INTERNATIONAL STANDARD

ISO/IEC 23001-7

Fourth edition
2023-08

Information technology — MPEG systems technologies —

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

Technologies de l'information — Technologies des systèmes MPEG —

Partie 7: Cryptage commun des fichiers au format de fichier de médias de la base ISO

Reference number
ISO/IEC 23001-7:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: 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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

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

This fourth edition cancels and replaces the third edition (ISO/IEC 23001-7:2016), which has been technically revised. It also incorporates the Amendment ISO/IEC 23001-7:2016/Amd 1:2019.

The main changes are as follows:

Addition of:

— item encryption, which allows image items to use protection schemes defined for media tracks,

— support for multiple keys and IVs per protected sample,

— `'svel'` sensitive encryption scheme, a codec-specific encryption scheme for which the encrypted bitstream remains a valid decodable bitstream,

— improved selective encryption using sample groups

A list of all parts in the ISO/IEC 23001 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Common Encryption specifies encryption and key mapping methods that enable decryption of the same file using different Digital Rights Management (DRM) and key management systems. It defines 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 necessarily 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 `ProtectionSystemSpecificHeaderBox`. 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 document 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.

The additions in this fourth edition are listed in the Foreword.

Information technology — MPEG systems technologies —

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

1 Scope

This document specifies common encryption formats for use in any file format based on ISO/IEC 14496-12. File, item, 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 document does not define a DRM system.

The AES-128 symmetric block cipher is used 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 documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.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 network abstraction layer (NAL) unit structured video in the ISO base media file format*

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

ISO/IEC 23008-12, *Information technology — High efficiency coding and media delivery in heterogeneous — Part 12: Image File Format (HEIF)*

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

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

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

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
block
16-byte extent of sample data that may be encrypted or decrypted by AES-128 block cipher

Note 1 to entry: This is commonly known as a cipher block.

3.1.2
CENC SAI
sample auxiliary information associated with a sample and containing cryptographic information such as initialization vector or subsample information

Note 1 to entry: The sample auxiliary information is defined in ISO/IEC 14496-12, and is not part of the sample data.

3.1.3
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

3.1.4
initialization vector
8 or 16-byte value used in combination with a key and a block to create the first cipher block in a chain, and derive subsequent cipher blocks in a cipher block chain

3.1.5
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

3.1.6
NAL structured video
video streams composed of NAL Units

Note 1 to entry: The carriage of NAL Units is specified in ISO/IEC 14496-15

3.1.7
protection scheme
encryption algorithm and information identified by the `scheme_type` in a `SchemeTypeBox` in a `ProtectionSchemeInfoBox`

3.1.8
sample
media sample when the protection applies to media tracks, or the payload of an item when the protection applies to items

Note 1 to entry: Media sample as defined in ISO/IEC 14496-12.

Note 2 to entry: Payload of an item as defined in ISO/IEC 14496-12.

3.1.9
selective encryption
change in the `isProtected` value of samples associated with the same sample description entry

Note 1 to entry: This is achieved using `CencSampleEncryptionInformationGroupEntry` sample groups.

3.1.10
subsample
byte range within a sample consisting of an unprotected part immediately followed by a protected part

3.2 Abbreviated terms

AES	Advanced Encryption Standard
AES-CTR	AES Counter
AES-CBC	AES Cipher-Block Chaining
AVC	Advanced Video Coding as specified in ISO/IEC 14496-10
CENC	Common ENCryption
DRM	Digital Rights Management
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
UUID	Universally Unique Identifier

4 Protection schemes

4.1 Scheme type signalling

Scheme signalling shall conform to ISO/IEC 14496-12. For media tracks, as defined in ISO/IEC 14496-12, the sample entry is transformed and a `ProtectionSchemeInfoBox` is added to the standard sample entry in the `SampleDescriptionBox` to denote that a stream is protected. The `ProtectionSchemeInfoBox` shall contain a `SchemeTypeBox` so that the scheme is identifiable. The `SchemeTypeBox` shall obey the following additional constraints:

— The `scheme_type` field shall be set to a value equal to a four-character code defined in Clause 10.

— The `scheme_version` field shall be set to 0x00010000 (Major version 1, Minor version 0).

The `ProtectionSchemeInfoBox` shall also contain a `SchemeInformationBox`. For media tracks, the `SchemeInformationBox` shall contain a `TrackEncryptionBox`, describing the default encryption parameters for the track.

The schemes identify general classes of algorithms used to encrypt data. Implementations should not rely solely on `scheme_type` and `scheme_version` to determine if they can process a file and should also take into account:

— parameters associated with the scheme (e.g. the pattern in case of pattern encryption, or the size of initialization vectors),

— use of `CencSampleEncryptionInformationGroupEntry` and the associated parameters (e.g. change in `isProtected`, change in number and/or values of keys, change in size of initialization vectors),

— value of the field `aux_info_type_parameter` associated with CENC SAI,

— versions and flags of the `SampleEncryptionBox` box if present,

— versions of the `ProtectionSystemSpecificHeaderBox` and `TrackEncryptionBox`,

— support for, and values of versions and flags, of `ItemEncryptionBox` and `ItemAuxiliaryInformationBox`.

This document does not define brands nor profiles to restrict or recommend combinations of these parameters. Derived specifications may restrict some of these aspects.

4.2 Common encryption scheme types

Five protection schemes are specified in this edition of Common Encryption. Each scheme uses syntax and algorithms specified in Clause 5 to Clause 9, as constrained in Clause 10. They are the following:

a) '**cenc**' – AES-CTR mode full sample and video NAL subsample encryption; see 10.1.

b) '**cbc1**' – AES-CBC mode full sample and video NAL subsample encryption; see 10.2.

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

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

e) '**sve1**' – AES-CTR content sensitive encryption, as defined in Annex A.

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 DRM systems) a place to store their own data using a common mechanism. This data is contained in the `ProtectionSystemSpecificHeaderBox` described in 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 8.2 or in the `ItemEncryptionBox` described in 8.3.

— 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 IV (for some schemes), etc. This data is contained in a `SampleGroupDescriptionBox` that is referenced by a `SampleToGroupBox`. See Clause 6 for further details.

— CENC SAI, containing cryptographic information for individual media samples such as initialization vectors and subsample encryption data. CENC SAI data is sample auxiliary information as defined in ISO/IEC 14496-12. CENC SAI may reference bytes in a `SampleEncryptionBox`. See Clause 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, optional subsample information and, for each key involved in the sample protection, a `Per_Sample_IV_Size`, `KID`, and an optional `constant_IV`. This can be accomplished by using the default values in the `TrackEncryptionBox` (see 8.2), and optionally by specifying parameters by sample group. Encryption parameters specified in a sample group override the corresponding default parameter values for the samples in that group defined in the `TrackEncryptionBox`. Samples not mapped to any sample group use the default parameters established in the `TrackEncryptionBox`.

When specifying the parameters by sample group, samples are mapped using the `SampleToGroupBox` to sample group descriptions in the `SampleGroupDescriptionBox` of type `CencSampleEncryptionInformationGroupEntry` as defined below.

The syntax of `CencSampleEncryptionInformationGroupEntry` is the same for all track types (i.e., is independent from the handler type of the track).

For fragmented files, it may be necessary to store both the mappings and descriptions 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.

```
aligned(8) class CencSampleEncryptionInformationGroupEntry
   extends SampleGroupEntry( 'seig' )
{
   unsigned int(1)       multi_key_flag;
   unsigned int(7)       reserved = 0;
   unsigned int(4)       crypt_byte_block;
   unsigned int(4)       skip_byte_block;
   unsigned int(8)       isProtected;
   if (multi_key_flag == 1) {
      unsigned int(16)       key_count;
   } else {
      key_count = 1;
   }
   for (i=1; I <= key_count; i++) {
      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:

`multi_key_flag` indicates that the multiple keys version of the sample group description is used. If this flag is set, multiple keys will be described for this sample group description entry; otherwise, a single key is described for this sample group description entry.

`isProtected` is the flag which indicates the encryption state of the samples in the sample group. See the `isProtected` field in subclause 9.1 for further details.

`key_count` indicates the number of keys that may apply to a sample associated to this sample group description entry. It is not required for a sample associated with this sample group description entry to use all the keys described.

`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 subclause 9.1 for further details.

`KID` is the key identifier used for samples in the sample group. See the `KID` field in subclause 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 subclause 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 subclause 9.1 for further details.

`skip_byte_block` specifies the count of the unencrypted blocks in the protection pattern. See subclause 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 initialization vector information associated with it. Both initialization vector and subsample encryption information may be given in a CENC SAI referenced by `SampleAuxiliaryInformationSizesBox` and `SampleAuxiliaryInformationOffsetBox`, as defined in ISO/IEC 14496-12, with `aux_info_type` equal to the scheme and `aux_info_type_parameter` equal to 0 or 1.

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.

The format of the CENC SAI for `aux_info_type_parameter` equal to 0 or 1 shall be:

