Microsoft Corporation

First Edition
2009-09-08

Revised
2010-03-09

Portable encoding of audio-video objects

The Protected Interoperable File Format (PIFF)

John A. Bocharov, Quintin Burns, Florin Folta, Kilroy Hughes, Anil Murching, Larry Olson, Patrik Schnell, John Simmons

10

Contents

Protected Interoperable File Format

A standard for delivery of audio-video content

1. Scope and Justification

This specification defines a standard multimedia file format for delivery and playback of multimedia content. It includes the audio-video container, stream encryption, and metadata to support content delivery for multi-bitrate adaptive streaming, optionally using a standard encryption scheme capable of supporting multiple DRM systems.

Although designed primarily for use in multi-bitrate adaptive streaming scenarios, this specification also has applicability for a wide range of other content delivery mechanisms, including:

- Second session or digital delivery of standard definition or portable media content from an optical disc to a PC or a portable device
- Internet download of multimedia content
- Broadcast download of multimedia content
- Progressive download and playback of multimedia content
- Side loading of multimedia content onto portable devices
- Storage, transfer and playback of multimedia content on flash memory media

The Protected Interoperable File Format (PIFF) is an ISO Base Media File Format brand [ISOFF]. The functional justifications for this brand are twofold:

1. Enabling DRM interoperability and extensibility
2. Providing a single encoding format appropriate for download, broadcast, streaming and multi-bitrate adaptive streaming.

Although envisioned primarily as a compatibility brand, content may be created with PIFF designated as the major brand.

This specification, combined with a specification for adaptive bitrate streaming [ABRSS], and industry accepted codec profiles for high, standard and portable definition output resolutions can provide the foundational basis for a truly interoperable online audio-video distribution standard.

2. References

The normative references are those industry standard specifications which PIFF references and/or builds upon. The informational references are for the most part internal Microsoft specifications.

2.1 Normative References

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

[ISOFF] *ISO 14496-12: Information technology — Coding of audio-visual objects – Part*

12: ISO Base Media File Format

[AVCFF] *ISO 14496-15: Information technology — Coding of audio-visual objects — Part 15: Advanced Video Coding (AVC) file format*

[H264] *ISO 14496-10: Information technology — Coding of audio-visual objects — Part 10: Advanced video coding*

[AAC] *ISO 14496-3: Information technology — Coding of audio-visual objects — Part 3: Audio*

[ISOTXT] *ISO 14496-17: Information technology – Coding of audio-visual objects – Part 17: Streaming text format*

[RFC2119] “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[ISOLAN] *ISO/IEC 639-3:2007 Codes for the representation of names of language – Part 3: Alpha-3 code for comprehensive coverage of languages*

[X667] *ITU-T Rec. X.667 (09/2004)* | *ISO/IEC 9834-8:2005, Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components*, http://www.itu.int/ITU-T/studygroups/com17/oid/X.667-E.pdf

2.2 Informational References

[MP4RA] Registration authority for code-points in the MP4 family, http://www.mp4ra.org

[ABRSS] *IIS Smooth Streaming Transport Protocol,* Microsoft published adaptive bitrate streaming protocol, September 8, 2009.

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119]. That is:

- “MUST”, “REQUIRED” or “SHALL”, mean that the definition is an absolute requirement of the specification.

- “MUST NOT” or “SHALL NOT” means that the definition is an absolute prohibition of the specification.

- “SHOULD” or “RECOMMENDED” mean that there may be valid reasons to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

- "SHOULD NOT" or "NOT RECOMMENDED" mean that there may be valid reasons when the particular behavior is acceptable, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

- "MAY" or "OPTIONAL" mean the item is truly optional.

3.2 Terminology

Adaptive bitrate streaming — Dynamically varying the video bit rate to provide continuous playback at the highest quality that available bandwidth and client rendering power will support.

AES — *The Advanced Encryption Standard (AES) is comprised of three block ciphers – AES-128, AES-192 and AES-256. See [AES].*

Annex B — *Annex B of [H264] specifies a framing format for H.264 so that it may be used by containers which do not provide the required framing found in [ISOFF]; e.g. MPEG-TS.*

Atom — *See Box.*

Box — Object-oriented building block defined by a unique type identifier and length (also called an Atom).

CBC mode — An AES encryption mode. In Cipher-block chaining (CBC), each block of plaintext is XORed with the previous ciphertext block before being encrypted.

Chunk — A contiguous set of samples for one track.

Container box — A box whose sole purpose is to contain and group a set of related boxes.

Cryptographically Random — Unpredictable, in that no polynomial-time algorithm, given any sequence of bits, can guess the succeeding K bits with probability greater than ½^K + 1/P(K) for any (positive) polynomial P and sufficiently large K.

CTR mode — An AES encryption mode. Counter (CTR) mode turns a block cipher into a stream cipher.

Hint Track — Special track which contains instructions for packaging one or more tracks into a streaming channel.

I Frame — An I frame is a full frame in an MPEG video stream.

IDR Frame — An IDR frame is a special kind of I frame defined for MPEG-4 AVC encoding.

ISO Base Media File — File format defined in reference [ISOFF].

IV — An initialization vector (IV) is a block of bits enabling multiple instances of a stream or block cipher to produce unique streams despite using the same encryption key.

Late Binding
Muxing an alternate track at the client device, rather than during the authoring process.

Media Data Box
Container box which holds actual media data for a presentation (‘mdat’).

Movie Box
A container box whose sub-boxes define the metadata for a presentation (‘moov’).

Movie Fragment
A movie fragment extends the presentation in time (see [ISOFF] for more information on the construction of fragmented movies).

NAL
The Network Abstraction Layer (NAL) is part of the H.264/AVC standard [AVCFF]. It specifies a “network friendly” video representation.

PIFF
Protected Interoperable File Format. An ISO Base Media File Format ‘code point’, defining a file format brand for which is DRM-interoperable and appropriate to local playback, streaming and adaptive streaming.

PPS
Picture parameter set. An active picture parameter set remains unchanged within a coded picture. See [H264].

Presentation
One or more motion sequences possibly combined with audio.

RTP
Real-time Transport Protocol. A packet format standard for delivering audio and video over the Internet.

Sample
In non-hint tracks, a sample is an individual frame of video, a time-contiguous series of video frames, or a time-contiguous compressed section of audio. In hint tracks, a sample defines the formation of one or more streaming packets. No two samples within a track may share the same time-stamp.

Sample Description
Structure defining the format of some number of samples in a track.

SPS
Sequence parameter set. An active sequence parameter set remains unchanged throughout a coded video sequence. See [H264].

Track
Collection of related samples in an ISO base media file.

Track Fragment
Within a movie fragment, there is a set of track fragments, zero or more per track (see [ISOFF] for more information on the construction of fragmented movies).

UUID
Universally Unique Identifier (UUID). Extensibility mechanism described in [ISOFF] and conforming to [X667].

3.3 Notation

This document uses a class based notation with inheritance (see also [ISOFF] and [ISOTXT]). The classes are consistently represented as structures on the disk and on the network as follows: the fields of a class appear in the disk structure in the same order they are specified, and all fields in a parent class appear before fields for derived classes.

