SMPTE 429-6-2006

SMPTE STANDARD

D-Cinema Packaging — MXF Track File Essence Encryption

Table of Contents

Approved
October 3, 2006

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in Part XIII of its Administrative Practices.

Proposed SMPTE Standard 429-6 was prepared by Technology Committee DC28.

1 Scope

This standard defines the syntax of encrypted D-Cinema non-interleaved MXF frame-wrapped track files and specifies a matching reference decryption model. It uses the AES cipher algorithm for essence encryption and, optionally, the HMAC-SHA1 algorithm for essence integrity. The D-Cinema track file format is designed to carry D-Cinema essence for distribution to exhibition sites and is specified in the Sound and Picture Track File specification.

This standard assumes that the cryptographic keys necessary to decrypt and verify the integrity of encrypted 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. The scope is limited to D-Cinema and does not 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.

SMPTE 336M-2001, Television — Data Encoding Protocol Using Key-Length-Value

SMPTE 377M-2004, Television — Material Exchange Format (MXF) — File Format Specification

SMPTE 429-3-2006, D-Cinema Packaging — Sound and Picture Track File

IETF 2898 (September 2000). PKCS #5: Password-Based Cryptography Specification – Version 2.0.

IETF 2104 (February 1997). HMAC: Keyed-Hashing for Message Authentication

National Institute of Standards and Technology (December 1, 2001). Recommendation for Block Cipher Modes of Operation Methods and Techniques (SP 800-38A).

National Institute of Standards and Technology, FIPS 197 (November 26, 2001). Advanced Encryption Standard (AES).

National Institute of Standards and Technology, FIPS PUB 186-2 (+Change Notice 1) (January 27, 2000). Digital Signature Standard (DSS).

3 Overview

This specification defines the encryption of the sensitive essence information contained in D-Cinema Track Files using the Advanced Encryption Standard (AES) cipher algorithm in Cipher Block Chaining (CBC) mode as defined in NIST SP 800-38A. 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 Track File to be encrypted using a single cryptographic key. The resulting encrypted Track File is extremely similar to a plaintext Track File, which is itself a constrained version of the MXF OP-ATOM operational pattern[1]. It differs in the following three areas.

First, the Essence Container Label associated with the plaintext track is replaced by an Encrypted Essence Container Label. The replacement Label signals the presence of encrypted essence and allows any receiving MXF application which cannot perform decryption to “fail fast” as described in SMPTE EG 41. The Encrypted Essence Container is defined 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 defined in Sections 5 and 5.1.

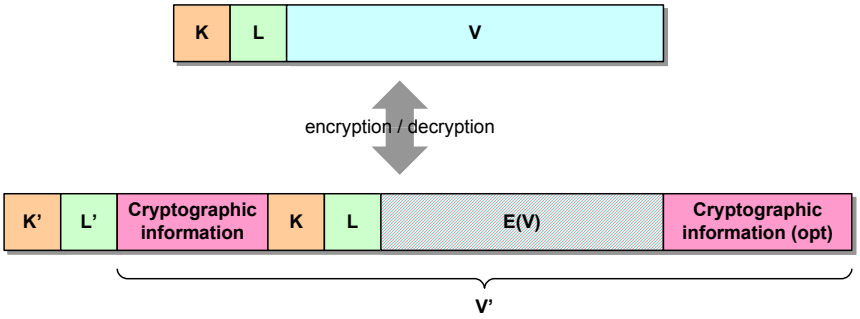

Figure 1 – Correspondence between Source and Encrypted Triplets.

Red hatching depicts the encrypted portion of the Encrypted Triplet; other items are left as plaintext. 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 SMPTE 336M for details on KLV (Key-Length-Value) coding. Each Encrypted Triplet, is designed to be processed independently, allowing decryption to start anywhere within the encrypted Track File. Figure 1 illustrates the correspondence between a plaintext and an Encrypted Triplet[2]. The value *V* of a source plaintext *KLV* Triplet is first encrypted to yield *E(V)*. The encrypted value *E(V)*, 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.

[1] This specification assumes that the reader is familiar with the MXF and Track File formats.

[2] 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.

