decryptors of this standard.

10.2 *'cbc1' AES-CBC scheme*

Support for the `'cbc1'` scheme is optional.

The `scheme_type` field of the scheme Type Box (`schm') SHALL be set to `'cbc1'`.

Encrypted video tracks using NAL Structured Video conforming to ISO/IEC 14496-15 SHALL be protected using Subsample encryption specified in section 9.5, and SHALL NOT use pattern encryption. As a result, the fields `crypt_byte_block` and `skip_byte_block` SHALL be 0.

Other tracks SHALL be protected using full sample encryption as specified in section 9.4.

The version of the Track Encryption Box `('tenc')` SHALL be 0.

Constant IVs SHALL NOT be used; `Per_Sample_IV_Size` SHALL NOT be 0 except for unencrypted sample groups.

Per_Sample_IV_Size (as defined in section 9.2) MAY be 16 (which specifies 128-bit Initialization Vectors), and SHALL be a single value per track, or zero for unencrypted samples.

In Subsampled video tracks, the `BytesOfProtectedData` size SHALL be a multiple of 16 bytes to avoid partial cipher blocks in Subsamples.

Note: Support for 'cbc1' scheme is not mandatory in the common encryption standard, and implementations that process the 'cbc1' scheme are also required to process the 'cenc' scheme so that files using the 'cenc' scheme can be processed by all decryptors of this standard.

10.3 *'cens' AES-CTR subsample pattern encryption scheme*

Support for the 'cens' scheme is optional.

The scheme_type field of the scheme Type Box (`schm') SHALL be set to 'cens'.

The version of the Track Encryption Box ('tenc') SHALL be 1.

Non-video encrypted tracks SHALL be protected using a full sample encryption scheme such as 'cenc', as specified in section 9.4.

Encrypted video tracks using NAL Structured Video conforming to ISO/IEC 14496-15 SHALL be protected using Subsample encryption specified in section 9.5, and SHALL use pattern encryption specified in section 9.6. As a result, the fields crypt_byte_block and skip_byte_block SHALL NOT be 0.

Constant IVs SHALL NOT be used; Per_Sample_IV_Size SHALL NOT be 0 except for unencrypted sample groups.

default_Per_Sample_IV_Size and Per_Sample_IV_Size SHOULD be 8-bytes.

Note 1: If a Per_Sample_IV_Size of 8 is used, then the InitializationVector values for a given KID will be unique for each sample if samples are less than 2^{64} cipher blocks in length and there are less than 2^{64} samples with unique 8-byte IVs in all tracks sharing the same KID. IV storage is half of that required for 16-byte IVs.

The BytesOfProtectedData size SHALL be a multiple of 16 bytes to avoid partial cipher blocks in Subsamples.

For an isProtected flag of 0x1 where the scheme_type field of the scheme type box is 'cens' (i.e. AES-CTR), counter values SHALL be unique per KID.

Note 2: Support for 'cens' scheme is not mandatory in the common encryption standard, and implementations that process the 'cens' scheme are also required to process the 'cenc' scheme so that files using the 'cenc' scheme can be processed by all decryptors of this standard.

10.4 *'cbcs' AES-CBC subsample pattern encryption scheme*

10.4.1 *Definition*

Support for the 'cbcs' scheme is optional.

The scheme_type field of the scheme Type Box (`schm') SHALL be set to 'cbcs'.

The version of the Track Encryption Box ('tenc') SHALL be 1.

Encrypted video tracks using NAL Structured Video conforming to ISO/IEC 14496-15 SHALL be protected using Subsample encryption specified in section 9.5, and SHALL use pattern encryption as specified in section 9.6. As a result, the fields crypt_byte_block and skip_byte_block SHALL NOT be 0.

Constant IVs SHALL be used; `default_Per_Sample_IV_Size` and `Per_Sample_IV_Size`, SHALL be 0.

Non-video encrypted tracks SHALL be protected using a full sample encryption scheme such as `'cbc1'`, as specified in section 9.4.