For example, an object specified as:

```
aligned(8) class Parent (unsigned int(32) p1_value,
            ..., unsigned int(32) pN_value) {
      unsigned int(32) p1 = p1_value;
      ...
      unsigned int(32) pN = pN_value;
}

aligned(8) class Child (
      unsigned int(32) p1_value, ... , unsigned int(32) pN_value,
      unsigned int(32) c1_value, ... , unsigned int(32) cN_value)
      extends Parent (p1_value, ..., pN_value) {
   unsigned int(32) c1 = c1_value;
   ...
   unsigned int(32) cN = cN_value;
}
```

Maps to:

```
aligned(8) struct {
      unsigned int(32) p1 = p1_value;
      ...
      unsigned int(32) pN = pN_value;
      unsigned int(32) c1 = c1_value;
      ...
      unsigned int(32) cN = cN_value;
}
```

When a box contains other box(es) as children, child box(es) always appear after any explicitly specified fields, and can appear in any order (i.e. sibling boxes can always be re-ordered without breaking compliance to the specification).

4. Introduction

The principal PIFF enhancements to the ISO Base Media File Format specification are support for seamless switching of alternate bitrate tracks for multi-bitrate adaptive streaming, and support for multiple DRM technologies in a single container file.

- Support for seamless switching of alternate bitrate tracks is accomplished by using the fragmented movie structure [ISOFF] and constraining container box settings to accommodate adaptive bitrate streaming (see also [ABRSS]).

- Multiple DRM support is accomplished by defining a standard encryption method, and by creating three new “uuid” boxes – the Protection System Specific Header Box, the Track Encryption Box, and the Sample Encryption Box.

The standard encryption method is AES 128 bit in either CTR mode or CBC mode, with a specified method for setting the initialization vector. By standardizing the encryption algorithm in this way, the same file can be used by multiple DRM systems, and multiple DRM systems can grant access to the same file thereby enabling playback of a single video file on multiple DRM systems. The differences between DRM systems are reduced to how they acquire the decryption key, and how they represent the usage rights associated with the file.

The data objects used by the DRM specific methods for retrieving the decryption key and rights object or license associated with the file are stored in the Protection System Specific Header Box. Any number of these boxes MAY be contained in the Movie Box ('moov'), each corresponding to a different DRM system. The Boxes and DRM system are identified by a SystemID. The data objects used for retrieving the decryption key and rights object are stored in an opaque data object of variable size within the Protection System Specific Header Box.

In addition to the consumer benefit, there is a significant supply chain improvement which results from making the container and encryption mechanism common to all DRM systems. Encoding and encryption of the movie can be done prior to insertion of any protection system specific header boxes. Space for these boxes can be reserved by using the free space box, so that offsets are preserved. The protection system specific header boxes can then be added as a late provisioning step, and additional protection system specific header boxes can be added at a later date.

Decryption is initiated when a device determines that the file has been protected by a stream type of 'encv' (encrypted video) or 'enca' (encrypted audio) – as is described in the ISO Base Media File standard [ISOFF]. The ISO parser examines the Scheme Information box within the Protection Scheme Information Box and determines that the track is encrypted using the PIFF scheme. The parser then looks for a Protection System Specific Header box that corresponds to a DRM which it supports. It uses the opaque data in that box to accomplish everything required by the particular DRM system to obtain a decryption key, obtain rights objects or licenses, authenticate the content, authorize the playback system, etc.

Using the key it obtains and a key identifier in the SampleEncryptionBox, which is shared by all the DRM systems, it can then decrypt audio and video samples reference by the SampleEncryptionBox using the decryption algorithms defined in section 6 of this specification.

The PIFF specification defines support for late binding or late muxing of alternate audio and video content. This enables receivers which support legacy or emerging codecs to use late binding to decode those streams at playback, while not burdening devices without support of those codecs with the additional download cost. It also enables such consumer features as downloading new audio tracks – e.g. a director commentary – without re-authoring that new audio track with an already downloaded video track. The muxing can take place realtime at the client device, and the multiplicity of authored SKUs becomes unnecessary.

5. Protected Interoperable File Format (PIFF)

The PIFF specification is a code point on the ISO Base Media File Format container specification [ISOFF]. The ISO file format is widely implemented on PCs and devices and allows for flexibility and interoperability.

Table 1 shows the PIFF Box type, structure, nesting level and cross references. The extensions to the ISO standard are shaded gray. References are provided for the definition of all boxes. The highlighted boxes are additions (uuid) for the PIFF specification – The Sample Encryption Box and the Protection System Specific Header Box. The Track Encryption Box is not shown since it is part of the Protected Sample Entry within the Sample Description Box.

Table 1 Protected Interoperable File Format (PIFF) brand

NESTING LEVEL							
0	1	2	3	4	5	SRC	Description
ftyp						ISO 4.3	File type and compatibility

NESTING LEVEL						SRC	Description
0	1	2	3	4	5		
moov						ISO 8.2.1	container for all metadata
	mvhd					ISO 8.2.2	movie header
	uuid					5.3.1	Protection System Specific Header Box
	trak					ISO 8.3.1	container for individual track
		tkhd				ISO 8.3.2	track header
		tref				ISO 8.3.3	track reference container
		mdia				ISO 8.4.1	container for media information in a track
			mdhd			ISO 8.4.2	media header
			hdlr			ISO 8.4.3	declares the media handler type
			minf			ISO 8.4.4	media information container
				vmhd		ISO 8.4.5	video media header
				smhd		ISO 8.4.5	sound media header
				nmhd		ISO 8.4.5	Null media header, overall information, some tracks only.
				dinf		ISO 8.7.1	data information box
					dref	ISO 8.7.2	data reference box, declares source of media data in track
				stbl		ISO 8.7.3	Sample table box, container for the time/space map
					stsd	ISO 8.5.2	Sample descriptions (codec types, initialization, etc.)
					stts	ISO 8.6.1.2	decoding, time to sample
					ctts	ISO 8.6.1.3	Composition time to sample
					stsc	ISO 8.7.4	sample-to-chunk
					stsz	ISO 8.7.3	sample sizes
					stz2	ISO 8.7.3	compact sample sizes
					stco	ISO 8.7.5	chunk offset
					co64	ISO 8.7.5	64-bit chunk offset

NESTING LEVEL							
0	1	2	3	4	5	SRC	Description
	mvex					ISO 8.8.1	movie extends box
		mehd				ISO 8.8.2	Movie extends header
		trex				ISO 8.8.3	track extends defaults
moof						ISO 8.8.4	movie fragment
	mfhd					ISO 8.8.5	movie fragment header
	traf					ISO 8.8.6	track fragment
		tfhd				ISO 8.8.7	track fragment header
		trun				ISO 8.8.8	track fragment run box
		sdtp				ISO 8.6.4	independent and disposable samples
		uuid				5.3.2	Sample Encryption Box
mdat						ISO 8.8.1	media data container
free						ISO 8.1.2	free space
skip						ISO 8.1.2	free space
mfra						ISO 8.8.9	movie fragment random access
	tfra					ISO 8.8.10	track fragment random access
	mfro					ISO 8.8.11	movie fragment random access offset

5.1 PIFF File Structure

The PIFF File Structure consists of two top-level Boxes: the Movie Fragment (‘moof’) Box for metadata, and the Media Data (‘mdat’) Box for samples.

Time spans are specified integer multiples of an increment known as the *TimeScale* and specified in the high-level metadata for the file [ISOFF].

The disk format for media is a specific layout of the ISO Base Media file format, and the network transmission can be a contiguous set of bytes corresponding to a movie fragment, copied directly from the file.

