Proposed Standard
Date: 2004-04-12
2004-08-04 – updated to RP210v9 RP224v7 proposed UL

AS-DCP Track File Essence Encryption

Application Specification for Digital Cinema Packaging (AS-DCP)

Document type: Standard
Document subtype:
Document stage: Draft
Document language: English

Contents

AS-DCP Track File Essence Encryption

Application Specification for Digital Cinema Packaging (AS-DCP)

1 Scope

This document defines the syntax of encrypted AS-DCP track files and specifies a matching reference decryption model[1]. It uses the AES cipher algorithm for essence encryption and, optionally, the HMAC-SHA1 algorithm for essence integrity.

This document assumes that the cryptographic keys necessary to decrypt and verify the integrity of encrypted AS-DCP track files will be available upon demand. More precisely, it does not specify the fashion with which cryptographic keys and usage rights are managed across d-cinema distribution and exhibition environments. In addition, this document does not address, but does not preclude, the use of watermarking, fingerprinting or other security techniques to provide additional protection. Finally, while some of the concepts and structures included in this document may prove useful in other applications, the intent is not to define a generic MXF encryption framework.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

1. SMPTE EG42-2004. *For Television – MXF Descriptive Metadata*

2. National Institute of Standards and Technology (December 1, 2001). *Recommendation for Block Cipher Modes of Operation Methods and Techniques (SP 800-38A)* [WWW document]. URL http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

3. IETF 2898 (September 2000). *PKCS #5: Password-Based Cryptography Specification – Version 2.0* [WWW document]. URL http://www.ietf.org/rfc/rfc2898.txt

4. IETF 2104 (February 1997). *HMAC: Keyed-Hashing for Message Authentication* [WWW document]. URL http://ietf.org/rfc/rfc2104.txt

5. National Institute of Standards and Technology (November 26, 2001). *Advanced Encryption Standard (AES)* [WWW document]. URL http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

6. SMPTE, Proposed 000X, 2004. *For Cinema – AS-DCP Picture and Sound Track File*

7. SMPTE, 377M-2004. *For Television – The Material Exchange File Format Specification.*

[1] This document reflects the Track File requirements documented by DC28.20 and fits within the overall DC28.20 AS-DCP framework.

3 Overview

This specification defines the encryption of the sensitive essence information contained in AS-DCP Track Files [6] using the Advanced Encryption Standard (AES) [2] cipher algorithm. As an option, it also allows the integrity of the same essence to be verified using the HMAC-SHA1 algorithm. More specifically this specification allows any individual track contained within a plaintext AS-DCP Track File to be encrypted using a single cryptographic key. The resulting encrypted track file is extremely similar to a plaintext AS-DCP track file, which is itself a constrained version of the MXF OP-ATOM operational pattern[2]. It differs in the following three areas.

First, the Essence Container Label associated with the plaintext track is replaced by a universal Encrypted Essence Container Label. The replacement Label signals the presence of encrypted essence and allows non-compliant implementations to fail gracefully. The Encrypted Essence Container is discussed in Section 4.

Second, cryptographic information associated with the encrypted track as a whole is inserted in the MXF header metadata as a Cryptographic Framework. The Cryptographic Framework contains a link to the single cryptographic key used to encrypt the essence track. It also lists the algorithms necessary to process the encrypted essence and contains the original Essence Container Label. The latter allows implementations to determine the nature of the plaintext essence without further processing. The Cryptographic Framework is discussed in Sections 5 and 5.1.

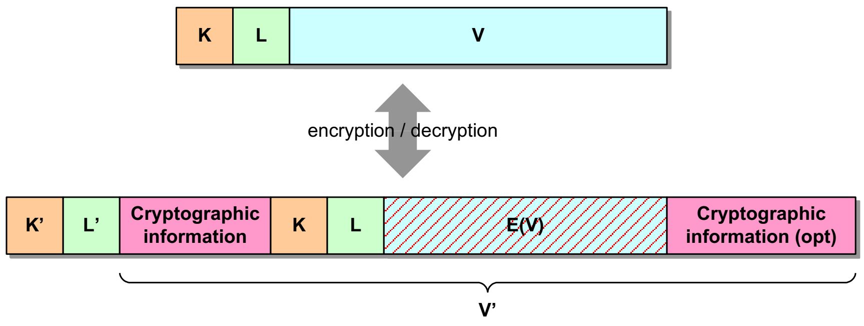

Figure 1. Correspondence between Source and Encrypted Triplets. Red hatching depicts the encrypted portion of the Encrypted Triplet; other items are left in the clear. Only the value item of Source Triplet is encrypted, allowing the essence information to be encrypted prior to wrapping. See Section 7 for a description of the cryptographic information associated with each Encrypted Triplet.

Third, the plaintext Triplets containing essence information have been replaced by Encrypted Triplets – see [e] for details on the KLV (Key-Length-Value) structure. Each Encrypted Triplet is designed to be processed independently, allowing decryption to start anywhere within the encrypted track file[3]. Figure 1 illustrates the correspondence between a plaintext and an Encrypted Triplet[4]. The value *V* of a source plaintext *KLV* Triplet is first encrypted to yield *E(V)*. The latter, along with K and L, is wrapped in a *K'L'V'* Encrypted Triplet. *K'* is a unique label common to all Encrypted Triplets, independent of their content. *L'* refers to the full length of *V'*. *V'* consists of *K*, *L* and *E(V)* from the source Triplet as well as cryptographic information specific to the Encrypted Triplet. This cryptographic information includes, for instance, the initialization vector used in generating *E(V)* and the message integrity code (MIC) used to verify the integrity of the Triplet. The structure of Encrypted Triplets is detailed in Section 7.