4 Encrypted Essence Container

In order to signal the presence of encrypted tracks, the Essence Container Label of any track containing Encrypted Triplets shall be replaced by the Encrypted Essence Container Label listed in Table 1. This replacement shall occur both in the Preface set and in the Partition Pack. The Essence Container Label in the File Descriptor (SMPTE 377M) shall however remain unchanged to identify the underlying plaintext essence.

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 Track Files as an MXF Descriptive Metadata (DM) Framework[3]. Specifically, Track Files may contain one or more Descriptive Metadata Tracks containing each a single Cryptographic Framework[4]. The Cryptographic Framework structure is detailed in Table 3.

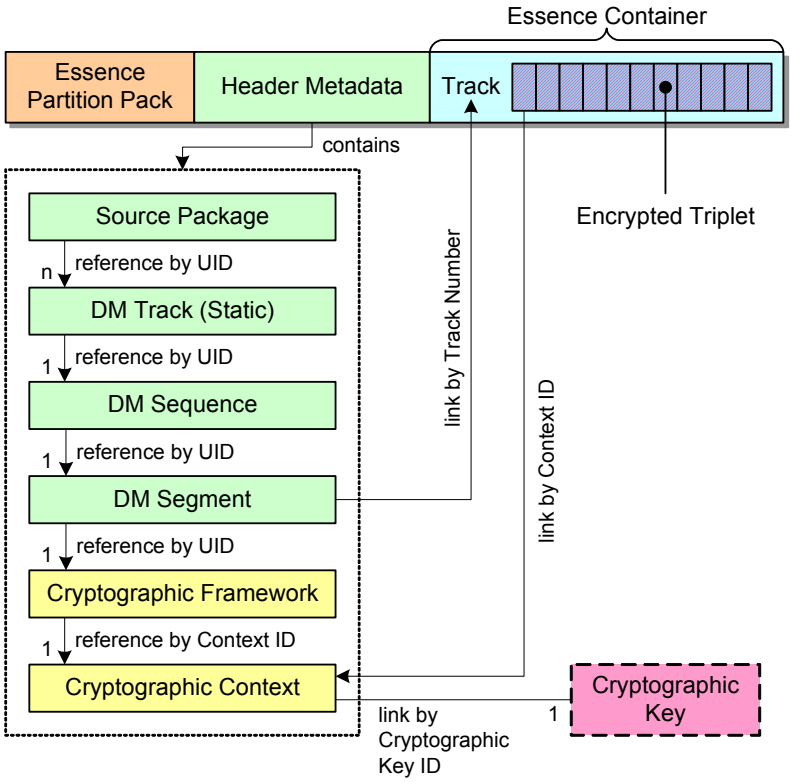

Figure 2 – Cryptographic Framework

The DM Track is static since in 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.

[3] DM frameworks are defined in SMPTE 377M under Plug-in Mechanism and follow the principles described in SMPTE EG 42.

[4] (Informative) The Cryptographic Framework is specified as a subclass of the DM Framework abstract superclass (see SMPTE 380M).

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.

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 defined in Section 6. Its purpose is to provide compatibility with other MXF Descriptive Metadata, allowing the Cryptographic Context to be exposed in a consistent manner.

The following defines the items contained in the Cryptographic Framework Set. With the exception of InstanceID and GenerationUID, which are already defined in SMPTE 377M, all Local Tag values for the descriptor shall be dynamically allocated as defined in SMPTE 377M section 8.2.2 (Local tag values). The translation from each dynamically allocated local tag value to its full UL value can be found using the Primer Pack mechanism defined in SMPTE 377M section 8.2 (Primer Pack).

Table 3 – Cryptographic Framework Set (Lengths are in bytes)

Item Name	Type	Len	UL	Req ?	Meaning
Cryptographic Framework Key	Set Key	16	060e2b34 02530101 0d010401 02010000	Req	Defines the Cryptographic Framework Set
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 6)

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 length of the Cryptographic Framework encoded using Basic Encoding Rules (BER) per SMPTE 336M.