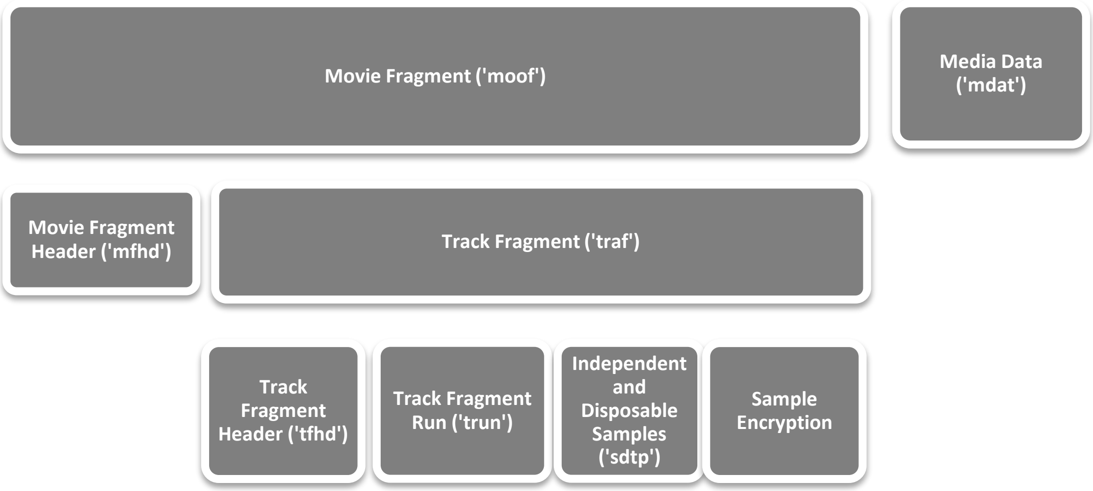
Movie Fragment ('moof')
Media Data ('mdat')
Movie Fragment Header ('mfhd')
Track Fragment ('traf')
Track Fragment Header ('tfhd')
Track Fragment Run ('trun')
Independent and Disposable Samples ('sdtp')
Sample Encryption

Figure 1 PIFF File Structure

The disk format used is based on the fragmented movie file format [ISOFF]. The organization of the disk file is as shown in Figure 2.

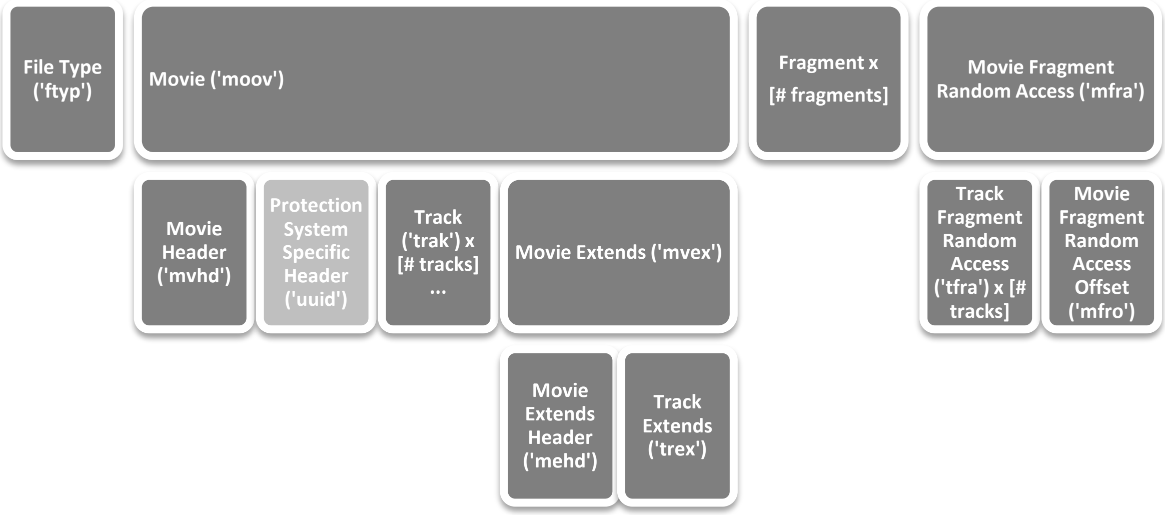
File Type ('ftyp')
Movie ('moov')
Fragment x [# fragments]
Movie Fragment Random Access ('mfra')
Movie Header ('mvhd')
Protection System Specific Header ('uuid')
Track ('trak') x [# tracks] …
Movie Extends ('mvex')
Track Fragment Random Access ('tfra') x [# tracks]
Movie Fragment Random Access Offset ('mfro')
Movie Extends Header ('mehd')
Track Extends ('trex')

Figure 2 PIFF disk file organization

5.2 PIFF Constraints on ISO Base Media File Format

The PIFF brand sets constraints on the use of the ISO Base Media File Format to assure that the encoded files can readily be used for multi-bitrate adaptive streaming.

Those constraints are captured in this section.

5.2.1 File Type box (‘ftyp’)

- The PIFF brand is 32 bits (4 octets) wide with the hexadecimal value `0x70696666` (‘`piff`’).

- Files conforming to the PIFF specification MAY use the PIFF brand as the major brand in the File Type box. At a minimum, the file MUST include the PIFF brand as a compatibility brand.

- Files compatible with the PIFF brand MUST include a File Type box designating a major brand.

- The minor version field is in network byte order (Big-endian). For files conforming to this version of the PIFF specification and using the PIFF brand as the major brand the minor version value MUST be 1 `0x00000001`. Note that if the PIFF brand is used as a compatibility brand, there are no requirements placed on the minor version field.

- For a file using the PIFF brand as the major brand, a file parser conforming to this version of the PIFF specification MUST support the minor version value given above

5.2.2 Movie Header (‘mvhd’)

- The following objects must have their default value:
  - rate
  - volume
  - matrix

5.2.3 Track Header Box (‘tkhd’)

- The following objects must have their default value:
  - Layer
  - alternate group
  - volume and matrix

- The Track_enabled flag SHOULD be set to 0 for chapter tracks and 1 otherwise.

- The Track_in_movie flag SHOULD be set to 0 for chapter tracks and 1 otherwise.

- The Track_in_preview flag SHOULD be set to 0 for chapter tracks and 1 otherwise.

- The width and height for a non-visual track MUST be 0.

5.2.4 Track Reference Box (‘tref’)

- This box SHOULD appear only for video tracks that have a corresponding chapter track (which is specified as a non-enabled text track), and/or a corresponding script stream track.

5.2.5 Media Header Box (‘mdhd’)

- The timescale is RECOMMENDED to be 10,000,000 (equivalent to increments of 100 ns). If a different value is used, then the timescale MUST be the same for all video tracks.

- If the language is unknown or the content is language-neutral, the [ISOLAN] code for undetermined (‘und’) SHOULD be coded into this field. The code ‘neu’, although not part of [ISOLAN], SHOULD be treated as a synonym of (‘und’) if encountered in this box.

5.2.6 Media Handler Box (‘hdlr’)

- Handler_type value of ‘hint’ SHOULD NOT be used. If it is included it MAY be ignored.

- The meta-box SHOULD NOT be used. If it is included it MAY be ignored.

5.2.7 Media Information Box (‘minf’)

- The sample tables SHOULD be empty, since sample data is specified on a per-fragment basis.

5.2.8 Video Media Header (‘vmhd’)

- The following objects MUST only have their default value:
  - version
  - graphicsmode
  - opcolor

5.2.9 Sound Media Header (‘smhd’)

- The following objects MUST only have their default value
  - version
  - balance

5.2.10 Null Media Header (‘nmhd’)

- The Null Media Header MUST be present if describing a text, marker, or script-stream track.

5.2.11 Data Reference Box (‘dref’)

- The data reference box MUST contain a single entry with the self-contained flag set.

5.2.12 Sample Description Box (‘stsd’)

- The sample description box MUST NOT contain entries of more than one type (audio, video, text, hint, and so on.)

- Hint tracks MAY be ignored.