[2] This specification assumes that the reader is familiar with the MXF and AS-DCP Track File formats.
[3] While this specification may be applied to both clip-wrapped and frame-wrapped essence, it is ideally suited for frame-wrapped essence, as required by the AS-DCP specification.
[4] This specification does not require the essence to be wrapped in a KLV Triplet to enable its encryption. In other words, essence may be encrypted prior to being wrapped in an Encrypted Triplet.

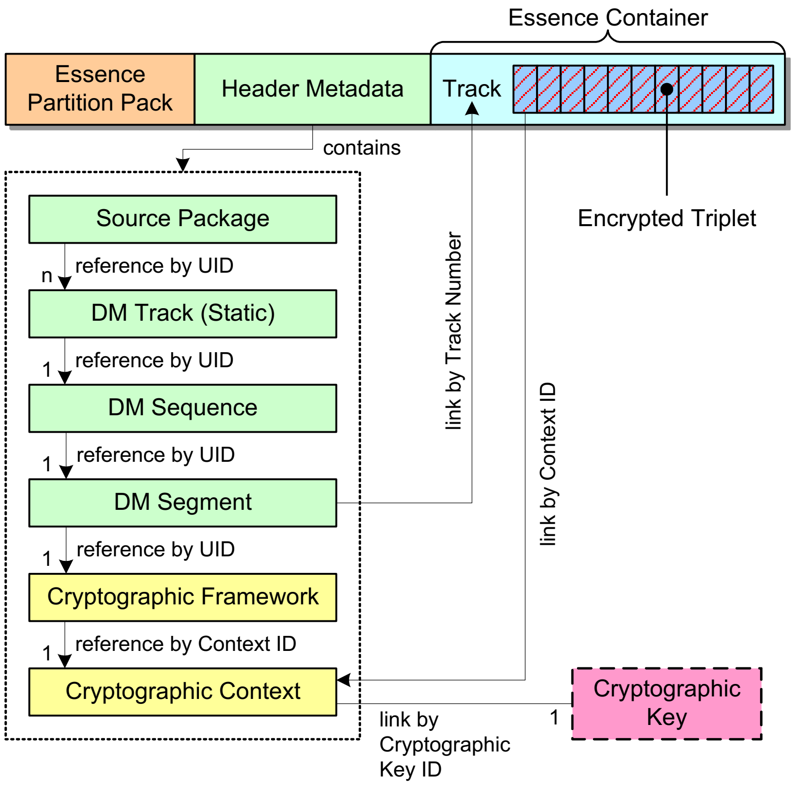

Figure 2. Cryptographic Famework. The DM Track is static since in AS-DCP encrypted track files a single cryptographic key is associated with any given track. Each Encrypted Triplet within a Track must refer to the same Cryptographic Context.

4 Encrypted Essence Container

In order to signal the presence of encrypted tracks, the Essence Container Label of the parent Essence Container of any track containing Encrypted Triplets shall be replaced by the Encrypted Essence Container Label listed in Table 1. This replacement shall occur both within the parent File Package and the Partition Pack.

Table 1. Encrypted Essence Container Label. See Section 10.1 for the complete structure of the Label.

`060e2b34 04010107 0d010301 020b0100`

5 Cryptographic Framework

As depicted in Figure 2, the Cryptographic Context shall be carried in encrypted AS-DCP Track Files as a lightweight MXF Descriptive Metadata Framework, following the principles described in EG42 [1]. Specifically, AS-DCP Track files may contain one or more Descriptive Metadata Tracks containing each a single Cryptographic Framework[5]. The Cryptographic Framework structure is detailed in Table 3.

The Cryptographic Framework forms a Cryptographic DM Scheme. The Cryptographic Framework Label listed in Table 2 shall be included in the Preface set as the identifier of the Cryptographic DM Scheme.

[5] The latter is specified, like any other MXF set as a subclass of an InterchangeObject. For AAF implementations, the Cryptographic Framework is a subclass of DM Framework which is, in turn, a subclass of the AAF interchange object.

Table 2. Cryptographic Framework Label. See Section 10.2 for the complete structure of the Label.

```
060e2b34 04010107 0d010401 02010100
```

The Cryptographic Framework does not contain actual cryptographic information and instead references a single Cryptographic Context, which is described in Section 5.1. Its purpose is to provide compatibility with other MXF Descriptive Metadata, allowing the Cryptographic Context to be exposed in a regular manner.

Table 3. Cryptographic Framework. Lengths are in bytes. The local tags is dynamically assigned.

Item Name	Type	Len	UL	Req ?	Meaning
Cryptographic Framework Key	Set Key	16	060e2b34 02530101 0d010401 02010000	Req	Defines the Cryptographic Framework
Length	BER Length	var	n/a	Req	Set length
InstanceID	UUID	16	060e2b34 01010101 01011502 00000000	Req	Unique identifier for the framework.
GenerationUID	UUID	16	060e2b34 01010102 05200701 08000000	Opt	Optional Generation Identifier
Context SR	Strong Ref (Descrip-tive Set)	16	060e2b34 01010109 06010104 020d0000	Req	Strong reference to the associate Cryptographic Context (see Section 5.1)

5.1 *Cryptographic Framework Key*

The Cryptographic Framework Key uniquely identifies the Set as a Cryptographic Framework being subject to this specification.

Table 4. Cryptographic Framework Key. See Section 10.3 for the complete structure of the Key.

```
060e2b34 02530101 0d010401 02010000
```

5.2 *Length*

The Length item specifies the BER-encoded length of the Cryptographic Framework.