```
aligned(8) class CencSampleAuxiliaryDataFormat
{
   if (aux_info_type_parameter==0) {
      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 ]
      }
   } else if (aux_info_type_parameter == 1) {
      unsigned int(16) multi_IV_count;
      for (i=1; i <= multi _IV_count; i++) {
         unsigned int(16) multi_subindex_IV;
         unsigned int(Per_Sample_IV_Size*8) IV;
      }
      unsigned int(32) subsample_count;
      {
         unsigned int(16) multi_subindex;
         unsigned int(16) BytesOfClearData;
         unsigned int(32) BytesOfProtectedData;
      } [subsample_count]
   }
}
```

Where:

`sample_info_size` is the size of the CENC SAI for this sample.

`InitializationVector` is the initialization vector for the sample, unless a `constant_IV` is present in the `TrackEncryptionBox`. See the `InitializationVector` field in 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 9.1 for further details.

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

`multi_IV_count` indicates the number of entries in the initialization vector loop; this value may be zero when constant initialization vectors are used.

`multi_subindex_IV` indicates the index of the associated key entry, where value one is the first entry, in the associated list; if this data is read for the processing of a media sample, the associated list is the `'seig'` sample group description entry associated with this sample; otherwise (this data is read for the processing of an item), the associated list is the list of key definitions in the `'ienc'` item property of this item. The associated key entry shall have a `Per_Sample_IV_Size` different from 0, i.e. key entries using constant IV shall not be present in this loop. if this data is read for the processing of a media sample (i.e. not an item) and `aux_info_type_parameter` is set to 1, the associated `'seig'` sample group description entry shall have the `multi_key_flag` set to 1; Within a CENC SAI, there shall not be two `multi_subindex_IV` with the same value.

`IV` indicates the initialization vector to be used for the first block of protected data for the associated key entry.

`multi_subindex` indicates the index of the associated key entry, where value one is the first entry, in the associated list (see `multi_subindex_IV`) for the following run of encrypted data.

If subsample encryption is not used (the size of the CENC SAI equals `Per_Sample_IV_Size`), then the entire sample is protected (see 9.4 for further details). In this case, for a media track, all CENC SAI will have the same size and hence the `default_sample_info_size` of the `SampleAuxiliaryInformationSizesBox` will be equal to the `Per_Sample_IV_Size` of the initialization vector. If `Per_Sample_IV_Size` is also zero (because constant IVs are in use) then the CENC SAI is then empty and should be omitted.

NOTE Even if subsample encryption is used, the size of the CENC SAI can 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` can then be used.

7.2 Sample encryption information box for storage of sample auxiliary information

7.2.1 Sample encryption box — Definition

Box Type: `'senc'`

Container: `TrackFragmentBox` or `TrackBox`

Mandatory: No

Quantity: Zero or one

The `SampleEncryptionBox` provides an optional storage location for CENC SAI of samples in a track or track fragment.

The `SampleEncryptionBox` may be used when samples in a track or track fragment are protected. Storage of `SampleEncryptionBox` in a `TrackFragmentBox` makes the necessary CENC SAI 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.

When version 0 of `SampleEncryptionBox` is used, `sample_count` shall be equal to the number of samples in the track or track fragment. Consequently, version 0 shall not be used when selective encryption is in use.

When version other than 0 of `SampleEncryptionBox` is used, the `SampleEncryptionBox` only contains CENC SAI for samples having their `isProtected` flag different from 0x00, either through default or through an explicit `CencSampleEncryptionInformationGroupEntry` sample to group mapping. The CENC SAI entries are listed in the same order as samples in the track or track fragment. For example, the first entry will describe the CENC SAI of the first protected sample in the track or track fragment, regardless of the number of unprotected samples before this protected sample. Consequently, for version other than 0 of `SampleEncryptionBox`, there is no CENC SAI for a sample with `isProtected` different from 0x00, and the corresponding `SampleAuxiliaryInformationSizesBox` entry shall be 0.

NOTE This means that for version other than 0, the index of CENC SAI into this box for a given sample depends on the number of previous samples with non-zero `isProtected`; retrieving this information through the `SampleAuxiliaryInformationSizesBox` and `SampleAuxiliaryInformationOffsetsBox` can be easier.

Derived specifications may further restrict the content of the `SampleEncryptionBox`, for example by enforcing that all samples in a track fragment are either protected or unprotected.

The following flags are defined for `SampleEncryptionBox`:

`senc_use_subsamples`: flag mask is 0x000002. This flag shall not be set if the version is other than 0.

The variable `UseSubSampleEncryption` is set as follows:

— if the version of the `SampleEncryptionBox` is 0 and the flag `senc_use_subsamples` is set, `UseSubSampleEncryption` is set to 1,

— otherwise, if the version of the `SampleEncryptionBox` is not 0 and the sample description entry associated with the sample uses a protection scheme mandating usage of subsamples for the described media type, `UseSubSampleEncryption` is set to 1,

— otherwise, `UseSubSampleEncryption` is set to 0.

7.2.2 Syntax

```
aligned(8) class SampleEncryptionBox extends FullBox('senc', version, flags)
{
   unsigned int(32)  sample_count;
   {
      if (version==0) {
         unsigned int(Per_Sample_IV_Size*8) InitializationVector;
         if (UseSubSampleEncryption) {
            unsigned int(16) subsample_count;
            {
               unsigned int(16) BytesOfClearData;
               unsigned int(32) BytesOfProtectedData;
            } [subsample_count ]
         }
      } else if ((version==1) && isProtected){
         unsigned int(16) multi_IV_count;
         for (i=1; i <= multi _IV_count; i++) {
            unsigned int(16) multi_subindex_IV;
            unsigned int(Per_Sample_IV_Size*8) IV;
         }
         unsigned int(32) subsample_count;
         {
            unsigned int(16) multi_subindex;
            unsigned int(16) BytesOfClearData;
            unsigned int(32) BytesOfProtectedData;
         } [subsample_count]
      } else if ((version==2) && isProtected) {
         unsigned int(Per_Sample_IV_Size*8) InitializationVector;
         if (UseSubSampleEncryption) {
            unsigned int(16) subsample_count;
            {
               unsigned int(16) BytesOfClearData;
               unsigned int(32) BytesOfProtectedData;
            } [subsample_count ]
         }
      }
   }[ sample_count ]
}
```

7.2.3 Semantics

`sample_count` is the number of CENC SAI coded in the `SampleEncryptionBox`. For version 0, it shall be either 0 or the number of samples in the track or track fragment where the `SampleEncryptionBox` is contained. For versions other than 0, it shall be the number of protected samples in the track or track fragment where the `SampleEncryptionBox` is contained.

`InitializationVector` shall conform to the definition specified in subclause 9.2. Only one `Per_Sample_IV_Size` shall be used within a track, 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 subclause 9.2.

`subsample_count` shall conform to the definition specified in subclause 9.1.

`BytesOfClearData` shall conform to the definition specified in subclause 9.1.

`BytesOfProtectedData` shall conform to the definition specified in subclause 9.1.

`multi_IV_count`, `multi_subindex_IV`, `IV` and `multi_subindex` shall conform to the definition specified in Clause 7.

8 Box definitions

8.1 Protection system specific header box

8.1.1 Definition

Box Type: `'pssh'`

Container: `MovieBox`, `MovieFragmentBox` or `MetaBox` if no `MovieBox` and no `MovieFragmentBox`

Mandatory: No

Quantity: Zero or more

The `ProtectionSystemSpecificHeaderBox` contains information needed by a content protection system to play back the content. The data format is specified by the system identified by `SystemID`, and is considered opaque for the purposes of this document. For fragmented tracks, the collection of `ProtectionSystemSpecificHeaderBox`es from the initial `MovieBox`, 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 DRM systems, by including `ProtectionSystemSpecificHeaderBox`es 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:

— For media tracks, examine all `ProtectionSystemSpecificHeaderBox`es in the `MovieBox` and in the `MovieFragmentBox` associated with the sample (but not those in other `MovieFragmentBox`es). For image items, examine all `ProtectionSystemSpecificHeaderBox`es in the `MetaBox` or in the `MovieBox`

— 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 `TrackEncryptionBox` or `ItemEncryptionBox` or the `KID` field of the appropriate sample group description entry) with one of the `KID` values in the `ProtectionSystemSpecificHeaderBox`. 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.

The data in a `ProtectionSystemSpecificHeaderBox` is associated with samples based on a matching `KID` value in the `ProtectionSystemSpecificHeaderBox` and sample group description or default `TrackEncryptionBox` or `ItemEncryptionBox` 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 `ProtectionSystemSpecificHeaderBox`es 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: `SchemeInformationBox`

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 for a group of samples by a sample group description of type `'seig'`. 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 both the value of `default_isProtected` is 1 and `default_Per_Sample_IV_Size` is 0, then the `default_constant_IV_size` for all samples that use these settings shall be present. A Constant constant IV shall not be used with counter-mode encryption.

NOTE The version field of the `TrackEncryptionBox` is set to a value greater than zero when the pattern encryption defined in 9.6 is used, and to zero otherwise.

For sample entries protected using multiple keys per sample, per-sample protection values shall be indicated for all samples using `CencSampleEncryptionInformationGroupEntry` sample grouping. As a consequence, the values in the `TrackEncryptionBox` are ignored.

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_isProtected ==1 && 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` should 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 9.1 for further details.

`default_KID` is the default key identifier used for samples in this track. See the `KID` field in 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 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 subclause 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 9.1 for further details.

8.3 Item encryption box

8.3.1 Definition

Box Type: `'ienc'`

Container: `ItemPropertiesBox`

Mandatory: Yes for protected items using schemes defined in this document

Quantity: Zero or more, at most one associated with an item

Items as defined in ISO/IEC 14496-12 may be protected using the schemes defined in this document. In this case, such items shall have:

— an associated `ProtectionSchemeInfoBox` with no `OriginalFormatBox` and with a `SchemeTypeBox` as per subclause 4.1,

— an associated `ItemEncryptionBox` property marked as essential,

— an associated auxiliary item with an `aux_info_type` matching the protection scheme used, as defined in subclause 8.4.

The payload of that auxiliary item shall be exactly one `CencSampleAuxiliaryDataFormat` as defined in Clause 7.

The storage of properties for such items shall be compliant with ISO/IEC 23008-12.

The `ItemEncryptionBox` contains default values for the `isProtected` flag, `Per_Sample_IV_Size`, and `KID` for the item. 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 associated item. For files with only one key for all items, this property box allows the basic encryption parameters to be specified once for all items instead of being repeated per item.

Items sharing this property are always protected; consequently, the default value for `isProtected` field (see 9.1) is 1.

If the value `Per_Sample_IV_Size` is 0, then the `constant_IV_size` for all items that use these settings shall be present. A constant IV shall not be used with counter-mode encryption.

`ItemEncryptionBox` properties shall be marked as essential in `ItemPropertyAssociation`.

NOTE The version field of the `ItemEncryptionBox` is set to a value greater than zero when the pattern encryption defined in 9.6 is used and to zero otherwise.

8.3.2 Syntax