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[5] contains cryptographic information that applies to encrypted essence tracks as a whole. The following defines the items contained in the Cryptographic Context Set. With the exception of InstanceID and GenerationUID, which are already defined in SMPTE 377M, all Local Tag values for the descriptor shall be dynamically allocated as defined in SMPTE 377M section 8.2.2 (Local tag values). The translation from each dynamically allocated local tag value to its full UL value can be found using the Primer Pack mechanism defined in SMPTE 377M section 8.2 (Primer Pack).

Table 5 – Cryptographic Context Set (Lengths are in bytes)

Item Name	Type	Len	UL	Req ?	Meaning
Cryptographic Context Key	Set Key	16	060e2b34 02530101 0d010401 02020000	Req	Defines the Cryptographic Context Set
Length	BER Length	var	n/a	Req	Set length
InstanceID	UUID	16	060e2b34 01010101 01011502 00000000	Req	Unique identifier for the context used by Cryptographic Framework to refer to the Context.
GenerationUID	UUID	16	060e2b34 01010102 05200701 08000000	Opt	Optional Generation Identifier
Context ID	UUID	16	060e2b34 01010109 01011511 00000000	Req	Unique identifier used by Encrypted Triplets to refer to the Context.
Source Essence Container Label	UL	16	060e2b34 01010109 06010102 02000000	Req	Essence Container Label for the source essence, prior to encryption
Cipher Algorithm	UL or zero	16	060e2b34 01010109 02090301 01000000	Req	Algorithm used for Triplet encryption, if any.
MIC Algorithm	UL or zero	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.

[5] (Informative) The Cryptographic Context is a subclass of an InterchangeObject (see SMPTE 380M Annex C)

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 length of the Cryptographic Context value encoded using Basic Encoding Rules (BER) per SMPTE 336M.

6.3 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.

6.4 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.

6.5 Cipher Algorithm

The Cipher Algorithm ID item identifies the algorithm and mode used to encrypt the Encrypted Triplets associated with this Cryptographic Context. It shall contain one of the values listed in Table 7.

Table 7 – Cipher Algorithms

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

The first row of Table 7 is a special value that shall be used to indicate that no cipher algorithm is necessary to process the Encrypted Triplets associated with the Cryptographic Context. The second row identifies the sole permitted value of the cipher algorithm — Section 10.6 defines the complete structure of this label.

6.6 MIC Algorithm

The MIC Algorithm ID item identifies the algorithm used to compute the (optional) message integrity code contained in the Encrypted Triplets associated with this Cryptographic Context. It shall contain one of the values listed in Table 8.

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

The first row of Table 8 is a special value that shall be used to indicate that no MIC algorithm is necessary to process the Encrypted Triplets associated with the Cryptographic Context. The second row identifies the sole permitted value of the MIC algorithm — Section 10.7 defines the complete structure of this label.

6.7 Cryptographic Key ID

The Cryptographic Key ID item uniquely identifies the cryptographic key used as input to the cipher and message authentication code algorithms applied to Encrypted Triplets. It shall be encoded as a UUID. See Section 9.1 for a description of its use.

7 Encrypted Triplet

The Encrypted Triplet Variable Length Pack, shown in Table 9, 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. All items in a pack are required. Any item labeled *Opt* means that the value of the Length-Value pair is not present and therefore the Length is zero.

Table 9 – Encrypted Triplet Variable Length Pack

Item Name	Type	Len	UL	Req ?	Meaning
Encrypted Triplet Key	Pack Key	16	060e2b34 02040101 0d010301 027e0100	Req	Defines the Encrypted Triplet Variable Length Pack
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 Section 6)
Plaintext Offset	UInt64	8	060e2b34 01010109 06090201 03000000	Req	Offset within the source at which encryption starts
Source Key	Key	16	060e2b34 01010109 06010102 03000000	Req	Key of the source Triplet.
Source Length	UInt64	8	060e2b34 01010109 04061002 00000000	Req	Length of the value of the source Triplet
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 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 Hashing for Message Authentication (HMAC)

7.1 Encrypted Triplet Key

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

Table 10 – Encrypted Triplet Key

(See Section 10.5 for the complete structure of the Key.)