5.3 *Context SR*

The Context SR item contains a strong reference to the Cryptographic Context associated with the Cryptographic Framework.

6 Cryptographic Context

The Cryptographic Context Set contains cryptographic information that applies to encrypted essence tracks as a whole.

Table 5. Cryptographic Context Set. Lengths are in bytes. The local tag is dynamically assigned.

Item Name	Type	Len	UL	Req ?	Meaning
Cryptographic Context Key	Set Key	16	060e2b34 02530101 0d010401 02020000	Req	
Length	BER Length	var	n/a	Req	Set length
InstanceID	UUID	16	060e2b34	Req	Unique identifier for the

			01010101 01011502 00000000		context used by Cryptographic Framework to refer to the Context.
Context ID	UUID	16	060e2b34 01010109 01011511 00000000	Req	Unique identifier used by Encrypted Triplets to refer to the Context.
GenerationUID	UUID	16	060e2b34 01010102 05200701 08000000	Opt	Optional Generation Identifier
Source Essence Container Label	UL Key	16	060e2b34 01010109 06010102 02000000	Req	Essence Container Label for the source essence, prior to encryption
Cipher Algorithm	UL Key	16	060e2b34 01010109 02090301 01000000	Req	Algorithm used for Triplet encryption, if any.
MIC Algorithm	UL Key	16	060e2b34 01010109 02090302 01000000	Req	Algorithm used for Triplet integrity, if any.
Cryptographic Key ID	UUID	16	060e2b34 01010109 02090301 02000000	Req	Unique identifier for the cryptographic key.

6.1 Cryptographic Context Key

The Cryptographic Context Key item uniquely identifies the Set as a Cryptographic Context being subject to this specification.

Table 6. Cryptographic Context Key. See Section 10.4 for the complete structure of the Key.

```
060e2b34 02530101 0d010401 02020000
```

6.2 Length

The Length item specifies the BER-encoded length of the Cryptographic Context value.

6.3 InstanceID

The InstanceID item uniquely identifies this particular Cryptographic Context. It is represented by a UUID. This value is referenced by the Cryptographic Framework.

6.4 Context ID

The Context ID item uniquely identifies this particular Cryptographic Context. It is represented by a UUID. This value is referenced by Encrypted Triplets.

6.5 Source Essence Container Label

The Source Essence Container Label item contains the original Label of the Essence Container to which the Encrypted Triplet belongs. This allows the type of essence contained in the Encrypted Container to be readily determined.

6.6 Cipher Algorithm

The Cipher Algorithm ID item contains a Universal Label that identifies the algorithm and mode used to encrypt the Encrypted Triplets associated with this Cryptographic Context. The permitted values for this item are listed in Table 7. If no cipher algorithm is necessary to process the Encrypted Triplets associated with the Cryptographic Context, then the special value listed in Table 7 shall be used.

Table 7. Cipher Algorithms.

Description	Value
No cipher algorithm used.	`00000000 00000000 00000000 00000000`
AES-CBC with 128-bit key.	`060e2b34 04010107 02090201 01000000`

6.7 MIC Algorithm

The MIC Algorithm ID item contains a Universal Label that identifies the algorithm used to compute the (optional) message integrity code contained in the Encrypted Triplets associated with this Cryptographic Context. The permitted values for this item are listed in Table 8. If no MIC algorithm is necessary to process the Encrypted Triplets associated with the Cryptographic Context, then the special value listed in Table 8 shall be used.

Table 8. Message Integrity Code Algorithms.

Description	Value
No MIC algorithm used.	`00000000 00000000 00000000 00000000`
HMAC-SHA1 with 128-bit key.	`060e2b34 04010107 02090202 01000000`

6.8 Cryptographic Key ID

The Cryptographic Key ID item uniquely identifies the cryptographic key[6] used as input to the cipher and message authentication code algorithms applied to Encrypted Triplets.

7 Encrypted Triplet

The Encrypted Triplet Variable Length Pack, shown in Table 10, contains both encrypted data and cryptographic information specific to the Triplet. As a Variable Length Pack, each individual item in the Value field comprises a Length field and the individual item value. Byte 6 of the Key value (04h) defines that the length field for each individual item is BER short or long form encoded in accordance with SMPTE 336M.

7.1 Encrypted Triplet Key

The Encrypted Triplet Key item uniquely identifies the Triplet as a Triplet being subject to this specification.

Table 9. Encrypted Triplet Key. See Section 10.5 for the complete structure of the Key.

```
060e2b34 02040107 0d010301 027e0100
```

7.2 Length

The Length item specifies the length of the value of the Encrypted Triplet. It is encoded with BER.

7.3 Cryptographic Context Link

The Cryptographic Context Link item refers to the Cryptographic Context used to encrypt the Encrypted Source Value item. It is represented by a UUID.

7.4 Plaintext Offset

The Plaintext Offset item specifies, in bytes, the offset within the Source Value where encryption started. The idea is to allow some header portion of the Source Value to remain in the clear. The Plaintext Offset must be less than or equal to the Source Length.

[6] Informative note: in a typical d-cinema scenario, the cryptographic key referred to by the Cryptographic Key ID parameter has been delivered using a Key Delivery Message.

Table 10. Encrypted Triplet Variable Length Pack. Lengths are in bytes.