```
aligned(8) class ItemEncryptionBox extends ItemFullProperty('ienc', 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) crypt_byte_block;
      unsigned int(4) skip_byte_block;
   }
   unsigned int(8) num_keys;
   for (i=1; i<= num_keys; i++) {
      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;
      }
   }
}
```

8.3.3 Semantics

`version` shall be zero unless pattern-based encryption is in use, whereupon it shall be 1.

`crypt_byte_block` specifies the count of the encrypted blocks in the protection pattern, where each block is of size 16-bytes, for items associated with this key entry. See 9.1 for further details.

`skip_byte_block` specifies the count of the unencrypted blocks in the protection pattern for items associated with this key entry. See the `skip_byte_block` field in 9.1 for further details.

`num_keys` indicates the number of key definition entries.

`Per_Sample_IV_Size` is the initialization vector size in bytes for items associated with this key entry. See the `Per_Sample_IV_Size` field in 9.1 for further details.

`KID` is the key identifier used for items associated with this key entry. See the `KID` field in 9.1 for further details.

`constant_IV_size` is the size of the initialization vector for items associated with this key entry.

`constant_IV`, if present, is the initialization vector for items associated with this key entry. See the `constant_IV` field in 9.1 for further details.

8.4 Item auxiliary information box

8.4.1 Definition

Box Type: `'iaux'`

Container: `ItemPropertiesBox`

Mandatory (per item): Yes for protected items using schemes defined in this document

Quantity (per item): At most one associated with any item

Quantity: Zero or one

An item may have some auxiliary information associated, similarly to sample of a track having auxiliary sample information. This information is usually declared through item properties and associations; it can also be declared through auxiliary information items, to allow sharing the auxiliary info between items and samples through the extent construction method of the item.

Auxiliary information items shall be declared with an `item_type` value of `'auxi'`. Items shall indicate their auxiliary information items through an item reference of type `'auxr'` from the item to the auxiliary information item(s).

Auxiliary information item may have an `ItemAuxiliaryInformationBox` property associated, indicating the type of auxiliary information and its parameter type (`aux_info_type` and `aux_info_type_parameter`).

If no `ItemAuxiliaryInformationBox` is present for an auxiliary information item, then the implied value of `aux_info_type` is:

— for a protected item, the `scheme_type` included in the `ProtectionSchemeInfoBox` of the referring item,

— otherwise the `item_type` of the referring item.

The default value of the `aux_info_type_parameter` is 0. The `ItemAuxiliaryInformationBox` shall be present when the referring item is not protected and is not defined using version 2 or higher of `ItemInfoEntry`.

There shall be at most one auxiliary information item with a given `aux_info_type` and `aux_info_type_parameter` associated with an item.

When present, the `ItemAuxiliaryInformationBox` property shall be marked as essential in `ItemPropertyAssociation`.

8.4.2 Syntax