Pattern Block length, i.e. `crypt_byte_block` + `skip_byte_block` SHOULD equal 10.

For all video NAL units, including in `'avc1'`, the slice header SHALL be unencrypted.

The first complete byte of video slice data (following the video slice header) SHALL begin a single Subsample protected byte range indicated by the start of `BytesOfProtectedData`, which extends to the end of the video NAL.

> Note 1: For AVC VCL NAL units, the encryption pattern starts at an offset rounded to the next byte after the slice header, i.e. on the first full byte of slice data. For HEVC, the encryption pattern starts after the `byte_alignment()` field that terminates the `slice_segment_header()`, i.e. on the first byte of slice data.

The Sample Auxiliary Information SHALL be present and SHALL identify protected ranges as Subsamples.

A decryptor can decrypt by parsing NAL units to locate video NALs by their type header, then parse their slice headers to locate the start of the encryption pattern, and parse their Part 15 NAL size headers to determine the end of the NAL and matching Subsample protected data range. It is therefore possible to decrypt a track using either (a) this algorithm, ignoring the Sample Auxiliary Information or (b) the Sample Auxiliary Information, ignoring this algorithm.

> Note 2: Support for `'cbcs'` scheme is not mandatory in the common encryption standard, and implementations that process the `'cbcs'` scheme are also required to process the `'cenc'` scheme so that files using the `'cenc'` scheme can be processed by all decryptors of this standard.

10.4.2 *'cbcs' AES-CBC mode pattern encryption scheme application (Informative)*

An encrypt:skip pattern of 1:9 (i.e., 10% partial encryption) is recommended. Even though the syntax allows many different encryption patterns, a pattern of ten Blocks is recommended. This means that the skipped Blocks will be (10-N). The number of encrypted cipher blocks N can span multiple contiguous 16-byte Blocks (e.g. 3 encrypted Blocks followed by 7 unencrypted Blocks would result in 30% partial encryption of the video data).

For example, to achieve 10% encryption, the first Block of the pattern is encrypted and the following nine Blocks are left unencrypted. The pattern is repeated every 160 bytes of the protected range, until the end of the range. If the protected range of the slice body is not a multiple of the pattern length (e.g. 160 bytes), then the pattern sequence applies to the included whole 16-byte Blocks, and a partial 16-byte Block that may remain where the pattern is terminated by the byte length of the range `BytesOfProtectedData`, is left unencrypted.

The encryption is restarted at every video NAL unit (and Subsample), i.e. if there is more than one video NAL unit in a sample they share the same Initialization Vector. A Constant IV stored in a sample group description will typically span all the Subsamples and samples in a movie fragment. This translates to each extent of `BytesOfProtectedData` in a Subsample restarting the encryption using the Constant IV.

11 XML representation of Common Encryption parameters

11.1 *Introduction*

In some cases, such as MPEG Dynamic Adaptive Streaming over HTTP (ISO/IEC 23009-1 DASH), it is useful to express the `default_KID` field from the Track Encryption Box (‘`tenc`’) and Protection System Specific Header Box (‘`pssh`’) in an XML manifest document accessible prior to availability of media. Then a media player application may read the XML `default_KID` value to determine if that key has been acquired, and may acquire a license using information in a `cenc:pssh` element in advance of media availability. To encourage consistency, an attribute and element to express the Common Encryption `default_KID` and `pssh` are specified in XML below. XML documents that allow extension attributes and elements SHOULD use the specified namespace, attribute, and element for consistency.