060e2b34 02040101 0d010301 027e0100

7.2 Length

The Length item specifies the length of the value of the Encrypted Triplet encoded using Basic Encoding Rules (BER) per SMPTE 336M.

7.3 Cryptographic Context Link

The Cryptographic Context Link item is a link to the Cryptographic Context where the cryptographic information used to create the Encrypted Source Value property is defined. Specifically it contains a UUID, which is the InstanceID of the target Cryptographic Context.

7.4 Plaintext Offset

The Plaintext Offset item shall specify, in bytes, the offset within the Source Value where first byte of encryption started. The Plaintext Offset value shall be less than or equal to the Source Length. As detailed in Section 9.2.4, implementations processing the Encrypted Track File shall handle any legal value of this parameter.

The purpose of the Plaintext Offset item is to allow some header portion of the Source Value to remain in the clear. The selection of appropriate values for the Plaintext Offset will likely depend on the kind of essence contained in the Encrypted Track File, and is hence beyond the scope of this standard.

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 length of the source plaintext Triplet encoded as a UInt64 integer. 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

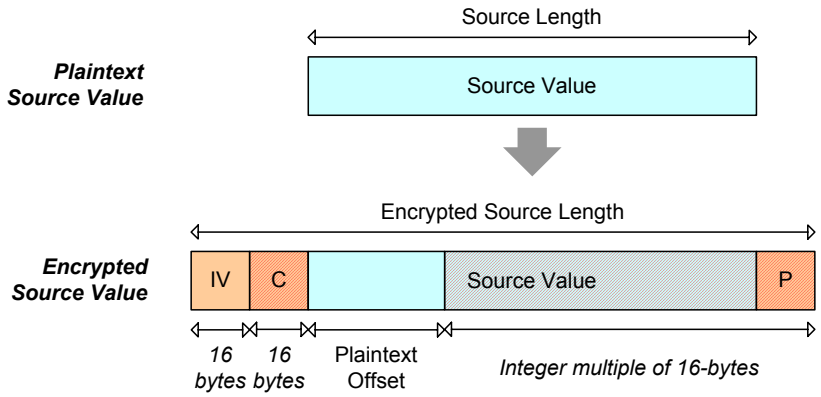

Figure 3 – Encrypted Source Value structure

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. The Encrypted Source Value corresponds to the hatched portion of Encrypted Triplet depicted in Figure 1.

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[6]. 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 the 16-byte block mode described in section B.2.4 of IETF 2898. 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 shall consist of the 16-byte value listed in Table 11, and may be used to verify that the correct cryptographic key is being applied. In other words, upon decryption of the Encrypted Triplet, the processing application may verify that the proper cryptographic key was used by comparing the Check Value recovered with that of Table 11.

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

4348554B 4348554B 4348554B 4348554B

The IV should be chosen randomly for each Encrypted Triplet. The IV shall not be based on an easily predictable sequence.

7.8 TrackFile ID [optional]

The optional TrackFile ID item uniquely identifies the Track File to which the Encrypted Triplet belongs. It shall be present if and only if the MIC item is present. This item is a UUID, and shall have the same value for all Encrypted Triplets within a given Track File.

INFORMATIVE NOTE – Use of this identifier is described under Package IDs in [Sound & Picture Track File]; RP205 gives guidance on assignment of UMIDs.

If no TrackFile ID is associated with the Encrypted Triplet, then the length of the TrackFile ID item shall be 0.

[6] Bruce Schneier, *Applied Cryptography* contains additional details on cryptographic initialization vectors.

7.9 Sequence Number [optional]

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

If no Sequence Number is associated with the Encrypted Triplet, then the length of the Sequence Number item shall be 0.

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. If the MIC item is present, the TrackFile ID item and the Sequence Number item shall also be present. The MIC algorithm shall be applied to every byte preceding the MIC item starting at the first byte of the Encrypted Source Value item (i.e. the first byte of the IV value). Length fields of TrackFileID, SequenceNumber, and MIC shall be included.