```
aligned(8) class ItemAuxiliaryInformationBox extends ItemFullProperty('iaux', version=0,
flags=0)
{
   unsigned int(32) aux_info_type;
   unsigned int(32) aux_info_type_parameter;
}
```

8.4.3 Semantics

`aux_info_type`: same semantics as sample auxiliary information in ISO/IEC 14496-12.

`aux_info_type_parameter`: same semantics as sample auxiliary information in ISO/IEC 14496-12.

9 Encryption of media data

9.1 Field semantics

Within the sample groups and CENC SAI 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 `SchemeTypeBox`, e.g. for `scheme_type` of `'cenc'`, the track default is AES-CTR mode encrypted using the `'cenc'` scheme);

0x02 – 0xFF: Reserved.

`Per_Sample_IV_Size` is the size in bytes of the `InitializationVector` field. The following are the 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. The following are the 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 have an `isProtected` flag value of 0x0, a `Per_Sample_IV_Size` value of 0x0, and the associated `KID` value is ignored and should be all zeros.

`InitializationVector` specifies the initialization vector needed for decryption of a sample. For an `isProtected` flag of 0x0, no initialization vectors are needed and the CENC SAI 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 the 128-bit IV value is made of `InitializationVector` value copied to bytes 0 to 7 and the bytes 8 to 15 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. `BytesOfClearData` may be zero if no clear bytes exist for this subsample.

`BytesOfProtectedData` specifies the number of bytes of protected data following the clear data. 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 unless a `TrackReferenceBox` is found for this track with one or more track references of type `'scal'` pointing to one or more tracks with sample entry code `'encv'`, in which case subclause 9.5.2.6 applies. 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, as defined in subclause 9.5.26, or whole-block full sample encryption, as defined in subclause 9.7, is in effect.

`skip_byte_block` shall be zero unless pattern-based encryption, as defined in subclause 9.5.2.6, or whole-block full sample encryption, as defined in subclause 9.7, is in effect.

When `skip_byte_block` is set to zero, for example when whole-block full sample encryption is used, the value of `crypt_byte_block` is ignored and should be set to zero.

9.2 Initialization vectors

The initialization vector (IV) values for each sample shall be either a constant IV, and located in the `TrackEncryptionBox`, `ItemEncryptionBox` or in a sample group description of type `'seig'`; or shall be signalled per sample, and be located in the CENC SAI of each protected sample. See subclause 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 AES-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 block values for AES-CTR mode.

— For 16-byte `Per_Sample_IV_Size`, initialization vectors for subsequent samples using AES-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 counter block for AES-CTR is unique. Even though the least significant bytes of the IV (bytes 8 to 15) are incremented as an unsigned 64-bit counter in AES-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.

AES-CBC mode initialization vectors need not be unique per sample or subsample, and may be generated randomly or sequentially, for example, 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.

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

Some schemes use a constant IV. 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 compared to a per-sample IV, which requires 8 bytes or 16 bytes per sample.

9.3 AES-CTR mode counter operation

Counter-mode schemes shall conform to AES [FIPS 197] and to AES Counter Mode [NIST 800-38A] with 128-bit keys.

AES-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). AES-CTR mode 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 counter block for AES-CTR shall be constructed from a per sample IV, and incremented as described below and in subclause 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 CENC SAI.

9.4 Full sample encryption

9.4.1 General

Full sample encryption may be used for all encrypted media types other than NAL structured video, which shall use subsample encryption.

9.4.2 Full sample encryption using AES-CTR mode

AES-CTR mode encryption shall use a unique IV per sample, and encrypt all bytes in the sample.

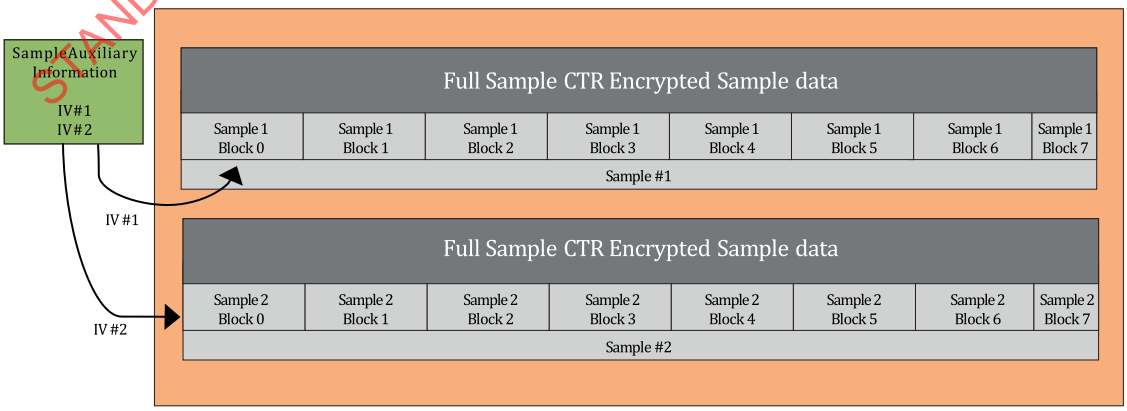

Figure 1 — Full sample encryption using AES-CTR mode

AES-CTR mode is a block cipher that can encrypt complete samples that are not a multiple of 16 bytes in size. In Figure 1, 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

Full sample AES-CBC mode shall conform to AES [FIPS 197] and to AES-CBC [NIST 800-38A] using 128-bit keys.

Each sample shall be encrypted as a continuous cipher block chain starting with an initialization vector, which shall be specified per sample. Per sample IVs shall be stored in CENC SAI as defined in 7.

Whole-block full sample encryption uses constant IVs, and is defined in 9.7.

Subsample Encryption of NAL structured video tracks is defined in 9.5.

Figure 2 shows an example encryption with two samples: AES-CBC mode requires all encrypted cipher blocks to be 16 bytes, and the schemes defined in this document leave partial blocks unencrypted (block 7 is shown as smaller than 16 bytes to illustrate that AES-CBC mode does not encrypt blocks smaller than 16 bytes to avoid adding padding that changes the sample size). Per sample IVs are applied at the start of each sample with `'cbc1'` full sample encryption.

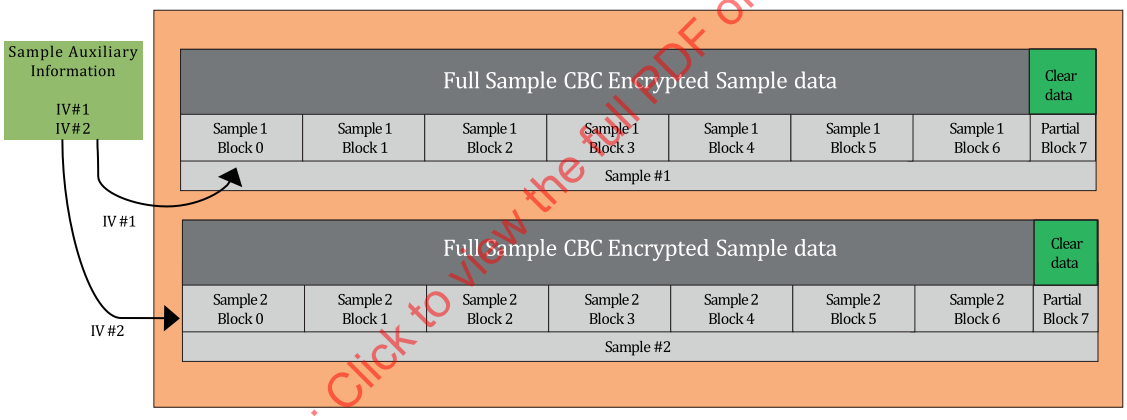

Figure 2 — Full Sample encryption using AES-CBC mode

9.5 Subsample encryption

9.5.1 Definition

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 (usually both are non-zero values). The total length of all of the subsamples shall be equal to the size of the sample itself.

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 counter for AES-CTR mode shall be incremented after each complete encrypted cipher block, ignoring subsample boundaries. Cipher block chaining of AES-CBC mode 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 AES-CTR mode shall be 16 bytes. A partial AES-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 AES-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. Cipher block chaining of AES-CBC 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 illustrates AES-CTR mode by 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.