- Sample entries for encrypted tracks (those containing any encrypted sample data) MUST encapsulate the existing sample entry with a protected sample entry such that:
  - The four-character-code in the sample entry is replaced to indicate the appropriate protection encapsulation (encv for video and enca for audio).
  - A Protection Scheme Information Box (‘sinf’) is included in the protected sample entry that has the original four-character-code of the sample entry in the OriginalFormatBox. The Protection Scheme Information Box (’sinf’) MUST conform to section 5.2.20.
  - The original sample entry data is preserved for the decoders use once the sample protection has been removed.

    This design follows the scheme defined in the *Support for Protected Streams* section (8.12) of [ISOFF].

5.2.13 Decoding Time to Sample Box (‘stts’)

- The Decoding Time to Sample SHOULD contain no entries.

5.2.14 Composition Time to Sample Box (‘ctts’)

- The Composite Time to Sample SHOULD contain no entries.

5.2.15 Track Extends Box (‘trex’)

- The file must be created such that each fragment stands on its own. Therefore, the default_* value SHOULD be initialized to 0, and MUST NOT be relied upon when constructing metadata for each fragment.

- The default_sample_description_index SHOULD be initialized to 1.

5.2.16 Track Fragment Box (‘traf’)

- The PIFF format uses one track per Movie Fragment. In other words, although ISO Media files have the capability of putting multiple tracks in a single Movie Fragment; each Fragment in the PIFF file format is a video fragment, or an audio fragment, etc.

5.2.17 Track Fragment Header (‘tfhd’)

- The PIFF format MUST use one track per fragment.

- The track_ID field MUST match the track_ID for the track in the Track Header Box.

- The base_data_offset field MUST be omitted.

- The sample_description_index contains an index of into the Sample Description table (‘stsd’) for this track. The Track Extends Box (‘trex’) specifies a default sample description index. This field is rarely needed – only when the track contains multiple sample types, and only for track fragments composed of samples that are not of the default sample type. In other cases, this field SHOULD be omitted by setting the sample-description-index-present field to 0.

- The default_sample_duration specifies the difference in decode time between each sample. This field SHOULD be set for video tracks with a fixed frame rate. When the default_sample_duration is used, samples typically vary in size, so a per-sample sample_size is set in the Track Run box (‘trun’), and the default_sample_size field is omitted.

- The default_sample_size specifies the size of each sample in bytes. This field SHOULD be set for audio tracks using a fixed-size-per-sample encoding. When the default_sample_size is used, samples typically vary in duration, so a per-sample sample_duration is set in the Track Run box (‘trun’), and the default_sample_size field is omitted.

- In the track fragment flags (tf_flags):
    - The sample-description-index-present flag SHOULD be set to 0 and the sample-description-index SHOULD be omitted.
    - The default-sample-duration-present flag MUST be set to 0 if the default_sample_duration is omitted.
    - The default-sample-size-present flag MUST be set to 0 if the default_sample_size is omitted.
    - The default-sample-flags-present flag MUST be set to 0 if and only if the default_sample_flags is omitted.
    - The base-data-offset-present and duration-is-empty flags MUST not be used.

5.2.18 Track Fragment Run Box ('trun')

- If this Track Fragment uses samples of varying size, the sample-size-present flag MUST be set and sample size MUST appear in the sample_size field for each sample.

- If this Track Fragment uses samples of varying duration, the sample-duration-present flag MUST be set and sample size MUST appear in the sample_duration field for each sample.

- The data_offset field MUST be set and its value MUST specify the offset from the beginning of the moof box. In other words, if only one trun is specified then the data_offset field MUST be the sum of the lengths of the moof box and all fields in the mdat box before the data field.

- The data-offset-present flag MUST be used.

- The first_sample_flags and the sample_flags are as defined for the Track Extends Box ('trex').
  - The first_sample_flags specifies the dependency and redundancy information for the first sample. For a video track, the first sample in a fragment MUST be an IDR frame, and its sample_depends_on flag MUST be set to 2.
  - The sample_flags specifies the dependency and redundancy information for each sample. For B-frames and P-frames, the sample_depends_on flag MUST be set to 1, and the sample_is_depended_on SHOULD be set to 1 if no B-frames depend on this sample (and 2 otherwise), but MAY be set to 0 if this information cannot be reliably determined.

- The sample_composition_time_offset specifies the offset between the decode time and composition time. See "8.15 Time to Sample Boxes" [ISOFF] for additional information.

5.2.19 Independent and Disposable Samples Box ('sdtp')

- Intentionally drop frames when the CPU can't keep up I-frames are indicated by setting their sample_depends_on flag to 2. For B-frames and P-frames, the sample_depends_on flag MUST be 1, and the sample_is_depended_on SHOULD be set to 1 if no B-frames depend on this sample (and 2 otherwise), but MAY be set to 0 if this information cannot be reliably determined.

5.2.20 Protection Scheme Information Box ('sinf')

- The IPMPInfoBox MAY be omitted, and if present, MAY be ignored.

- The SchemeTypeBox MUST be included and MUST comply with section 5.2.21.

  Per section 8.12 of [ISOFF], namely Support for Protected Streams, PIFF uses a Protection Scheme Information Box ('sinf') in place of the standard sample entry in the Sample Description Box to denote that a stream is encrypted. The Protection Scheme Info box contains a Scheme Type Box ('schm') so that the scheme is identifiable.

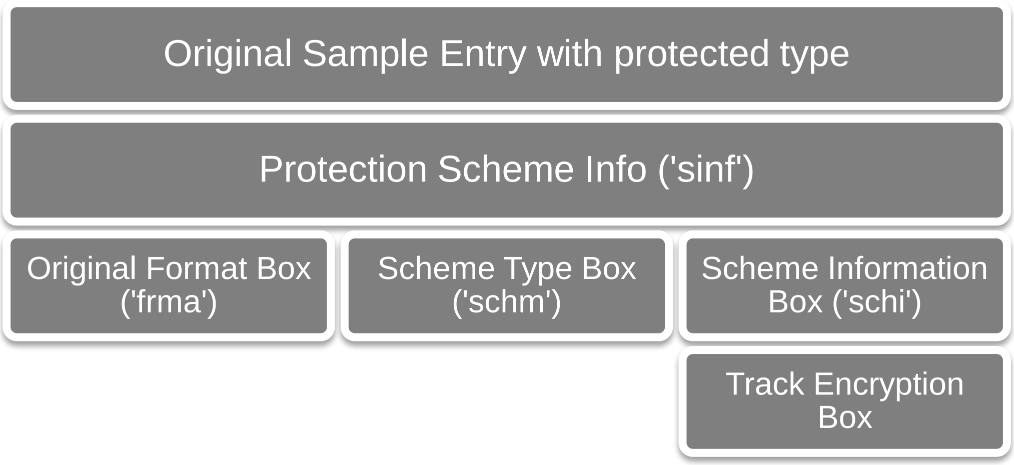
Original Sample Entry with protected type
Protection Scheme Info ('sinf')
Original Format Box ('frma')
Scheme Type Box ('schm')
Scheme Information Box ('schi')
Track Encryption Box

Figure 3 Placement of the Track Encryption Box in PIFF

5.2.21 Scheme Type Box ('schm')

- The PIFF scheme type is 32 bits (4 octets) wide with the hexadecimal value `0x70696666` (‘`piff`’).

- The scheme_version MUST be either `0x00010000` (Major Version 1, Minor version 0) or 0x00010001 (Major Version 1, Minor version 1). All tracks in a presentation must use the same scheme_version. The 0x00010001 version MUST be used if any SampleEncryptionBox sets the “Use SubSample Encryption” flag (0x2).

5.2.22 Scheme Information Box ('schi')

- If the Scheme Information Box is present it MUST contain a TrackEncryptionBox describing the default encryption parameters for the track.