11.2 *Definition of the XML `cenc:default_KID` attribute and `cenc:pssh` element*

The `cenc:default_KID` attribute and `cenc:pssh` element SHALL be defined within the `"urn:mpeg:cenc:2013"` namespace by the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cenc="urn:mpeg:cenc:2013"
targetNamespace="urn:mpeg:cenc:2013" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<!-- KID is a 128-bit integer written in canonical UUID notation -->
	<xs:simpleType name="KeyIdType">
		<xs:restriction base="xs:string">
			<xs:pattern value="[A-Fa-f0-9]{8}-[A-Fa-f0-9]{4}-[A-Fa-f0-9]{4}-[A-Fa-f0-9]{4}-[A-Fa-f0-
			9]{12}"/>
		</xs:restriction>
	</xs:simpleType>
	<!-- space-delimited list of KIDs -->
	<xs:simpleType name="KeyIdListType">
		<xs:list itemType="cenc:KeyIdType"/>
	</xs:simpleType>
	<!-- attribute used within the DASH mp4protection descriptor -->
	<xs:attribute name="default_KID" type="cenc:KeyIdListType"/>
	<!-- element used within system specific UUID ContentProtection descriptors -->
	<xs:element name="pssh" type="xs:base64Binary"/>
</xs:schema>
```

Figure 10 – XML elements and attributes defined for Common Encryption

Documents SHOULD use namespace prefix: “`cenc:`”.

`default_KID` is a string in UUID format [1]. Any 128-bit number may be written using this hyphenated hexadecimal notation (even if it is not generated as a UUID), though use of mathematically unique UUIDs throughout the system is highly recommended to prevent number collisions between independent content producers.

`cenc:pssh` is a base64 encoded ‘`pssh`’ box with `SystemID` matching the `SystemID` of the containing Content Protection Descriptor element.

Note: Frequently used SystemID identifier values indexed to protection systems and their specifications may be found on the DASH Industry Forum web site: http://dashif.org/identifiers

11.3 *Use of the cenc:default_KID attribute and cenc:pssh element in DASH ContentProtection Descriptor elements*

11.3.1 *Introduction*

The MPEG DASH standard specifies Content Protection Descriptors for use in Media Presentation Descripton (MPD) XML documents [3]. The XML syntax of the DASH `ContentProtection Descriptor` element is specified in section 5.8 of DASH. The `Descriptor` complex type allows the addition of an attribute and/or element in a declared namespace different from the DASH namespace. This extension mechanism may be used to add the `cenc:default_KID` attribute and `cenc:pssh` element defined above to store information that is defined in this Common Encryption standard for storage in ISO Media files also in an MPD.

11.3.2 *Addition of cenc:default_KID attributes in DASH ContentProtection Descriptors*

The `default_KID` (the default Key Identifier field stored in the Track Encryption Box '`tenc`') identifies the default key used to encrypt samples in an encrypted ISO Base Media track. It may be used with the DASH specified "`mpeg:dash:mp4protection:2011`" content protection scheme to identify the protection scheme and `default_KID` used in an ISO Media file. The attribute `@value` is specified to contain the four character code of the ISO Media Scheme Type Box ('`schm`'). If a Common Encryption scheme such as '`cenc`' is contained in the `@value` attribute, the encryption scheme can be decrypted by any number of DRM key management systems that have access to the media key(s) and support that decryption scheme.

The `default_KID` field in '`tenc`' is a big endian array of 16 bytes, and is defined above to be stored in the `cenc:default_KID` attribute in DASH `ContentProtection` Descriptor elements as a UUID string.[1]

When a `ContentProtection` descriptor refers to several tracks, and these use different default Key Identifiers in different '`tenc`' boxes, the `cenc:default_KID` attribute SHALL store a space-delimited list of those different `default_KID` values.

The following is an example of `cenc:default_KID` contained in a `ContentProtection` Descriptor element:

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

Figure 11 – Example use of `ContentProtection Descriptor` with `cenc:default_KID` attribute

NOTE: For global uniqueness, a UUID [1] SHOULD be used for each unique `KID`/key value pair to prevent duplicate IDs for different keys by independent publishers. Publishers may use the same key value and `KID` in more than one track or file according to their rights management intentions.