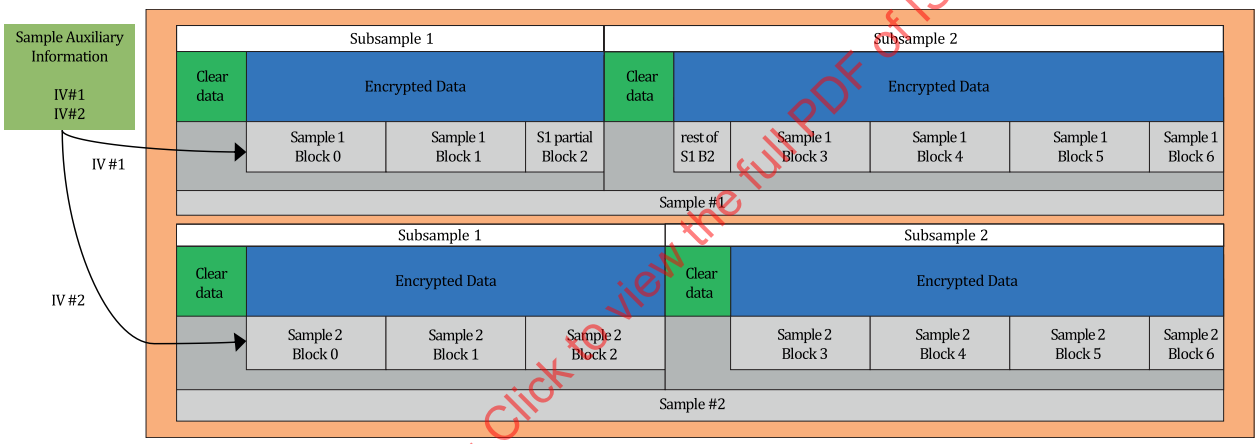

Figure 3 — AES-CTR mode subsample encryption using one IV per sample

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

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 AES-CTR mode can encrypt cipher blocks smaller than 16 bytes without adding padding that changes the sample size. Block 6 is unencrypted if AES-CBC mode is 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

This subclause describes the methods and reasons for using subsample encryption on NAL structured video samples.

Network Abstraction Layer (NAL) 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, as illustrated in Figure 4: 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 also contain a slice header.

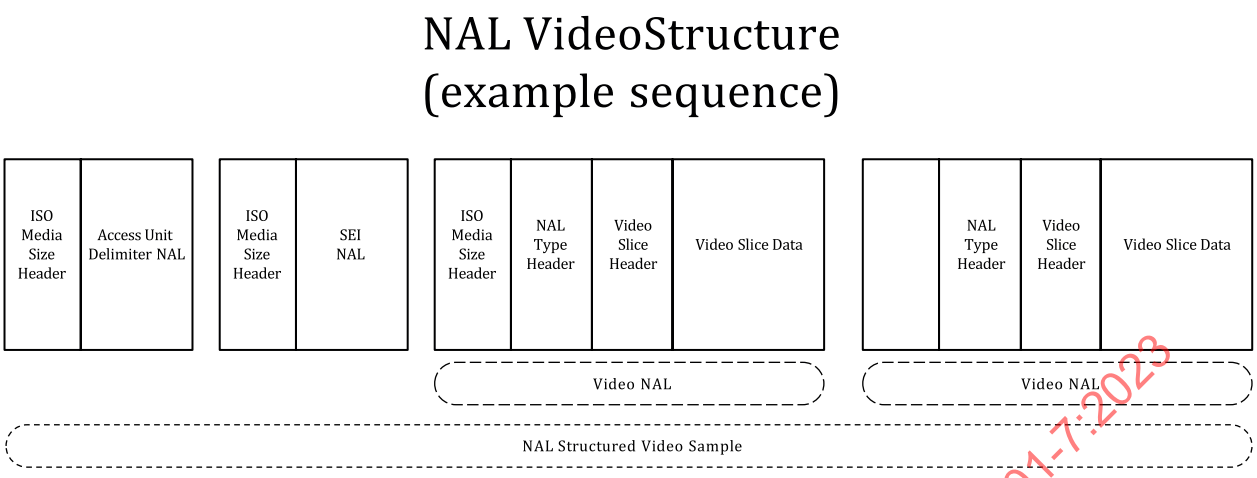

Figure 4 — NAL structured video ISO Media sample containing multiple NAL Units

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, as illustrated in Figure 5. 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.

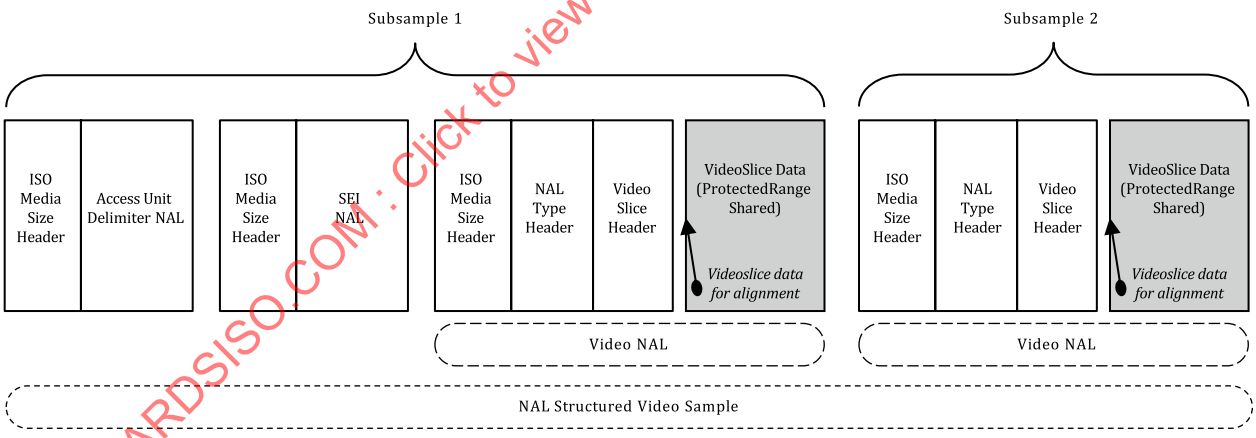

Figure 5 — Subsample encryption applied to NAL structured video

As illustrated in Figure 5, the Protected Range of a subsample does not always encrypt all video data. The start of the range can leave some video unencrypted to accomplish byte or 16 byte block alignment of the Protected Range. The Protected Range can 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 NAL size headers and lack decoding parameter NALs, such as Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) NALs (e.g. `'avc1'` sample entry format). Some decoders are designed to decode video elementary streams in byte stream format (ISO/IEC 14496-10:2022, Annex B) 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 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). Since NAL startcodes are always unencrypted in Common Encryption, any startcodes in encrypted data are invalid and can be ignored by processors. ISO Base Media file parsers ignore all startcodes. NALs before encryption and after decryption include emulation prevention as specified in the NAL structured video specifications, so startcodes may be reliably detected in decoders.

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 need to 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 CENC SAI. 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

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 document but were prohibited by application specifications. A recommendation to leave slice headers unencrypted for `'avc1'` allows possible legacy content with encrypted slice headers to remain conformant to this new edition. If new content encrypts slice headers, it will possibly 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, for example, 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 (e.g. 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.

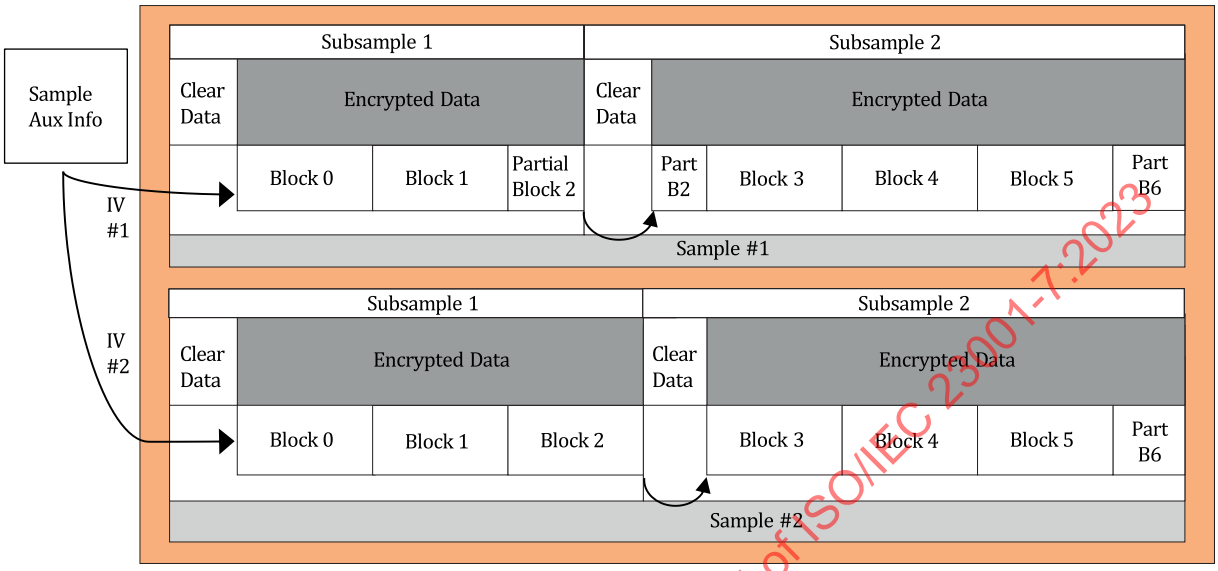

Figure 6 — Subsample encryption of video NALs using AES-CTR mode

AES-CTR mode is a block cipher that can encrypt partial cipher blocks. In Figure 6, 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 AES-CTR mode allows encryption of partial cipher blocks. Also, cipher block 2 of Sample 1 is continued in the next subsample to form a 16-bytes 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 document. The protection scheme `'cens'` can 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

AES-CBC mode `'cbc1'` scheme starts each sample with a per sample IV, then forms 16-bytes 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), as illustrated in Figure 7.

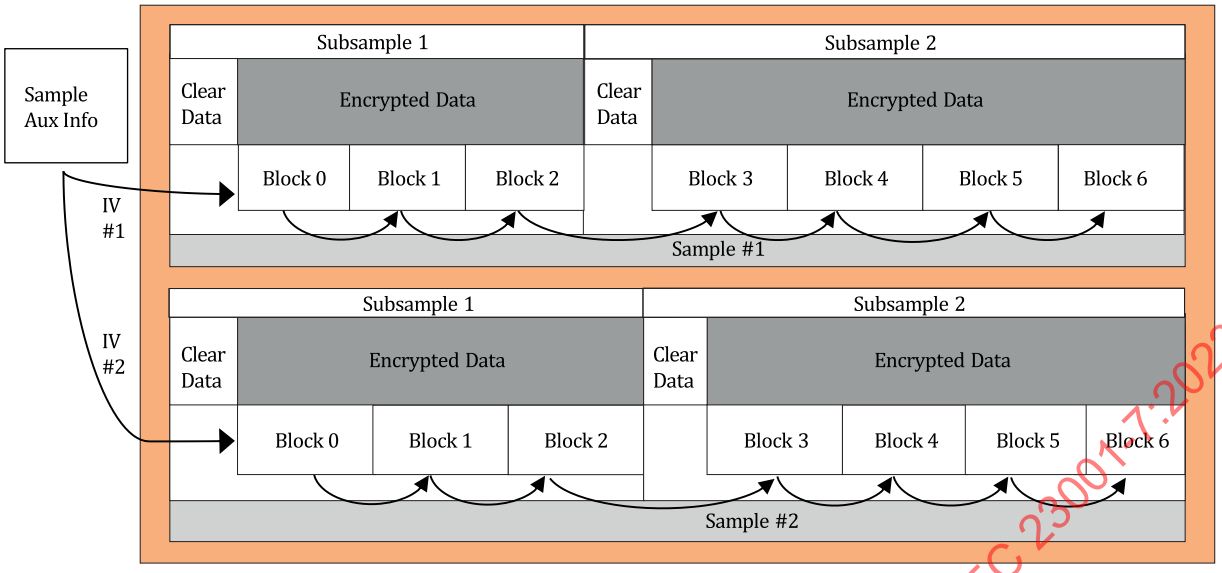

Figure 7 — Subsample encryption of video NALs using AES-CBC mode and `'cbc1'` scheme

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

AES-CBC mode `'cbcs'` scheme starts each subsample with a constant IV, then encrypts complete 16-bytes cipher blocks leaving any partial blocks unencrypted at the end of the subsample's `BytesOfProtectedData`, as illustrated in Figure 8. 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.

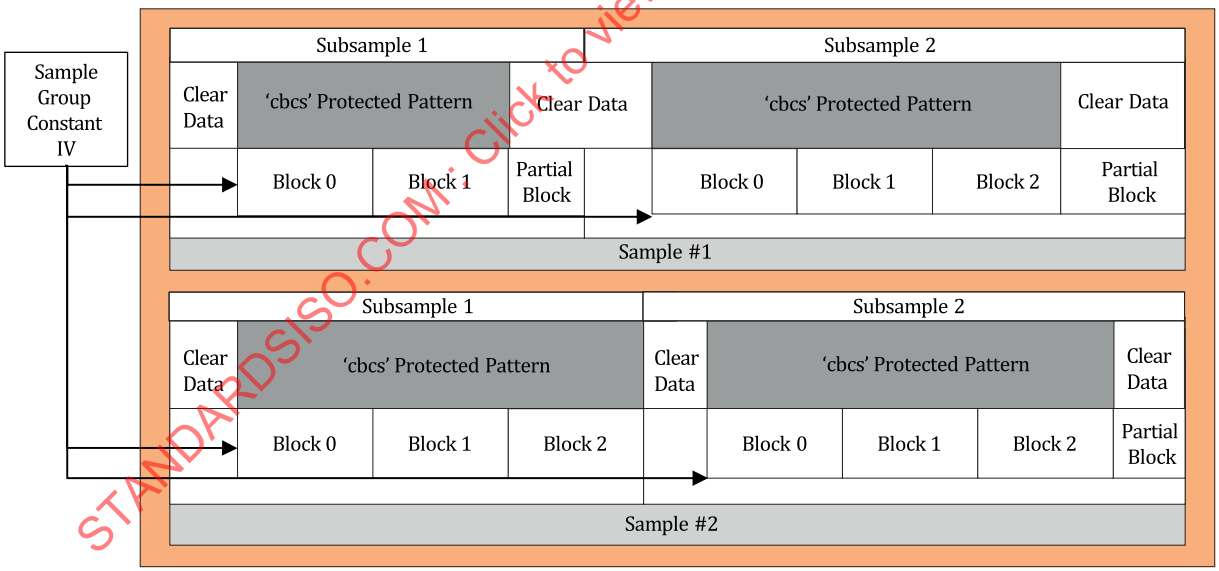

Figure 8 — Subsample encryption using AES-CBC mode and `'cbcs'` scheme

9.5.2.6 Subsample encryption applied to NAL structured video with extractors

In this clause, "protected extractor track " refers to a track with sample entry code `'encv'` for which the original format, as indicated by the `data_format` of the `OriginalFormatBox` in the `ProtectionSchemeInfoBox`, indicates possible presence of extractors, as defined in ISO/IEC 14496-15.

When a TrackReferenceBox is found in a protected extractor track which refers to one or more tracks with sample entry code 'encv', the following restrictions apply in addition to restrictions defined in subclause 9.5.2.2:

— Extraction process shall take place before the decryption process.

— Extractors NAL units shall not be encrypted.

— The following applies to the CENC SAI found in a protected extractor track:

  — When both values of BytesOfClearData and BytesOfProtectedData are 0, it is a placeholder for the CENC SAI of the NAL structured video samples that result from resolving the constructors within the extractor NAL unit as defined in ISO/IEC 14496-15:2019, A.7.

  — Otherwise, the values of BytesOfClearData and BytesOfProtectedData apply to the extracted sub_sample.

— The CENC SAI shall be present for each referenced track in 'tref' with sample entry code 'encv'.

— When an extractor NAL unit as defined in ISO/IEC 14496-15:2019, A.7 with an associated CENC SAI containing a value of 0 for both BytesOfClearData and BytesOfProtectedData points to a track with sample entry code 'encv':

  — Exactly one VCL NAL unit shall be encrypted in each subsample of the referenced track.

  — Let RefBytesOfClearData (resp. RefBytesOfProtectedData) be the BytesOfClearData (resp. BytesOfProtectedData) values from the CENC SAI associated with the subsample of which the byte range is extracted, then the CENC SAI for the resolved VCL NAL unit shall be modified as follows:

    — BytesOfClearData = RefBytesOfClearData SampleConstructor.sample_offset + InlineConstructor.length

    — BytesOfProtectedData = RefBytesOfProtectedData

    — The value SampleConstructor.sample_offset is the value of sample_offset in the SampleConstructor in the extractor that is processed.

    — The value InlineConstructor.length is the sum of the values in the length fields inside the InlineConstructor, if any are found in the extractor that is processed, otherwise 0.