	Item Name	Type	Len	UL	Req ?	Meaning
	Encrypted Triplet Key	Pack Key	16	060e2b34 02040107 0d010301 027e0100	Req	
↔	Length	BER Length	var	n/a	Req	Pack length
	Cryptographic Context Link	UUID	16	060e2b34 01010109 06010106 03000000	Req	Link to the Cryptographic Context associated with this Triplet (see Figure 2).
	Plaintext Offset	Uint64	8	060e2b34 01010109 06090201 03000000	Req	Offset within the source at which encryption starts
	Source Key	Key	16	060e2b34 01010106 06010102 03000000	Req	Key of the source Triplet.
↔	Source Length	Uint64	var	060e2b34 01010109 04061002 00000000	Req	Length of the value of the source Triplet
s ↔	Encrypted Source Value	Array of Bytes	var	060e2b34 01010109 02090301 03000000	Req	Encrypted Source value starting at Plaintext Offset
	TrackFile ID	UUID	16	060e2b34 01010109 06010106 02000000	Opt	Identifies the AS-DCP Track File containing this Triplet.
	Sequence Number	Uint64	8	060e2b34 01010109 06100500 00000000	Opt	Sequence number of this Triplet within the Track File
	MIC	Array of Bytes	20	060e2b34 01010109 02090302 02000000	Opt	Keyed HMAC

7.5 Source Key

The Source Key item contains the unmodified Key of the source plaintext Triplet. Note that this Key refers to the identifier of a Triplet, not a cryptographic key.

7.6 Source Length

The Source Length item contains the unmodified BER-encoded length of the source plaintext Triplet. It shall be used in conjunction with the Length item to determine the amount of padding, in bytes, introduced by the encryption process.

7.7 *Encrypted Source Value*

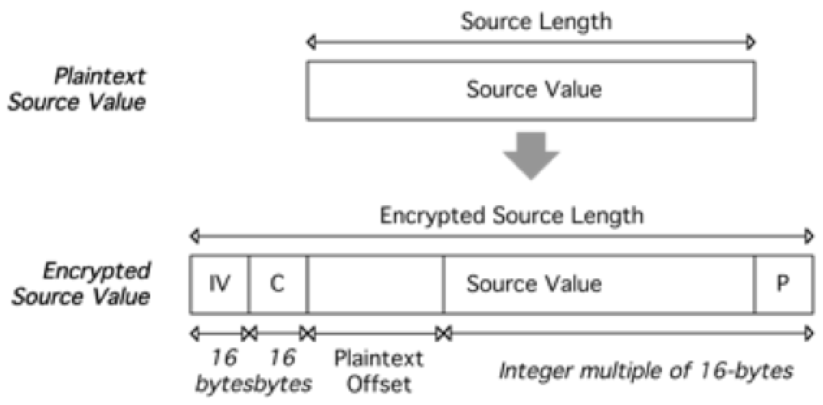

Figure 3. Encrypted Source Value structure. The overlay red hatches indicate encryption. The Initialization Vector (IV), Check Value (C) and Padding (P) values are integral to the Encrypted Source Value item and, as such, are not independent items within the Encrypted Triplet Pack.

As shown in Figure 3, the Encrypted Source Value contains the encrypted source value of the Source Triplet, including Initialization Vector, Check Value and Padding values. It is divided into plaintext and encrypted portions. The plaintext portion contains a 16-byte Initialization Vector (IV) and the first (Plaintext Offset) bytes of the source value – see [a] for additional details on cryptographic initialization vectors. The encrypted portion contains a 16-byte Check Value (C) and the last (Source Length - Plaintext Offset) bytes of the Source Value followed by Padding (P). The size (Padding Length) of the latter is at least one and must be chosen so that (Source Length - Plaintext Offset + Padding Length) is a multiple of 16 bytes, the block size of the AES-128 algorithm. The Padding bytes shall be set in accordance with [3]. The encryption process shall use the algorithm and cryptographic key found in the Cipher Algorithm and Cryptographic Key ID items, respectively, of the Cryptographic Context with which the Encrypted Triplet is associated. The Check Value consists of a static 16-byte value listed in Table 11, and may be used to verify that the correct cryptographic key is being applied.

Table 11. Encrypted Triplet Check Value (hexadecimal notation in network byte order).

0x4348554B4348554B4348554B4348554B

The IV should be chosen randomly for each Encrypted Triplet. Under no circumstances shall the IV be based on an easily predictable sequence.

7.8 *TrackFile ID [optional]*

The optional TrackFile ID item uniquely identifies the AS-DCP Track File in which the Encrypted Triplet appears. It is required only if the MIC item is present. This item is represented by a UUID, and must have the same value for all Encrypted Triplets within a given Track File. The value of this field should equal the UUID used in AS-DCP Composition Lists referencing this file.

7.9 *Sequence Number [optional]*

The optional Sequence Number item must contain an integer which increments by one for each successive Encrypted Triplet contained in a given essence track. It is required only if the MIC item is present. The Sequence Number of the first Encrypted Triplet in a given essence track should be 1.

7.10 *MIC [optional]*

The optional MIC item contains a message integrity code computed using the MIC algorithm and cryptographic key contained in the MIC Algorithm and Cryptographic Key ID items of the Cryptographic

Context associated with this Encrypted Triplet. The MIC algorithm must be applied to every byte preceding the MIC item starting at the first byte of the Encrypted Source Value item.

The key used in the MIC algorithm (MICKey) is derived from the key (CipherKey) referred to by Cryptographic Key ID using

```
   MICKey =
      trunc( HMAC-SHA-1( CipherKey, 0x00112233445566778899aabbccddeeff ) ),
```

where trunc() selects the first 128-bits of the HMAC-SHA-1 output (blocks H0 through H3 as described in the SHA-1 specification).