With unique KIDs, a license request using the cenc:default_KID attribute value is sufficient to identify a DRM license containing the encryption key(s) used to encrypt the media; and that license can enable decryption and playback of the Components, Representations, or Adaptation Sets that the ContentProtection Descriptor element and default_KID describe.

Common Encrypted content described in an MPD SHALL include an mp4protection Content Protection Descriptor. A cenc:default_KID attribute SHOULD be contained in the mp4protection Content Protection Descriptor to identify the default_KID in the content, and need not be duplicated in each UUID Content Protection descriptor specific to each protection system.

11.3.3 *Addition of the cenc:pssh element in Protection System Specific UUID ContentProtection Descriptors*

DASH ContentProtection Descriptor elements in an MPD may use a urn:uuid schemeIdUri to identify a specific DRM system using the SystemID value used in the Protection System Specific Header Box ('pssh') defined in this specification. Each DRM system may also specify additional elements and attributes for its scheme and SystemID that can be used for license acquisition or other functions of the identified DRM system.

In addition to containing a Content Protection Descriptor with "urn:mpeg:dash:mp4protection:2011" to notify a DASH player that the content is encrypted, it is also recommended that an MPD include a ContentProtection Descriptor with schemeIdUri of urn:uuid for each SystemID that can provide a DRM license, and include sufficient information in the descriptor to enable license acquisition. License acquisition can be enabled by adding a cenc:pssh element to each urn:uuid scheme descriptor so that a player can find the same license acquisition information it would find in a 'pssh' box. An MPD will normally be processed before Media Segments are downloaded, so license acquisition information in an MPD will normally take precedence over information stored in 'pssh' boxes. Note that the cenc:pssh element contains a complete 'pssh' box, not just the contents of the box, so that parsing will be identical from the MPD or file.

11.3.4 *Example of two Content Protection Descriptors in an MPD*

The following example shows the use of two Content Protection Descriptors in a DASH MPD to identify an Adaptation Set encrypted with Common Encryption, and a key management system that may be used to decrypt the Adaptation Set. Multiple key management systems may be listed, each in its own Content Protection Descriptor, identified by the system's SystemID.

The first Content Protection Descriptor with schemeIdUri of urn:mpeg:dash:mp4protection:2011 indicates that the 'cenc' scheme (Common Encryption CTR mode, no pattern) was used to encrypt the referenced media, and the track's cenc:default_KID.

The second Content Protection Descriptor with schemeIdUri of urn:uuid type, indicates a hypothetical DRM system "Acme" with a SystemID represented as a UUID string, and a cenc:pssh element containing a base64 encoded 'pssh' box with that SystemID to provide license acquisition information.

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

<ContentProtection schemeIdUri="urn:uuid:d0ee2730-09b5-459f-8452-200e52b37567"
        value="Acme 2.0">

        <!-- base64 encoded 'pssh' box with this Acme SystemID -->
        <cenc:pssh>
           YmFzZTY0IGVuY29kZWQgY29udGVudHMgb2YgkXB
           zc2iSIGJveCB3aXRoIHRoaXMgU3lzdGVtSUQ=
        </cenc:pssh>
</ContentProtection>
```

Figure 12 – Example Content Protection Descriptors for scheme and DRM

A single `mpeg:dash:mp4protection:2011` Content Protection Descriptor may be sufficient for key management if license acquisition information is provided in the media (e.g. in '`pssh`' boxes in an initialization segment), in a player application, or by an Internet service that can resolve the `cenc:default_KID` value to a license.

Alternatively, the publisher of a DASH MPD may provide all the license acquisition information in MPD Content Protection Descriptors so that a DASH player may immediately acquire a license for a DRM system the player supports on receipt of the MPD by using the Content Protection Descriptor containing a `cenc:pssh` element for that DRM system. Early acquisition of licenses from an MPD is particularly useful for live streaming to avoid a large number of simultaneous license request on arrival of the first presentation segment on a large number of clients. DRM licenses are individualized, so cannot be cached. A large number of simultaneous license requests could result in errors, or delays in the start time of a live presentation for some viewers.