— The resulting subsample information shall be conformant to CENC, and the decrypted version of every sample in the extractor track shall comply with the file and/or track brand and track description entry type.

9.6 Pattern encryption

9.6.1 Definition

Pattern encryption specifies a pattern of encrypted and clear ("skipped") 16 byte blocks over the protected range of a subsample.

NOTE Subsamples are primarily defined to protect 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 TrackEncryptionBox 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 or in ItemEncryptionBox) 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 or in ItemEncryptionBox).

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.

When AES-CTR mode is used, the IV shall apply to the first encrypted cipher block of each sample.

When AES-CBC mode is used, the IV shall apply to the first encrypted cipher block of each subsample.

9.6.2 Example of pattern encryption applied to a video NAL unit

Figure 9 illustrates usage of pattern encryption. 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.

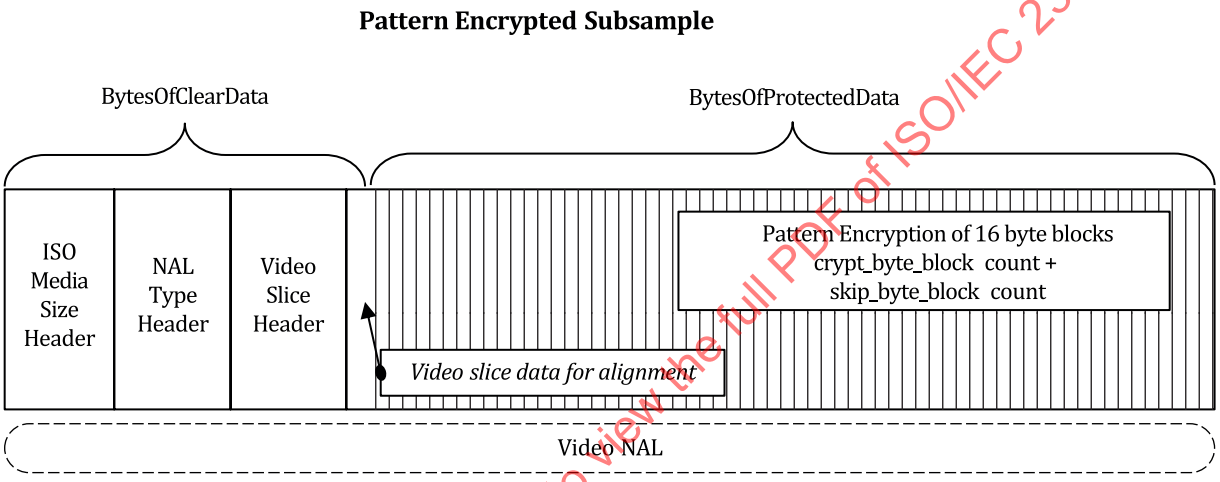

Figure 9 — Pattern encryption of a Subsample aligned with a video NAL Unit

9.7 Whole-block full sample encryption

In whole-block full sample encryption, the entire sample shall be protected. Every sample shall be encrypted starting at offset 0 (there is no unprotected preamble) up to the last 16-byte boundary, leaving any trailing 0-15 bytes in the clear. The constant IV shall be applied at the start of every sample.

9.8 Content sensitive encryption

9.8.1 Definition

Content sensitive encryption is a bitstream syntax aware scrambling scheme that modifies media bitstream in such a way that an unauthorized receiver (i.e. a standard decoder) can normally decode the ciphered stream, while the displayed content is made non-intelligible by the encryption. This scheme can be applied to media tracks, in which case they shall conform to ISO/IEC 14496-15, and to image items, in which case they shall conform to ISO/IEC 23008-12.

The coding format for tracks or image items using this scheme shall either comply to ISO/IEC 14496-10 or to ISO/IEC 23008-2.

The scheme operates in compressed domain based on entropic coder, by identifying the elements of the stream that can be ciphered without disrupting a standard decoding process. The bits to be encrypted are chosen with respect to the considered video standard to ensure full compatibility, achieved by selecting the bits (generally parts of code-words) for which each of the encrypted configuration

modifies the decoding process context but does not create de-synchronization nor lead to non-compliant bitstream.

The selected elements will depend on the video coding specification used:

— for AVC|H264 CAVLC, elements and bits shall be encrypted according to Annex A.1

— for AVC|H264 CABAC, elements and bits shall be encrypted according to Annex A.2

— for HEVC|H265, elements and bits shall be encrypted according to Annex A.3

NOTE For CABAC entropic coding (i.e. in Annex A.2 and Annex A.3), the bits considering as cipherable regarding to Content Sensitive encryption process are listed as bins since the considered codewords are first binarized before the bypassed Coded Engine.

9.8.2 Content sensitive encryption applied to a video NAL unit

In content sensitive encryption, 16 bytes block cypher cannot be used directly on the payload. Consequently, a video parser shall be used to locate bits to cipher/decipher. These bits, listed in Annex A, depend on the coding standard and potentially the entropic coding mode used.

Content sensitive encryption scheme shall use the AES-CTR mode for its cipher.

The encryption and decryption processes are performed with a simple XOR operation between the identified bits in the syntax and the cypher blocks, as shown in Figure 10.

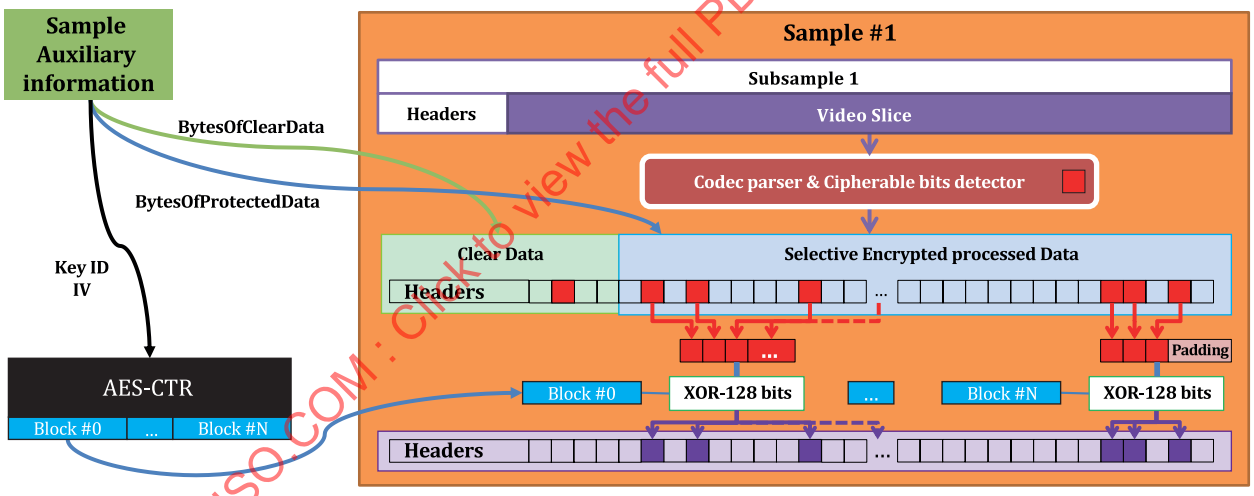

Figure 10 — Content Sensitive Encryption scheme

Samples protected with content sensitive scheme shall follow subsample encryption of NAL structured video tracks as defined in 9.5.2. The samples shall be divided into one or more contiguous subsamples. Each subsample consists of an unprotected part of `BytesOfClearData`, potentially 0, followed by a protected part. Bits contained in the clear part of the subsample shall not be selected for the decryption process. The first bit selectable for the encryption process in the range of protected data shall be XORed with the first bit of the first encrypted cipher block.

The counter for AES-CTR shall be incremented after each completely used encrypted cipher block (128 bits) or at the subsample boundaries; this implies that if not all bits were consumed on the last cipher block of a subsample, the block shall be considered as consumed and the counter shall be incremented before processing any subsequent subsample associated with that cipher.

10 Protection scheme definitions

10.1 'cenc' AES-CTR scheme

Support for the `'cenc'` scheme is mandatory. This scheme uses counter-mode protection as specified in subclause 9.3.

The `scheme_type` field of the `SchemeTypeBox` shall be set to the four-character code `'cenc'`.

Encrypted video tracks or items using NAL unit structured video conforming to ISO/IEC 14496-15 shall be protected using subsample encryption specified in subclause 9.5 and shall not use pattern encryption. As a result, the fields `crypt_byte_block` and `skip_byte_block` shall be 0.

When a single key applies to each sample, encrypted tracks or items not using NAL structured video shall be protected using full sample encryption as specified in subclause 9.4. Derived specifications may relax this constraint to allow usage of subsample encryption as specified in subclause 9.5, in which case pattern encryption shall not be used.

When multiple keys apply to samples, encrypted tracks or items not using NAL structured video shall be protected using subsample encryption as specified in subclause 9.5.

For tracks, the version of the `TrackEncryptionBox` shall be 0. For items, the version of the `ItemEncryptionBox` shall be 0.

Constant IVs shall not be used; `Per_Sample_IV_Size` shall not be 0, except for unencrypted sample groups.

Counter values shall be unique among all samples protected (`isProtected` flag of 0x1) by a given `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.

NOTE 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`.