The key used in the MIC algorithm (MICKey) is derived from the key (CipherKey) referred to by Cryptographic Key ID using the combination of algorithms defined in Appendix 3.1 and Appendix 3.3 of FIPS 186-2. Specifically the MICKey shall equal to x_1 per Appendix 3.1 using CipherKey as the seed-key XKEY, setting $XSEED_j = 0$ and constructing the function G(t,c) per Appendix 3.3. In addition, since Appendix 3.1 is being used as a general random number generator, the term “mod q” in step 3.c shall be omitted, per the “General Purpose Random Number Generation” of the Change Notice 1 addendum. x_0 shall be discarded.

If no MIC is associated with the Encrypted Triplet, then the length of the MIC item shall be 0.

8 Encrypted Track File Constraints

Encrypted Track Files shall follow the same specification as plaintext Track Files per SMPTE 429-3, 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

Track Files may contain one or more DM Tracks in the MXF File Package which describe the Track File. Since the same cryptographic key is used to encrypt all Encrypted Triplets within a given encrypted essence track, the Cryptographic Framework DM Track shall be a Static DM Track, see 377M at Specifications for the Generic Package.

Each Cryptographic Framework DM Track shall contain a DM Sequence comprising one DMSegment, see 377M Annex at Specifications for the Generic Package, which shall contain a strong reference to a Cryptographic Framework.

Linking between the Cryptographic Framework DM Tracks and the Essence Track shall be made using the TrackIDs property of the DM Segment contained within the DM Track.

In most cases, 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 tracks, metadata or essence, TrackIDs shall 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.

INFORMATIVE NOTE – As Index Tables for encrypted and plaintext Track Files point into different data streams, their contents may differ. KLV Fill packets may be used, as defined in SMPTE 377M, to support any KAG requirements and/or to pad each frame of the Essence Container to allow for identical Index Tables in encrypted and plaintext representations, or Index Tables using a non-zero EditUnitByteCount value (see SMPTE 377M at Index Table Segments).

9 Reference Decryption Processing Model

This section specifies the complete decryption processing model for encrypted Track Files. More specifically it defines a reference process by which a valid encrypted Track File is mapped into a valid plaintext 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 (MPEG-4 Part 10 and H.264) 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 Track File. It is the responsibility of the creator of the encrypted Track File to produce an input file which will result in a valid plaintext Track File when decrypted – assuming all relevant cryptographic keys are available. To be considered valid, an encrypted Track File must meet this requirement, in addition to complying with other parts of the specification.

Figure 4 – Reference Decryption Model

If the input file is not a valid encrypted Track File, then the decryption may terminate on an unrecoverable error. In fact, even if the input is a valid encrypted 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 standard.

In Figure 5, the Cryptographic Key ID is provided to the Key Management module and its value is used to identify the Cipher Key required to decrypt the encrypted triplets. The definition of the Key Management module is beyond the scope of this standard.