8 Encrypted Track File Constraints

Encrypted AS-DCP track files shall follow the same specification as plaintext AS-DCP track files [6], with the following additional constraints.

8.1 Encrypted Essence Track

A single cryptographic key, and hence Cryptographic Context, shall be used to encrypt any given essence track. In other words, all Encrypted Triplets associated with a given encrypted track shall refer to the same Cryptographic Context.

8.2 Cryptographic Framework DM Track

AS-DCP Track Files may contain one or more DM Tracks in the MXF File Package which describes the essence of the Track File. Since the same cryptographic key is used to encrypt all Encrypted Triplets within a given encrypted essence track, these DM Tracks shall be Static DM Tracks, per [7] Annex B.17.3.

Each DM Track shall contain a DM Sequence of one and only one DMSegment, per [7] Annex B.19, which shall contain a strong reference to a Cryptographic Framework.

Linking between the DM Tracks and the Essence Track shall be indicated by the TrackIDs property of the DM Segment contained within the DM Track.

In most cases, AS-DCP Files contain only a single Essence Track, and thus the TrackIDs property may safely be omitted. If applications require separate Cryptographic Contexts for separately encrypted Essence and Descriptive Metadata, TrackIDs must be specified.

8.3 Index Tables

In a plaintext track file, each Index Table entry locates a Triplet containing a single frame of essence. Similarly, in an encrypted track file, each Index Table entry shall point to an Encrypted Triplet wrapping a single Triplet, itself containing a single frame of essence.

KLV Fill packets may be used, as defined in [7], to support any KAG requirements and/or to pad each frame of the Essence Container to allow for simple Index Tables using a non-zero EditUnitByteCount value (see [8], table 19).

9 Reference Processing Model

This section specifies a complete processing model for encrypted AS-DCP track files. More specifically it defines a reference process by which a valid encrypted AS-DCP track file is mapped into a valid plaintext AS-DCP track file. It does not however define subsequent operations, such as rendering the underlying essence.

The objective of a fully-specified decryption model is to alleviate the need for a formal specification of the encryption process and facilitate compliance testing. This is conceptually modeled on the exact match approach taken by the Advanced Video Coding (MPEG4 Part 10 and H264L) video compression standard, which defines a single correct bit pattern at the output of a compliant decoding process. The decryption model itself does not force the output to be a valid AS-DCP track file. It is the responsibility of the creator of the encrypted AS-DCP track file to produce an input file which will result in a valid plaintext AS-DCP track file when decrypted – assuming all relevant cryptographic keys are available. To be considered valid, an encrypted AS-DCP Track File must meet this requirement, in addition to complying with other parts of the AS-DCP specification.

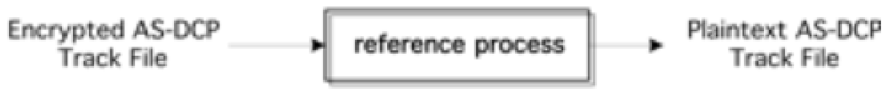

Figure 2. Reference Decryption Model.

If the input file is not a valid encrypted AS-DCP track file, then the decryption may terminate on an unrecoverable error. In fact, even if the input is a valid encrypted AS-DCP track file, the lack of proper decryption keys will disrupt the decryption process, effectively resulting in an error status. The decryption model will report encountered errors but implementation behavior of under such conditions is left to the application.

9.1 Overall Flow

This section describes the overall flow of information in the decryption model. The decryption model is divided into modules which perform specific functions using specific inputs and outputs. Some of these modules are weakly coupled and could accommodate different algorithms or specialized needs that are outside the scope of this document.

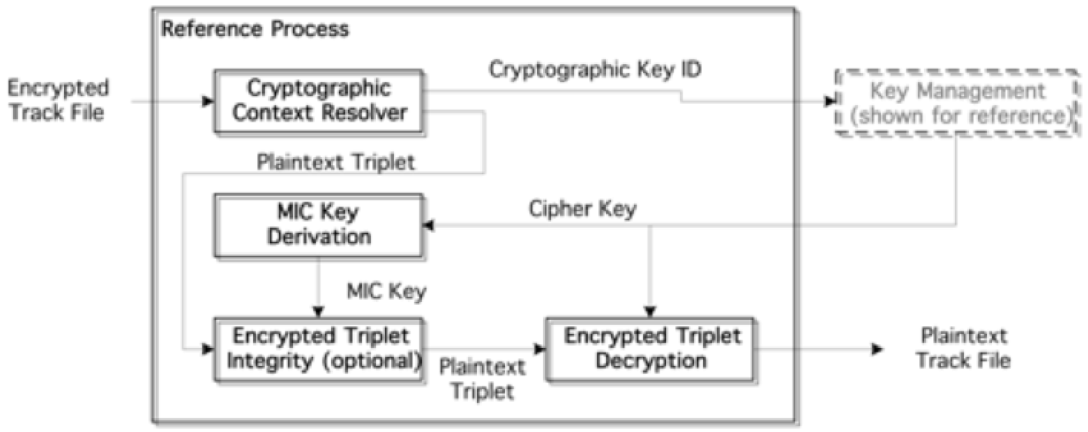

Figure 4. Decryption Process Flowchart. The Key Management Module is not covered by this specification and shown here only for reference.

9.2 Modules

The reference processing model contains a number of modules, each with specific interfaces. This section describes these modules in detail.