- Any other boxes present SHOULD be ignored.

5.2.23 Sample-to-Chunk Box (‘stsc’)

- The entry_count MUST be zero.

5.2.24 Chunk Offset Boxes (‘stco’ or ‘co64’)

- The entry_count MUST be zero.

- One (and only one) of the two flavors of this box MUST be present as per [ISOFF].

5.2.25 Sample Size Boxes (‘stsz’ or ‘stz2’)

- The sample_count MUST be zero.

- One (and only one) of the two flavors of this box MUST be present per [ISOFF].

5.3 PIFF Extensions to ISO Base Media File Format

The following boxes are added to the Protected Interoperable File Format using the UUID extensibility mechanism to provide DRM interoperability.

5.3.1 Protection System Specific Header Box

Box Type	‘uuid’
Container	Movie (‘moov’)
Mandatory	No
Quantity	Any number

This box contains a header needed by a Content Protection System to play back the content. The header’s format is specified by the System it is targeted to, and is considered opaque from for the purposes of this specification.

The receiver then provides the data encapsulated in the Data field to the selected Content Protection System to enable playback. For license-based systems, the header information typically includes data such as the URL of the license server(s) used, key identifiers (KIDs) for which licenses MAY be obtained, and/or embedded licenses.

A single presentation MAY be constructed to be playable by multiple Content Protection Systems, by including one Protection System-Specific Header Box for each System supported. Receivers that process such presentations MUST match the SystemID field in this box to the SystemID(s) of the System(s) they support, and select one of the Protection System-Specific Header Boxes for a single playback session.

5.3.1.1 Syntax

```
aligned(8) class ProtectionSystemSpecificHeaderBox extends
FullBox(‘uuid’,
         extended_type=0xd08a4f18-10f3-4a82-b6c8-32d8aba183d3,
         version=0, flags=0)
  {
      unsigned int(8)[16]         SystemID;
      unsigned int(32)            DataSize;
      unsigned int(8)[DataSize]   Data;
  }
```

5.3.1.2 Semantics

- SystemID specifies a UUID that uniquely identifies the content protection system that this header belongs to.
- DataSize specifies the size in bytes of the Data member.
- Data holds the content protection system specific data.

5.3.1.3 Currently Recognized System Identifiers

Microsoft PlayReady uses the SystemID `9A04F079-9840-4286-AB92-E65BE0885F95`.

5.3.2 Sample Encryption Box

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

The Sample Encryption box contains the sample specific encryption data. It is used when the sample data in the track or fragment is encrypted. The box MUST be present for Track Fragment Boxes or Sample Table Boxes that contain or refer to sample data for tracks containing encrypted data. It SHOULD be omitted for unencrypted content.

5.3.2.1 Syntax

```
aligned(8) class SampleEncryptionBox extends FullBox(‘uuid’,
extended_type= 0xA2394F52-5A9B-4f14-A244-6C427C648DF4, version=0,
flags=0)
{
      if (flags & 0x000001)
      {
          unsigned int(24)    AlgorithmID;
          unsigned int(8)     IV_size;
          unsigned int(8)[16] KID;
      }
      unsigned int (32)      sample_count;
      {
          unsigned int(IV_size)  InitializationVector;

          if (flags & 0x000002)
          {
              unsigned int(16) NumberOfEntries;
              {
                  unsigned int(16) BytesOfClearData;
                  unsigned int(32) BytesOfEncryptedData;
              } [ NumberOfEntries]
          }

      }[ sample_count ]
}
```

5.3.2.2 Semantics

- `flags` is inherited from the FullBox structure. The SampleEncryptionBox currently supports the following Flags values:
  0x1 – Override TrackEncryptionBox parameters
  0x2 – Use SubSample Encryption

  If the Override TrackEncryptionBox parameters flag is set, then the SampleEncryptionBox specifies the AlgorithmID, IV_size, and KID parameters. If not present, then the default values from the TrackEncryptionBox SHOULD be used for this fragment.

  If the Use SubSample Encryption flag is set, then the track that this SampleEncryptionBox refers to must use the encryption algorithm described in Section 6.2. Further, this means that the SubSample mapping data follows each

InitializationVector. The SubSample mapping data consists of the number of subsamples for the sample followed by an array of values describing the number of bytes of clear data and the number of bytes of encrypted data for each subsample.

- `AlgorithmID` is the identifier of the encryption algorithm used to encrypt the track. The currently supported algorithms are:

0x0	– Not encrypted
0x1	– AES 128-bit in CTR mode
0x2	– AES 128-bit in CBC mode

  If the `AlgorithmID` is 0x0 (Not Encrypted) then the key identifier MUST be ignored and MUST be set to all zeros and the sample_count MUST be set to 0 (since no InitializationVectors are needed).

- `IV_size` is the size in bytes of the InitializationVector field. Supported values:

`8`	– Specifies 64-bit initialization vectors. Supported for AES-CTR.
`16`	– Specifies 128-bit initialization vectors. Supported for both AES-CTR and AES-CBC.

- `KID` is a key identifier that uniquely identifies the key needed to decrypt samples referred to by this sample encryption box.

- `sample_count` is the number of samples in this track fragment and also declares the number of rows in the following table (the table can have zero rows).

- `InitializationVector` specifies the initialization vector required for decryption of the sample.

- `NumberOfEntries` specifies the number of sub sample encryption entries present for this sample.

- `BytesOfClearData` specifies the number of bytes of clear data at the beginning of this sub sample encryption entry. Note that this value may be zero if no clear bytes exist for this entry.

- `BytesOfEncryptedData` specifies the number of bytes of encrypted data following the clear data. Note that this value may be zero if no encrypted bytes exist for this entry.

For an `AlgorithmID` of Not Encrypted, no initialization vectors are needed and this table SHOULD be omitted.

For an `AlgorithmID` of AES-CTR, if the `IV_size` field is 16 then the `InitializationVector` specifies the entire 128 bit IV value used as the counter value. If the `InitializationVector` field is 8, then its value is copied to bytes 0 to 7 of the 16 byte block passed to AES ECB and bytes 8 to 15 are set to zero. However the initial counter value is specified, bytes 8 to 15 are used as a simple block counter that is incremented for each block of the sample processed and is kept in network byte order.

Regardless of the length specified in the `IV_size` field, the initialization vectors for a given key MUST be unique for each sample in all Tracks. It is RECOMMENDED that the initial initialization vector be randomly generated and then incremented for each additional protected sample added. This provides entropy and ensures that the initialization vectors are unique.

For an AlgorithmID of AES-CBC, initialization vectors must by 16 bytes long and MUST be constructed such that the IV for the first sample in a fragment is randomly generated and subsequent samples within the same fragment use the last block of ciphertext from the previous sample as their IV. Note that the IV for each sample is still added to the SampleEncryptionBox (even though it can be retrieved from the previous sample) to facilitate random sample access.

The sub sample encryption entries SHALL NOT include an entry with a zero value in both the BytesOfClearData field and in the BytesOfEncryptedData field. Further, it is RECOMMENDED that the sub sample encryption entries be as compactly represented as possible. For example, instead two entries with {15 clear, 0 encrypted}, {17 clear, 500 encrypted} use one entry of {32 clear, 500 encrypted}.

See Section 6, Encryption of Track Level Data, for further details on how encryption is applied.

5.3.3 Track Encryption Box

Box Type	‘uuid’
Container	Scheme Information Box (‘schi’)
Mandatory	No
Quantity	Zero or one

The Scheme Information Box contains the content protection scheme applied to the track. The scheme information box MUST contain a compliant Track Encryption Box. It MAY contain other boxes. Any box not understood by a client SHOULD be ignored.