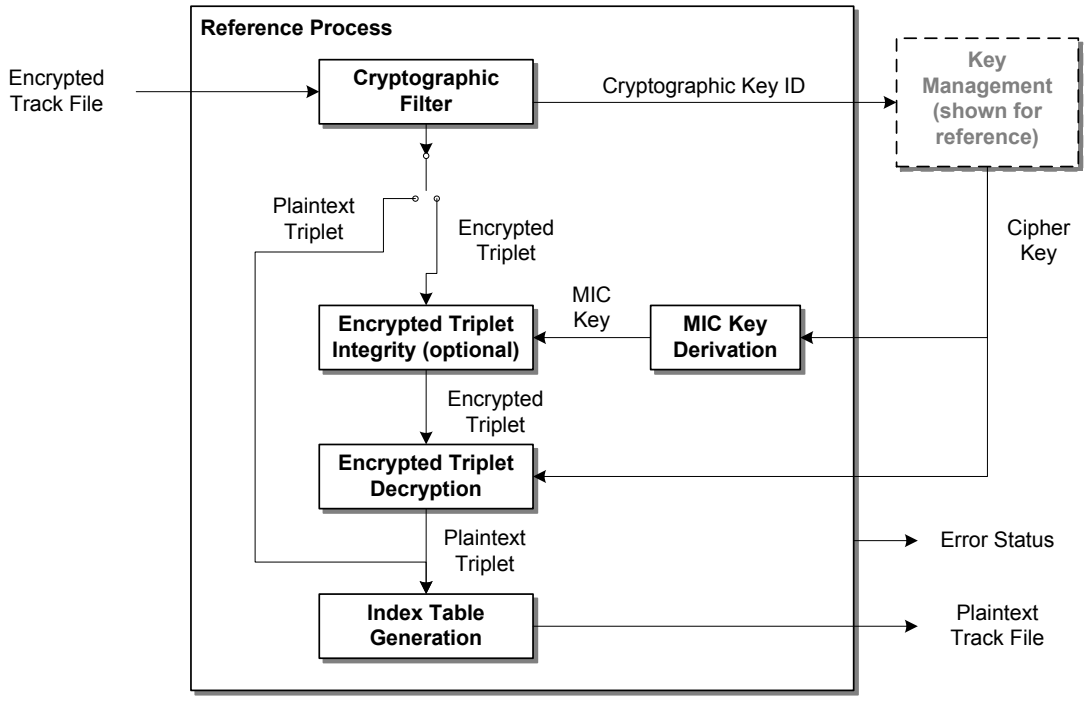

Figure 5 – Decryption Process Flowchart.

The Key Management module is outside the scope of 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 Cryptographic Filter Module

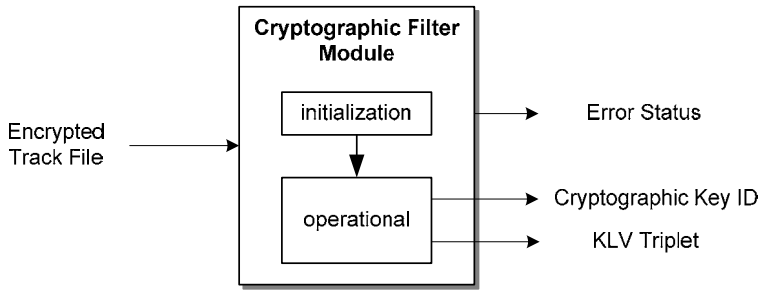

Figure 6 – Cryptographic Filter Module

The Cryptographic Filter module conceptually operates in two phases, namely initialization and operational phases. In the initialization phase, Cryptographic Contexts are extracted from the encrypted Track File and retained in the module. In the operational phase, the module shall process the Encrypted Track File according to the following rules.

- Unless otherwise specified, every KLV Triplet contained in the Encrypted Track File shall remain unmodified.
- DM Tracks containing a Cryptographic Framework, and corresponding Header metadata, shall be discarded.
- Each Index Table corresponding to an Encrypted Essence Container shall be discarded. Index Tables are recreated by the Index Table Generation Module.
- The Essence Container Label of any track containing Encrypted Triplets shall be replaced by the original Essence Container Label stored in the corresponding Cryptographic Context.
- Each Encrypted Triplet shall be decrypted according to the flowchart in Figure 5. The module shall output each Encrypted Triplet unchanged. It shall also output the Cryptographic Key ID found in Cryptographic Context Set referenced by the Cryptographic Context Link element of the Encrypted Triplet. If the Cryptographic Context Link elements do not match, the resolution has failed and an error status shall be returned.

The Cryptographic Filter is one of only two modules in the Reference Processing Model which retains state from one Triplet to another (the other being the Index Table Generator).

9.2.2 MIC Key Derivation Module

If the optional MIC item is present in the Encrypted Triplet, the MIC Key Derivation Module shall be used to prepare a cryptographic key for use by the Encrypted Triplet Integrity Module. As depicted in Figure 7, it takes as input a Cipher Key and produces as output a MIC Key.

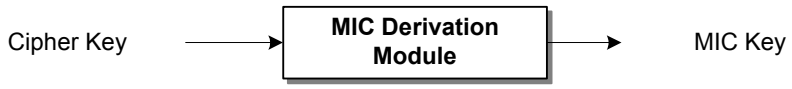

Figure 7 – MIC Key Derivation Module

The output key (MIC Key) shall be derived from the input key (CipherKey) using the algorithm defined in Section 7.10.