9.2.1 MIC Key Derivation Module

If the optional Integrity Pack item is present in the Encrypted Triplet, the MIC Key Derivation Module is used to prepare a cryptographic key for use by the Encrypted Triplet Integrity Module. As depicted in Figure 5, it takes as input a Cipher Key and produces as output a MIC Key.

Cipher Key
MIC Derivation Module
MIC Key

Figure 5. MIC Key Derviation Module

The output key (MIC Key) is derived from the input key (CipherKey) using

```
MICKey =
    trunc( HMAC-SHA-1(CipherKey, 0x00112233445566778899aabbccddeeff ) ).
```

where trunc() selects the first 128-bits of the HMAC-SHA-1 output (blocks H0 through H3 as described in the SHA-1 specification).

9.2.2 Encrypted Triplet Integrity Module

If the optional Integrity Pack item is present in the Encrypted Triplet, the Encrypted Triplet Integrity module may be used to verify the integrity of the encrypted source value and integrity pack items of the Encrypted Triplet. As depicted in Figure 6, it takes as input a single Encrypted Triplet and cryptographic key, and produces as output the same Encrypted Triplet and an error status code. The value of the latter indicates whether the integrity check succeeded. The module carries no state information from one Encrypted Triplet to the next, i.e. it is memory-less. Although not specified as part of the Reference Processing Model, a stateful process shall be performed on elements of the Integrity Pack, in order to detect duplicated, missing, or reordered Encrypted Triplets.

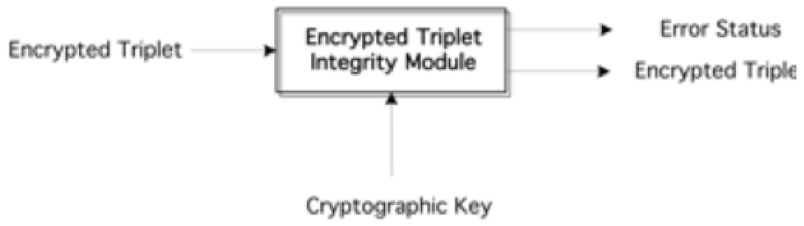

Figure 6. Encrypted Triplet Integrity Module.

The module operates using the HMAC-SHA1 algorithm. As specified in Section 6.7, no other message integrity check algorithm is used in conjunction with AS-DCP track files. Given an input Encrypted Triplet, the module creates its output Encrypted Triplet and Error status according to the following rules.

- The output Encrypted Triplet is identical to the input Encrypted Triplet.
- If the input Triplet is not encrypted (i.e. the K item is not the Encrypted Triplet Key in Table 9), or if the optional MIC item is not present, no other processing occurs. If the input Triplet is encrypted and the MIC item is present, the rule below applies.
- Using the input cryptographic key, the HMAC-SHA1 algorithm is applied to the concatenation of every byte preceding the MIC item starting at the first byte of the Encrypted Source Value item. The result of this computation is compared with the MIC item of the Encrypted Triplet. If the two differ, the integrity check failed and an error status is returned.

9.2.3 Encrypted Triplet Decryption Module

The core decryption operations, e.g. executing an AES block decryption, occur in the Encrypted Triplet Decryption module. As depicted in Figure 7, the latter takes as input a single Triplet and cryptographic key, and produces as output a single Triplet and an error status code as output. The module carries no state information from one KLV Triplet to the next, i.e. it is memory-less.

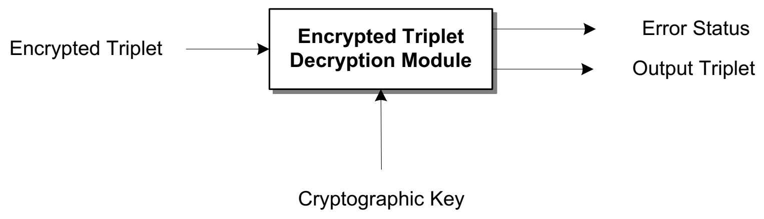

Figure 7. Encrypted Triplet Decryption Module.

The module operates in a mode based on AES-128 in CBC mode [5,7]. As specified in Section 6.6, no other symmetric encryption algorithm is used in conjunction with AS-DCP track files. Given an input Encrypted Triplet, the module creates its output KLV Triplet according to the following rules.

- If the input Triplet is not encrypted (i.e. the K item is not the Encrypted Triplet Key of Table 9), the output Triplet is set equal to the input Triplet and no other processing occurs. If the input Triplet is encrypted, the rules below apply.
- The output K item is equal to the Source Key item of the input Encrypted Triplet.
- The output L item is equal to the Source Length item of the input Encrypted Triplet.
- If the Plaintext Offset item is equal to the output L item (i.e., there is no encrypted portion of the essence) then cryptographic processing may be omitted entirely. The first 32 bytes of the Encrypted Source Value item may therefore be ignored and the following L bytes copied to the output V item, completing the decryption process. If the Plaintext Offset does not equal L, the following steps apply.
- The first 16 bytes (in network byte order) of the Encrypted Source Value item found in the input Encrypted Triplet are used as the Initialization Vector specified by the CBC mode.
- The next 16 bytes of the Encrypted Source Value item shall be processed according to AES-128 in CBC mode. The 16-byte output block shall be equal to the Check Value of Table 11; if not an error status is returned and no Triplet is output.
- The next (Plaintext Offset) bytes of the Encrypted Source Value item are copied to the beginning of output V item unchanged, i.e. as plaintext. If (Plaintext Offset) is larger than the output L, then an error status is returned and no Triplet is output.
- The remaining length of the Encrypted Source Value input field must be a multiple of 16 bytes – the block size of AES-128. If not, an error status is returned and no Triplet is output.
- Subsequent groups of 16 bytes are processed as cipher blocks according to AES-128 in CBC mode, using the Cryptographic Key input.
- The fir–t (L - Plaintext Offset) bytes produced by this process are copied to the output V item. If there have not been enough ciphertext blocks to genera–e (L - Plaintext Offset) bytes, then an error status is returned and no Triplet is output. This step removes the cryptographic padding that was added during encryption to make the AES input a multiple of 16 bytes long. Notice that the values of the padding bytes are specified but not checked.