The Track Encryption box contains default values for the AlgorithmID, IV_size, and KID for the entire track. These values will be used as the encryption parameters for this track unless overridden by a SampleEncryptionBox with the Override TrackEncryptionBox parameters flag set. Since most files will only have one key per file, this box allows the basic encryption parameters to be specified once per track instead of being repeated in each fragment.

5.3.3.1 Syntax

```
aligned(8) class TrackEncryptionBox extends FullBox(‘uuid’,
extended_type=0x8974dbce-7be7-4c51-84f9-7148f9882554, version=0,
flags=0)
{
    unsigned int(24)    default_AlgorithmID;
    unsigned int(8)     default_IV_size;
    unsigned int(8)[16] default_KID;
}
```

5.3.3.2 Semantics

- `default_AlgorithmID` is the default encryption algorithm identifier used to encrypt the track. It can be overridden in any fragment by specifying the Override TrackEncryptionBox parameters flag in the Sample Encryption Box. See the AlgorithmID field in the Sample Encryption Box for further details.

- `default_IV_size` is the default Initialization Vector size in bytes. It can be overridden in any fragment by specifying the Override TrackEncryptionBox parameters flag in the Sample Encryption Box. See the IV_size field in the Sample Encryption Box for further details.

- `default_KID` is the default key identifier used for this track. It can be overridden in any fragment by specifying the `Override TrackEncryptionBox` parameters flag in the Sample Encryption Box. See the KID field in the Sample Encryption Box for further details.

5.4 Decryption flow of a protected PIFF file (Informative)

Here are the steps to process an encrypted PIFF file:

1. The parser opens the file and examines the streams to decrypt. In the Sample Description table it discovers that the stream is protected because it has a stream type of ‘encv’ or ‘enca’. If the player does not understand the protected track type, it SHOULD fail gracefully.

2. The parser examines the Scheme Type box within the Protection Scheme Information Box and determines that the track is encrypted via the specified scheme. It also extracts the original type of the stream (since it was replaced via ‘encv’ or ‘enca’).

3. The parser looks at the Scheme Information Box within the Protection Scheme Information Box to see if a TrackEncryptionBox containing default values for the KID, IV_size, and AlgorithmID is present.

4. The parser now knows to look for a Protection System Specific Header Box within the Movie Box that corresponds to a content protection system it supports.

5. The Protection System Specific Header Box is used to ensure that the license or licenses needed to decrypt the content are available on the client before playback begins. Thus the content protection system can search for licenses locally or acquire them as necessary before the playback pipeline is fully setup and initialized.

6. The parser uses the Sample Table metadata along with the Movie and Track fragment random access Boxes to figure out which sample to play at any given time in the presentation. Once a sample is located in a fragment, it will use the SampleEncryptionBox for that fragment along with any default values from the TrackEncryptionBox to get the correct key, initialization vector, and sub sample data (if necessary) for the sample. Either the fragment is not encrypted and can be passed directly to the decoder or the content will need to be decrypted using the proper encryption parameters. Normally a decryption transform component handles the work of figuring out if decryption is necessary, figuring out the necessary license for decryption, setting up the decryption context for the key, caching the decryption context for future use, applying sample protection, etc. All the media pipeline needs to do is provide the KID, sample data, subsample data (if necessary) and appropriate initialization vector to the decryption transform component for each sample in the fragment.

6. Encryption of Track Level Data

Encrypted track level data in PIFF files MUST use AES 128-bit encryption either in counter mode (AES-CTR) or cipher block chaining mode (AES-CBC). Encrypted AVC Video Tracks MUST follow the scheme outlined in section 6.2, which describes a NAL unit based encryption scheme to allow reformatting of the H.264 stream for decoders that do not understand AVC formatted streams natively. All other types of tracks MUST follow the scheme outlined in section 6.3, which describes a sample based encryption scheme.

6.1 IV Handling

Whether AES-CBC or AES-CTR mode is used, the initialization vector values for each sample are located in the SampleEncryptionBox of the MovieFragmentBox associated with the encrypted samples.

In order to minimize the number of counter value resets for hardware implementations of AES-CBC, the first initialization vector of the first sample in a fragment MUST be randomly generated using a Cryptographically Random, random number generator. Each subsequent sample in the fragment uses the last block of ciphertext from the previous sample as its IV. This is graphically represented in Figure 4.

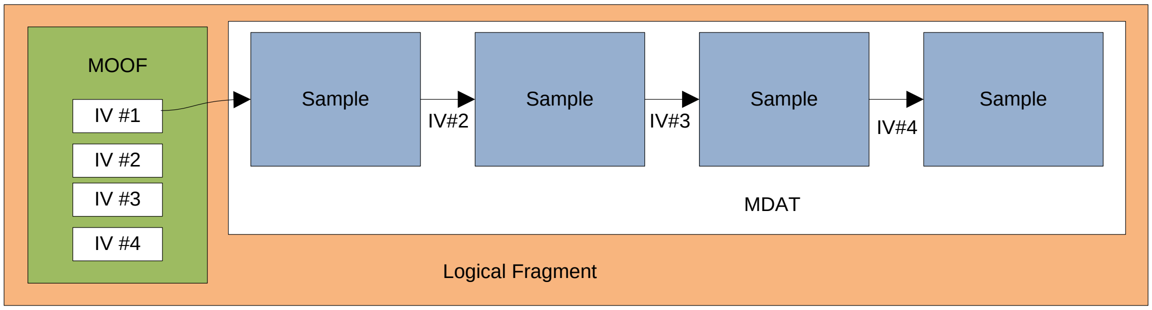

Figure 4 IV handling for AES-CBC

Note that the SampleEncryptionBox stores the IV for each sample even though it is the same as the last ciphertext block of the previous sample. This simplifies sample level random access.

In AES-CTR mode, the SampleEncryptionBox also stores the IV for each sample but there is no chaining relationship between the samples. The IV to sample relationship is represented in Figure 5.

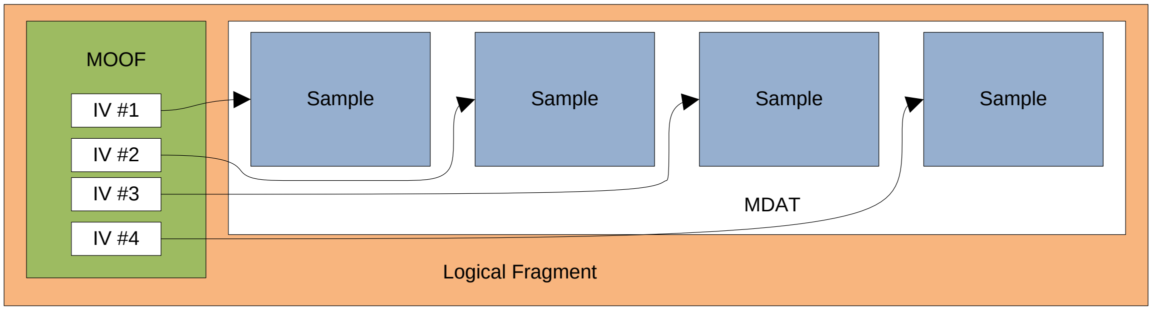

Figure 5 IV handling for AES-CTR

6.2 AVC Video Tracks – NAL Unit as the Basic Encryption Element

[H264] specifies the building blocks of the H.264 elementary stream, the Network Abstraction Layer (NAL) units. These units can be used to build H.264 elementary streams for various different applications. [AVCFF] specifies how the H.264 elementary stream data is to be laid out in an [ISOFF] base media file format container.