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

10.2 'cbc1' AES-CBC scheme

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

The `scheme_type` field of the `SchemeTypeBox` shall be set to `'cbc1'`.

Encrypted video tracks or items using NAL structured video shall be protected using subsample encryption specified in 9.5, and shall not use pattern encryption. As a result, the fields `crypt_byte_block` and `skip_byte_block` shall be 0.

When a single key applies to each sample, other tracks or items shall be protected using full sample encryption as specified in subclause 9.4. Derived specifications may relax this constraint to allow usage of subsample encryption as specified in subclause 9.5, in which case pattern encryption shall not be used.

When multiple keys apply to samples, other tracks or items shall be protected using subsample encryption as specified in subclause 9.5, and shall not use pattern encryption.

For tracks, the version of the `TrackEncryptionBox` shall be 0. For items, the version of the `ItemEncryptionBox` 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 should be 16 (which specifies 128-bit initialization vectors), and shall be a single value per track, or zero for unencrypted samples.

In video tracks using subsample encryption, the BytesOfProtectedData size shall be a multiple of 16 bytes to avoid partial cipher blocks in subsamples.

10.3 'cens' AES-CTR subsample pattern encryption scheme

Support for the 'cens' scheme is optional. This scheme uses counter-mode protection as specified in subclause 9.3.

The scheme_type field of the SchemeTypeBox shall be set to 'cens'.

For tracks, the version of the TrackEncryptionBox shall be 1. For items, the version of the ItemEncryptionBox shall be 1.

Encrypted video tracks or items using NAL structured video shall be protected using subsample encryption specified in subclause 9.5, and shall use pattern encryption specified in subclause 9.6. As a result, crypt_byte_block shall not be 0 and skip_byte_block shall not be 0.

When a single key applies to each sample, encrypted tracks or items not using NAL structured video shall be protected using whole-block full-sample encryption as specified in subclause 9.7, and hence skip_byte_block shall be 0. Derived specifications may relax this constraint to allow usage of subsample encryption as specified in subclause 9.5, in which case pattern encryption as specified in subclause 9.5.2.6 shall be used.

When multiple keys apply to samples, encrypted tracks or items not using NAL structured video shall be protected using subsample encryption as specified in subclause 9.5 and shall use pattern encryption specified in subclause 9.5.2.6.

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 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.

Counter values shall be unique among all samples protected (isProtected flag of 0x1) by a given KID.

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 SchemeTypeBox shall be set to 'cbcs'.

For tracks, the version of the TrackEncryptionBox shall be 1. For items, the version of the ItemEncryptionBox shall be 1.

Encrypted video tracks or items using NAL structured video shall be protected using subsample encryption specified in subclause 9.5, and shall use pattern encryption as specified in subclause 9.5.2.6. As a result, crypt_byte_block shall not be 0 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.

The value of `default_constant_IV_size`, and `constant_IV_size` if present, shall be 16, indicating 128 bits IVs are used.

When a single key applies to each sample, encrypted tracks or items not using NAL structured video shall be protected using either:

— pattern encryption specified in subclause 9.5.2.6, and hence `skip_byte_block` shall not be 0,

— whole-block full-sample encryption as specified in subclause 9.7, and hence `skip_byte_block` shall be 0.

Derived specifications may relax this constraint to allow usage of subsample encryption as specified in subclause 9.5, in which case pattern encryption as specified in subclause 9.5.2.6 shall be used.

When multiple keys apply to samples, encrypted tracks or items not using NAL structured video shall be protected using subsample encryption as specified in subclause 9.5 and shall use pattern encryption specified in subclause 9.5.2.6.

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 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.

Unless it is empty, as noted in subclause 7.1, the CENC SAI 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 CENC SAI or (b) the CENC SAI, ignoring this algorithm.

10.4.2 `'cbcs'` AES-CBC mode pattern encryption scheme application

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 results 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. The use of `'cbcs'` on audio samples with `BytesOfProtectedData` less than the sample size allows identification of clear preambles on audio samples. It is recommended that the protected range of

audio data not skip blocks because it may be partially decodable, and because the small reduction in processing compared to video does not justify the additional complexity.

10.5 'sve1' AES-CTR sensitive encryption scheme

Support for the 'sve1' scheme is optional.

The scheme_type field of the SchemeTypeBox shall be set to the four-character code 'sve1'.

Tracks or items carrying media coding types for which no sensitive encryption mode is defined in Annex A shall not use this protection scheme.

Encrypted video tracks or items using NAL unit structured video conforming to ISO/IEC 14496-15 shall be protected using subsample encryption specified in subclause 9.5 and shall use sensitive encryption as specified in subclause 9.8. The NAL unit header should be left in the clear part of the subsample.

NOTE In AVC CAVLC the slice header can be encrypted but can still be parsed.

Pattern encryption shall not be used. As a result, the fields crypt_byte_block and skip_byte_block shall be 0 and the version of the TrackEncryptionBox shall be 0.

For tracks, the version of the TrackEncryptionBox shall be 0. For items, the version of the ItemEncryptionBox shall be 0.

Constant IVs shall not be used; Per_Sample_IV_Size shall not be 0, except for unencrypted sample groups.

11 XML representation of Common Encryption parameters

11.1 General

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 TrackEncryptionBox and ProtectionSystemSpecificHeaderBox 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 schema shown in Figure 11.