9.2.4 Cryptographic Context Resolver Module

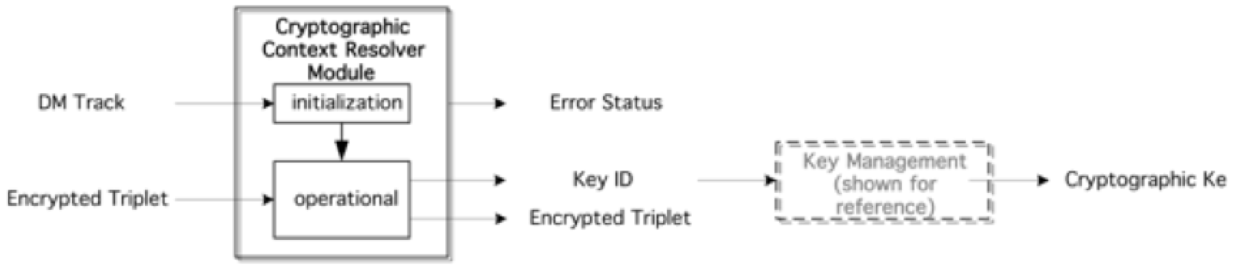

Figure 8. Cryptocontext Resolver Module. The Key Management Module is not covered by this specification and shown here only for reference.

The Cryptographic Context Resolver module conceptually operates in two phases, namely initialization and operational phases. In the initialization phase, the Cryptographic Context is extracted from the encrypted Track File and retained in the module. In the operational phase, the module outputs Encrypted Triplet and Cryptographic Key ID according to the following rules:

- The output Encrypted Triplet is identical to the input Encrypted Triplet
- If the Cryptographic Context Link element of the input Encrypted Triplet matches the Cryptographic Context Link element of the Cryptographic Context Set, the output Cryptographic Key ID equals the Cryptographic Key ID element of the Cryptographic Context Set
- If the Cryptographic Context Link elements do not match, the resolution has failed and an error status is returned
- The Cryptographic Context Resolver is the only module in the Reference Processing Model which retains state from one Triplet to another.

10 Label and Key Structures

10.1 Encrypted Essence Container Label

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label side	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	04h	Labels
6	Registry Designator	01h	Labels Registry
7	Structure Designator	01h	Labels Structure
8	Version Number	vvh	Registry Version at the point of registration of this label
9	Item Designator	0Dh	Organisationally Registered
10	Organisation	01h	AAF Association
11	Application	03h	Essence containers
12	Structure Version	01h	Version 1
13	Essence Container Kind	02h	MXF Generic Container
14	Mapping Kind	0Bh	Encrypted Essence Container
15	Locally Defined	01h	Frame Wrapped
16	Reserved	00h	

Note: Bytes 1-12 of this label are defined by the essence container label [379M].

10.2 Cryptographic Framework Label

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label side	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	04h	Labels
6	Registry Designator	01h	Labels Registry
7	Structure Designator	01h	Labels Structure
8	Version Number	vvh	Registry Version at the point of registration of this label
9	Item Designator	0Dh	Organizationally Registered
10	Organization	01h	AAF Association
11	Application	04h	MXF/AAF compatible Metadata Labels
12	Structure Version	01h	Version 1
13	Scheme Kind	02h	Cryptographic DM Scheme

14	Scheme Designator	01h	Cryptographic Scheme version 1
15	Scheme Designator	01h	AS-DCP Cryptographic Framework
16	Reserved	00h	

Note: Bytes 1-12 of this label are defined for MXF descriptive metadata schemes [377M]

10.3 Cryptographic Framework Key

Byte No.	**Description**	**Value (hex)**	**Meaning**
1	Object Identifier	06h	
2	Label side	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	02h	Groups (Sets and Packs)
6	Registry Designator	53h	Local Set, 2-octet length fields, 2-octet tag fields
7	Structure Designator	01h	Set/Pack Dictionary
8	Version Number	vvh	Registry Version at the point of registration of this key
9	Item Designator	0Dh	Organizationally Registered
10	Organization	01h	AAF Association
11	Application	04h	MXF/AAF compatible Set Keys
12	Structure Version	01h	Version 1
13	Scheme Kind	02h	Cryptographic DM Scheme
14	Set Designator	01h	CryptographicFramework
15		00h	
16		00h	

Editorial Note: Experimental Metadata Designator shall be replaced prior to issuance of this Standard.

Note: Bytes 1-12 of this key are specified for structural metadata sets in [377M].

10.4 Cryptographic Context Key

Byte No.	**Description**	**Value (hex)**	**Meaning**
1	Object Identifier	06h	
2	Label side	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	02h	Groups (Sets and Packs)
6	Registry Designator	53h	Local Set, 2-octet length fields, 2-octet tag fields
7	Structure Designator	01h	Set/Pack Dictionary
8	Version Number	vvh	Registry Version at the point of registration of this key
9	Item Designator	0Dh	Organizationally Registered
10	Organization	01h	AAF Association
11	Application	04h	MXF/AAF compatible Set Keys
12	Structure Version	01h	Version 1
13	Scheme Kind	02h	Cryptographic DM Scheme
14	Set Designator	02h	CryptographicContext
15		00h	
16		00h	

Editorial Note: Experimental Metadata Designator shall be replaced prior to issuance of this Standard.