In the [AVCFF] layout, the container level samples are composed of multiple NAL units, each separated by a Length field that tells how long the NAL is.

An example of an unencrypted NAL layer is given in Figure 6.

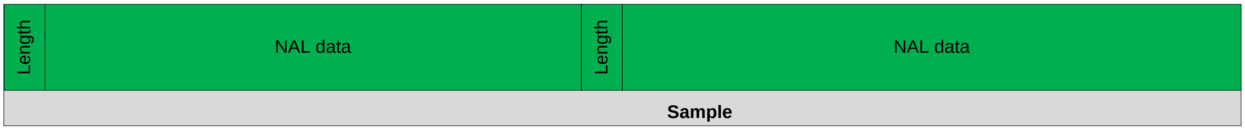

Figure 6 Example of a AVC Video Sample showing NALs

Not all decoders are designed to deal with an [AVCFF] or AVC formatted streams. Some decoders are designed to handle different H.264 elementary stream layouts; for example,[H264], Annex B. Further, it can be difficult to reformat the elementary stream in order to support transmitting the data over a network using protocols like RTP without first decrypting the samples.

The stored bitstream can be converted to Annex B bytestream format by adding startcodes and PPS/SPS NALs as “sequence headers”. It may be convenient to remove the NAL size headers during the decryption process since the size headers provide the necessary information to determine the size of the encrypted and clear stream segments, but are not compliant with Annex B streams at the decoder. It is also possible for the file parser/stream editor to convey the size information to the decryptor “out of band”, through APIs, rather than with temporary information in the stream. In order to facilitate stream reformatting before decryption, it is necessary to leave the NAL length fields in the clear as well as the `nal_unit_type` field (the first byte after the length). In addition:

1) The length field is a variable length field. It can be 1, 2, or 4 bytes long and is specified in the SampleEntry for the track (it can be found at `AVCSampleEntry.AVCConfigurationBox.` `AVCDecoderConfigurationRecord.lengthSizeMinusOne`)

2) There are multiple NAL units per sample, requiring multiple pieces of clear and encrypted data per sample.
3) When using AES-CBC mode, it only works on 16-byte boundaries and thus encrypting data that is not evenly divisible into 16-byte blocks requires special handling or padding.

Encrypted AVC Tracks MUST use the SubSample encryption feature of the SampleEncryptionBox to tell the decryption component exactly what parts of a sample are and are not encrypted. See section 5.3.2 for details on how to represent subsamples in the SampleEncryptionBox.

6.2.1 AES-CBC Mode

The application encrypting the content MUST choose a clear prefix length that leaves at least the `nalLength` and the `nal_unit_type` fields in the clear. The packager SHALL use the following “padding algorithm” to calculate the final value of the `BytesOfClearData` for the sub sample entry in the `SampleEncryptionBox`. The algorithm will increase the amount of clear data at the beginning of each NAL to the point that the remaining data is evenly divisible into 16-byte blocks:

```
static int GetNumberOfBytesInClear(int nalLengthSize,
                                   int nalLength,
                                   int clearPrefixLength)
{
    if ((nalLengthSize != 1)  &&
        (nalLengthSize != 2)  &&
        (nalLengthSize != 4))
    {
        throw new Exception("nalLengthSize must be 1, 2, or 4 bytes.");
    }

    if (nalLength <= 0)
    {
        throw new Exception("nalLength must be 1 or more bytes");
    }

    if (clearPrefixLength < nalLengthSize + 1)
    {
        throw new Exception("clearPrefixLength < nalLengthSize + 1");
    }

    int totalLengthOfNalData = nalLengthSize + nalLength;

    //
    //  Calculate the maximum number of bytes that could be encrypted
    //
    int maxNumberOfEncryptedBytes = totalLengthOfNalData -
                                    clearPrefixLength;

    //
    //  Use the modulus operator to figure out if the maximum number of
    //  bytes calculated above fit evenly into 16 byte encryption
    //  blocks or not.
    //
    int additionalClearData = maxNumberOfEncryptedBytes % 16;
    //
    //  The total number of bytes in the clear is calculated by adding
    //  the clear prefix length to the number of bytes of possible
    //  encrypted data that did not evenly fit into 16 byte blocks.
    //
    return clearPrefixLength + additionalClearData;
}
```

This algorithm ensures that the `clearPrefixLength` is large enough to leave the length field and `nal_unit_type` field in the clear to allow for stream reformatting. It also calculates any additional clear bytes to allow the rest of the NAL to be broken into 16-byte blocks for encryption.

Here is a diagram of what this scheme looks like:

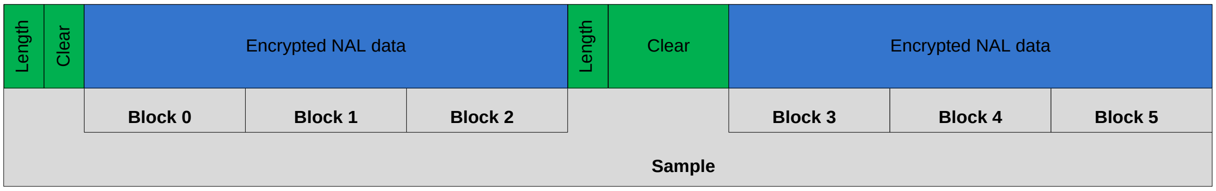

Figure 7 Example NAL Unit based encryption scheme for AES-CBC

Some NAL units are so small that the entire NAL will be in the clear. This is fine since no sensitive data exists in such a NAL that would need to be protected (i.e. the NAL is all stream metadata and contains no media data).

If we look at this scheme at the NAL level with the initialization vector relationships shown it looks like this:

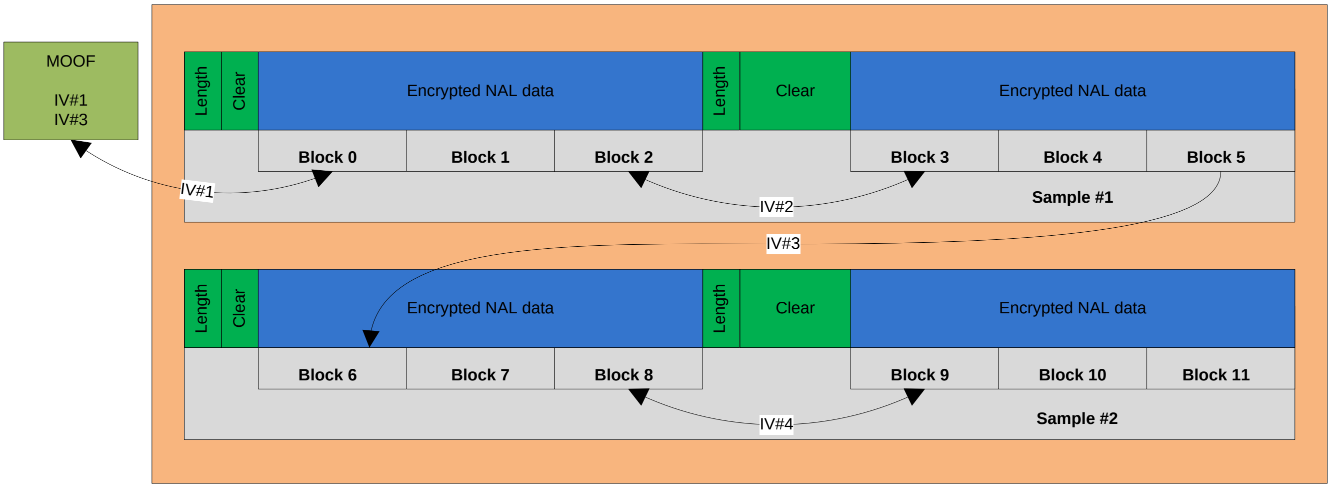

Figure 8 NAL Unit based encryption scheme for AES-CBC with IVs shown

Since the clear data (padding replacement) is in the front of the sample, the IV for the first NAL SHALL be the IV. The IV for the N-th NAL SHALL be the last ciphertext block of the previous NAL (N-1).