9.2.3 Encrypted Triplet Integrity Module

If the optional MIC 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 MIC items of the Encrypted Triplet. As depicted in Figure 8, 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 between successive Encrypted Triplets, i.e. it is memory-less.

INFORMATIVE NOTE – Elements of the MIC may be used to detect duplicated, missing, or reordered Encrypted Triplets as noted in Annex A; however this function is not included in the Reference Processing Model.

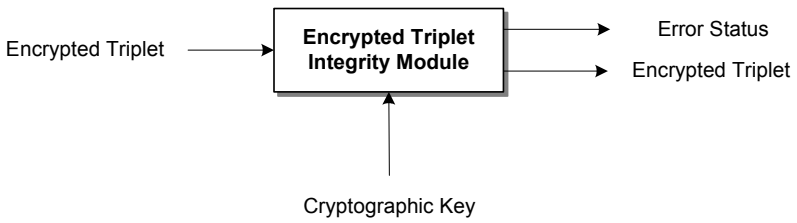

Figure 8 –Encrypted Triplet Integrity Module.

The module shall operate using the sole algorithm defined in Section 6.7, i.e. the HMAC-SHA1 algorithm. Given an input Encrypted Triplet, the module shall create its output Encrypted Triplet and Error status according to the following rules.

- The output Encrypted Triplet shall be 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 10), or if the optional MIC item is not present, no other processing shall occur. If the input Triplet is encrypted and the MIC item is present, the rules below shall apply.
- Using the input cryptographic key, the HMAC-SHA1 algorithm shall be 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 has failed and an error status shall be returned.

9.2.4 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 9, 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.

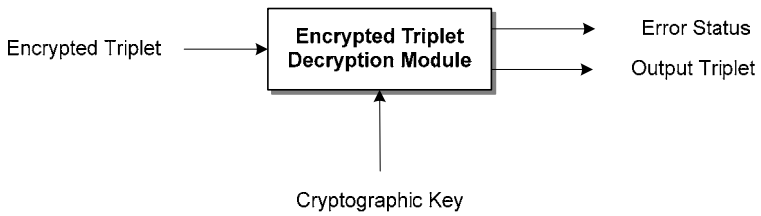

Figure 9 – Encrypted Triplet Decryption Module.

The module shall operate using the sole algorithm defined in Section 6.5, i.e. AES-128 in CBC mode [FIPS 197, NIST SP 800-38A]. Given an input Encrypted Triplet, the module shall create its output KLV Triplet according to the following rules, also illustrated in Figure 10.

- If the input Triplet is not encrypted (i.e. the K item is not the Encrypted Triplet Key of Table 10), the output Triplet shall be set equal to the input Triplet and no other processing shall occur. If the input Triplet is encrypted, the rules below shall apply.
- The output K item shall be equal to the Source Key item of the input Encrypted Triplet.
- The output L item shall be 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 the first 32 bytes of the Encrypted Source Value item shall be ignored and the following L bytes copied to the output V item, and no other processing shall occur. If the Plaintext Offset does not equal L, the following steps shall apply.
- The first 16 bytes (in network byte order) of the Encrypted Source Value item found in the input Encrypted Triplet shall be used as the Initialization Vector used 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 shall be returned and no Triplet is output.
- The next (Plaintext Offset) bytes of the Encrypted Source Value item shall be 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 shall be returned and no Triplet shall be 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 shall be returned and no Triplet shall be output.
- Subsequent groups of 16 bytes shall be processed as cipher blocks according to AES-128 in CBC mode, using the Cryptographic Key input.
- The first (L - Plaintext Offset) bytes produced by this process shall be copied to the output V item. If there have not been enough ciphertext blocks to generate (L - Plaintext Offset) bytes, then an error status shall be returned and no Triplet shall be 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.