```
<?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">
   <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 11 — XML elements and attributes defined for Common Encryption

Documents should use namespace prefix: "`cenc:`".

`default_KID` is a 128-bit integer valid against the `KeyIdListType` given in Figure 12.

`cenc:pssh` is a base64 encoded `ProtectionSystemSpecificHeaderBox` 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 can be found on the DASH Industry Forum web site: https://dashif.org/identifiers

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

11.3.1 General

The MPEG DASH standard specifies content protection descriptors for use in Media Presentation Descripton (MPD) XML documents.[2] The XML syntax of the DASH `ContentProtection Descriptor` element is specified in Clause 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 `TrackEncryptionBox` 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 `SchemeTypeBox`. 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 `TrackEncryptionBox` can be stored in the `cenc:default_KID` attribute in DASH `ContentProtection` Descriptor elements as defined in subclause 11.2.

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

Figure 12 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 12 — Example use of ContentProtection Descriptor with cenc:default_KID attribute

For global uniqueness, a UUID 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 ProtectionSystemSpecificHeaderBox defined in this document. If present, schemeIdUri shall be compliant with IETF RFC 4122. 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 that is found in a ProtectionSystemSpecificHeaderBox. 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 ProtectionSystemSpecificHeaderBoxes. The cenc:pssh element contains a complete ProtectionSystemSpecificHeaderBox, 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 example in Figure 13 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 `ProtectionSystemSpecificHeaderBox` 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 ProtectionSystemSpecificHeaderBoxwith this Acme SystemID
-->
      <cenc:pssh>
         YmFzZTY0IGVuY29kZWQgY29udGVudHMgb2YgkXB
         zc2iSIGJveCB3aXRoIHRoaXMgU3lzdGVtSUQ=
      </cenc:pssh>
</ContentProtection>
```

Figure 13 — 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 `ProtectionSystemSpecificHeaderBox`es 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 can result in errors, or delays in the start time of a live presentation for some viewers.

Annex A
(normative)

Content sensitive encryption scheme

A.1 Code-words containing bits selected for encryption for MPEG-4/AVC CAVLC

A.1.1 General

This Annex gives the list of all VLC tables and code words that shall be encrypted by the content sensitive encryption process for AVC|H264 CAVLC mode (i.e. entropy_coding_mode is equal to 0 in ISO/IEC 14496-10 in Picture Parameter set RBSP syntax).

A.1.2 Slice QP Delta

In ISO/IEC 14496-10:2022, 7.3.3, **slice_qp_delta** is coded in se(v) (i.e. signed Exponential-Golomb code-word), and the value is given in Table A.1. But the value of slice_qp_delta shall be limited such that $SliceQP_Y$ is in the range of $-QpBdOffset_Y$ to +51, inclusive. So this code-word is eligible to be cyphered if only if, its absolute value is less than:

$$2^{Floor(Log2(Min(QpBdOffset_Y+26+pic\_init\_qp\_minus26\,,25-pic\_init\_qp\_minus26\,)))}$$

Then, the suffix bits (i.e. the bits following the first '1'), as illustrated by the highlighting (bold font) in Table A.1, shall be selected for Content Sensitive encryption.

Table A.1 — the Slice_QP_delta code-words table

Index	Slice_QP_Delta value	Code-word
0	0	1
1	1	01**0**
2	-1	01**1**
3	2	001**00**
4	-2	001**01**
5	3	001**10**
6	-3	001**11**
7	4	0001**000**
8	-4	0001**001**
...	....	**...**

A.1.3 Macroblock type

In ISO/IEC 14496-10:2022, 7.3.5, **mb_type** specifies the macroblock type is coded in ue(v) (i.e. unsigned Exponential-Golomb code-word) but the semantics of mb_type depend on the slice type:

— If slice_type is I, the bits highlighting (bold font) in Table A.2 , shall be selected for CONTENT SENSITIVE ENCRYPTION. Moreover, blocks located on the border of the video slice shall not be selected for Content Sensitive encryption.

Table A.2 — mb_type with I slice

Index	Symbol	Code-word
0	I_NxN	1
1	I_16x16_0_0_0	01**0**
2	I_16x16_1_0_0	01**1**
3	I_16x16_2_0_0	0010**0**
4	I_16x16_3_0_0	0010**1**
5	I_16x16_0_1_0	0011**0**
6	I_16x16_1_1_0	0011**1**
7	I_16x16_2_1_0	000100**0**
8	I_16x16_3_1_0	000100**1**
9	I_16x16_0_2_0	0001**010**
10	I_16x16_1_2_0	0001**011**
11	I_16x16_2_2_0	0001**100**
12	I_16x16_3_2_0	0001**101**
13	I_16x16_0_0_1	0001**110**
14	I_16x16_1_0_1	0001**111**
15	I_16x16_2_0_1	000010000
16	I_16x16_3_0_1	000010001
17	I_16x16_0_1_1	00001**0010**
18	I_16x16_1_1_1	00001**0011**
19	I_16x16_2_1_1	00001**0100**
20	I_16x16_3_1_1	00001**0101**
21	I_16x16_0_2_1	00001**0110**
22	I_16x16_1_2_1	00001**0111**
23	I_16x16_2_2_1	00001**1000**
24	I_16x16_3_2_1	00001**1001**
25	I_PCM	000011010

— If slice_type is P, the bits highlighting (bold font) in Table A.3 shall be selected for Content Sensitive encryption.

Table A.3 — mb_type for P slice

Index	Symbol	Code-word
0	P_L0_16x16	1
1	P_L0_L0_16x8	01**0**
2	P_L0_L0_8x16	01**1**
3	P_8x8	00100
4	P_8x8ref0	00101
...	...	...

A.1.4 PCM sample Luma and Chroma

In ISO/IEC 14496-10:2022, 7.3.5, **pcm_sample_luma** and **pcm_sample_chroma** are sample values in the raster scan within the macroblock and are coded Fixe-length coding. Therefore, all bits shall be selected for Content Sensitive encryption.