This generally means the last block of the previous NAL is the IV of the next encrypted NAL; however, it is possible that the previous NAL is a clear NAL (it was too small to be encrypted) and thus it cannot be assumed that the IV value is always the last block of the previous NAL.

6.2.2 AES-CTR Mode

AES-CTR mode can encrypt arbitrary length data without need for padding. The block counter SHALL start at 0 for the first block in the first NAL of the sample. It MUST be incremented for each block encrypted within the sample. The block counter is not incremented between NALs, meaning that the encrypted regions of the sample are essentially treated as if they were logically contiguous even though they are actually broken up by areas of clear data. If we look at this scheme at the NAL level it looks like this:

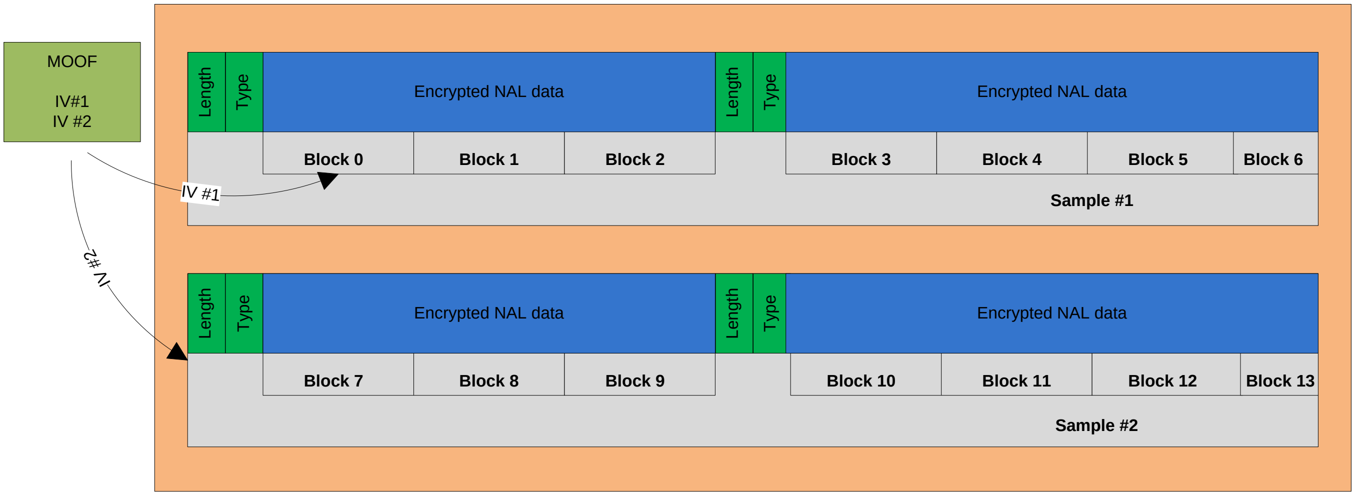

Figure 9 NAL Unit based encryption scheme for AES-CTR with IVs shown

Note that AES-CTR mode is a stream cipher and is therefore not block based. However, the blocks are shown to illustrate the underlying blocks used in generating the stream cipher (this is why Blocks 6 and 13 are not shown as full 16 byte blocks, the unused bytes of the stream cipher are discarded during the encryption or decryption process).

6.3 Non-AVC Encrypted Tracks – Sample as the Basic Encryption Element

For elementary streams other than AVC formatted H.264, the entire sample MUST be encrypted as a single encryption unit.

6.3.1 AES-CBC Mode

AES-CBC mode is a block cipher which means that it cannot handle arbitrary sized data without padding or special handling. Instead of implementing a padding algorithm, any data at the end of a sample that does not divide evenly into a block SHALL be left in the clear. Here is a diagram of what an encrypted sample looks like:

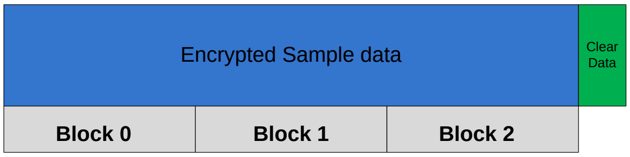

Figure 11 Sample based encryption scheme for AES-CBC

6.3.2 AES-CTR Mode

AES-CTR mode is a stream cipher which means that handles arbitrary sized data without padding or special handling. Here is a diagram of what an encrypted sample looks like:

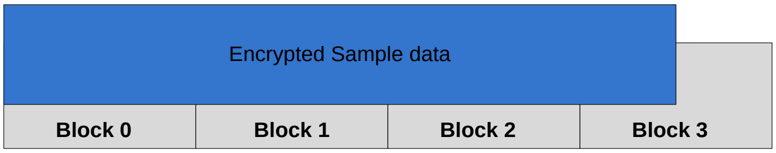

Figure 12 Sample based encryption scheme for AES-CTR

7. Formatting of UUID data

The PIFF specification uses the UUID extensibility mechanism described in [ISOFF] as well as including UUID data in several of the specified objects. All UUIDs written to the PIFF container MUST conform to [X667].

This specification calls for UUIDs to be written in the following format:

```
typedef struct {
unsigned32 time_low;
unsigned16 time_mid;
unsigned16 time_hi_and_version;
unsigned8 clock_seq_hi_and_reserved;
unsigned8 clock_seq_low;
byte node[6];
} uuid_t;
```

where the unsigned32 and unsigned16 values are written in network byte order (big endian).

Note that the PIFF specification follows the [ISOFF] convention of expressing UUIDs as a sixteen byte array even though the data is structured above (the `usertype` definition from the basic Box definition is an example, `unsigned int(8)[16] usertype = extended_type)`.

Further note that all UUID values given in this specification are displayed in little endian form.

8. Revision History

2009-09-08	1 rev 4 (B)	*Initial publication*
2010-01-15	1 rev 5 (A)	Section 5.2.1, constraints on use of the file type box; update the minor version required value to correspond to this version of the specification; clarify that if PIFF is not used as a major brand, there are no requirements on the minor version field.
		Section 5.2.21, constraints on use of the scheme type box; add option for scheme type (major version 1, minor version 1), used if the sample encryption box sets the SubSample encryption flag. Clarify that all tracks in a presentation must use the same protection scheme version.
		Section 5.3.2, the sample encryption box extension to the ISO Base Media File Format; add subsample encryption flag and its use.
		Section 5.4, informative decryption flow; add subsample encryption option to decryption flow.
		Section 6.2, AVC Video tracks and the NAL unit as the basic encryption element.
2010-02-02	1 rev 5 (B)	Section 5, fix section references in Table 1 to the ISO Base Media File Format specification.
		Section 5.3.1.3; fix System ID formatting to be in 8-4-4-4-12 format.
2010-03-01	1 rev 5 (F)	Section 5.2.15 Indicate that the default_sample_description_index SHOULD be initialized to 1.
		Section 5.2.17 Change use of the base_data_offset field to be ISO compliant - the base_data_offset field must be omitted and the base-data-offset-present field is no longer required to be set to 1.
		Section 5.2.18 Change use of the data_offset field to be ISO compliant - The data_offset field MUST be set and its value must specify the offset from the beginning of the moof box. The data-offset-present-flag MUST be used.
		Section 7 Clarify that all UUID values given in this specification are displayed in little endian format.