Note: Bytes 1-12 of this key are specified for structural metadata sets in [377M].

10.5 Encrypted Triplet Key

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label side	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	02h	Groups (Sets and Packs)
6	Registry Designator	04h	Variable Length Pack, BER element length encoding
7	Structure Designator	01h	??
8	Version Number	vvh	Registry Version at the point of registration of this key
9	Item Designator	0Dh	Organisationally Registered
10	Organisation	01h	AAF Association
11	Application	03h	Essence containers
12	Structure Version	01h	Version 1
13	Essence Container Kind	02h	MXF Generic Container
14	Mapping Kind	7Eh	Encrypted Essence
15	Locally Defined	01h	Encrypted Triplet
16	Reserved	00h	

Editorial Note: Experimental Metadata Designator shall be replaced prior to issuance of this Standard.

10.6 Encryption Algorithm Specifier Label

Byte No.	**Description**	**Value (hex)**	**Meaning**
1	Object Identifier	06h	
2	Label side	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	04h	Labels
6	Registry Designator	01h	Labels Registry
7	Structure Designator	01h	Labels Structure
8	Version Number	vvh	Registry Version at the point of registration of this label
9	Item Designator	0Fh	Experimental Metadata
10		01h	
11		03h	
12		01h	
13	Algorithm Class	01h	Symmetric Encryption
14	Algorithm Designator	01h 02h	No cipher used AES-128 CBC
15-16	Reserved	00h	

Note: Bytes 1-8 of this label are defined by the KLV data encoding protocol [336M].

10.7 Integrity Algorithm Specifier Label

Byte No.	**Description**	**Value (hex)**	**Meaning**
1	Object Identifier	06h	
2	Label side	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	04h	Labels
6	Registry Designator	01h	Labels Registry
7	Structure Designator	01h	Labels Structure
8	Version Number	vvh	Registry Version at the point of registration of this label
9	Item Designator	0Fh	Experimental Metadata
10		01h	
11		03h	
12		01h	
13	Algorithm Class	02h	Integrity Check
14	Algorithm Designator	01h 02h	No algorithm used HMAC-SHA-1 with 128-bit key
15-16	Reserved	00h	

Note: Bytes 1-8 of this label are defined by the KLV data encoding protocol [336M].

11 Bibliography

a. Bruce Schneier, *Applied Cryptography (Second Edition)*, Wiley, 1996.

b. SMPTE, Proposed 390M, 2003. For Television – MXF OP-ATOM

d. SMPTE 298M-1997, for Television — Universal Labels for Unique Identification of Digital Data

e. SMPTE 336M–2001, Television — Data Encoding Protocol Using KLV

f. SMPTE RP210-2002, Metadata Dictionary Registry of Metadata Element Descriptions

g. SMPTE RP 224, SMPTE Labels Registry

h. Track File requirements documented by DC28.20

12 Annex A – Security Properties (informative)

This specification has the following security properties.

- The standard allows the first part of the essence frame value to be unencrypted. The size of this part can be set independently for each frame.
- The second part of each essence frame is encrypted using a strong algorithm (AES) in an appropriate and well-understood mode (Cipher Block Chaining).
- The proposal provides partial integrity protection (tamper detection). Assuming that the 44-byte sequence consisting of the TrackFile ID, Sequence Number and MIC items is delivered to a secure processing device along with the Encrypted Source Value, some types of manipulation may be detected.
    - An attack which changes the order of essence frames in the track file will be detectable, based on sequence numbers.
    - An attack which deletes or repeats complete frames will be detectable, based on sequence numbers.
    - An attack which inserts or substitutes frames from a different Track File will be detectable, even if that Track File uses identical encryption and MAC keys, based on TrackFile ID.
    - An attack which deletes, adds, or changes any bits of the ciphertext will be detectable.
    - An attack which splices together parts of different frames will be detectable.
- Certain types of tampering are not detectable using the Integrity Check Pack.
    - An attacker is free to change the length of KLV Fill items.
    - An attacker is free to change all the metadata in the file including the Key and Length of any triple and most of the fields within the Value portion of the triples (e.g., the Plaintext Offset or Cryptographic Context Link, but not the Integrity Check Pack), and all the fields in the Cryptographic Context and the Cryptographic Framework.
- The derived MIC key may be computed in a secure environment and delivered to a less secure integrity-checking device without risking the exposure of the Cipher Key encrypting the content itself.
- Only an entity that knows the decryption key can tell whether the file will decrypt properly. For example, an attacker could interfere with cryptographic key delivery or synchronization (e.g. by modifying the Cryptographic Context Pack while it is on the server), and only during playback will the problem be noticed.

Change History

Ver	Date	By	Sect	Description
1	11 November 2003		All	Initial Draft
2	1 March 2004		Most	Addressed Comments from 2848B ballot New overview section Refined scope Changed section order SMPTE ballot format Updated Reference Decryption Model Many other changes
3	12 April 2004		Most	Editorial tweaks Fixed linkage between Framework, Context and Encrypted Triplet. Normative references. Expanded Label definition tables. Added a section detailing the structure of the Labels and Keys defined in this document.
4	2004-07-21		Most	Updated Labels
5	2004-08-04		Table 3 &5	Took Set Keys from Table 4 & 6 (not 2 & 4)