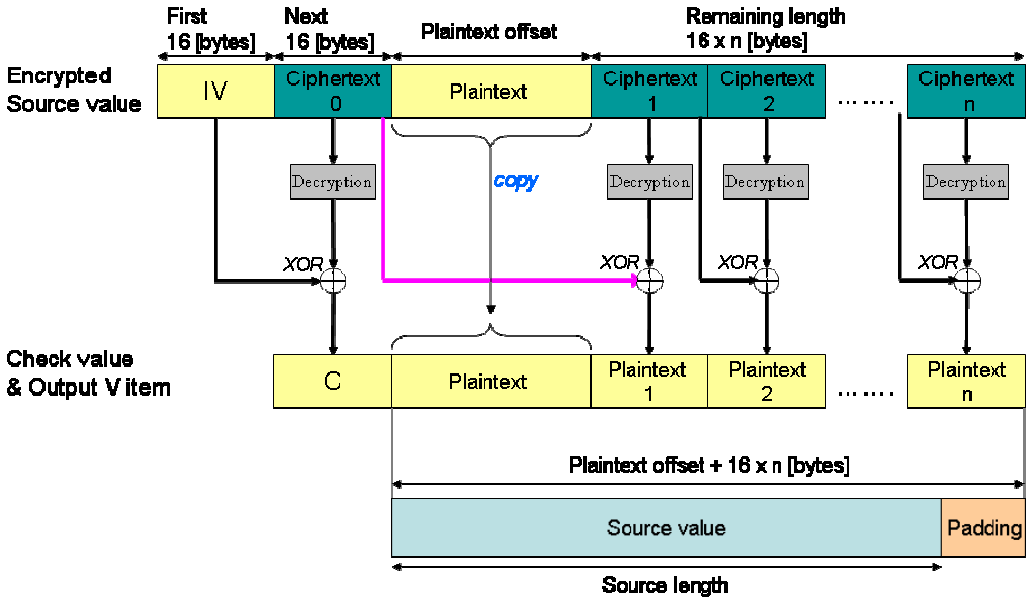

Figure 10 – Encrypted Triplet Decryption process.

9.2.5 Index Table Generation Module

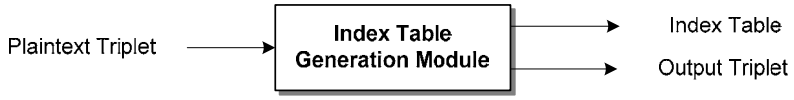

Figure 11 – Index Table Generation Module

The Index Table Generation module conceptually operates in two phases, namely operational and post-operational phases. In the operational phase, the plaintext triplets shall be passed through unmodified while positional information is extracted. In the post-operational phase, the module outputs an index table as defined in Section 8.3.

10 Label and Key Structures

10.1 Encrypted Essence Container Label

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label size	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	07h	Registry Version at the point of registration of this label
9	Item Designator	0Dh	Organizationally 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 size	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	07h	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	Encrypted Track File 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 size	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	01h	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
15,16	Reserved	00h	

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 size	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	01h	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
15,16	Reserved	00h	

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 size	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	Set/Pack Dictionary
8	Version Number	01h	Registry Version at the point of registration of this key
9	Item Designator	0Dh	Organizationally 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	

10.6 AES-CBC with 128-bit Key UL

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label size	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	07h	Registry Version at the point of registration of this label
9	Item Designator	02h	Administrative
10		09h	Encryption
11		02h	Data Encryption
12		01h	Data Encryption Algorithms
13	Algorithm Designator	01h	AES-128 CBC
14-16	Reserved	00h	

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

10.7 HMAC-SHA1 with 128-bit Key UL

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label size	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	07h	Registry Version at the point of registration of this label
9	Item Designator	02h	Administrative
10		09h	Encryption
11		02h	Data Encryption
12		02h	Data Hashing Algorithms
13	Algorithm Designator	01h	HMAC-SHA1 128
14-16	Reserved	00h	

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

Annex A (Informative)
Security Properties

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.

Annex B (Informative)
Bibliography

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

SMPTE 390M-2004, Television — Material Exchange Format (MXF) — Specialized Operational Pattern "Atom" (Simplified Representation of a Single Item)

SMPTE RP 205, Application of Unique Material Identifiers in Production and Broadcast Environments

SMPTE RP 210, Metadata Dictionary Registry of Metadata Element Descriptions

SMPTE RP 224, SMPTE Labels Registry

SMPTE EG 41-2004, Material Exchange Format (MXF) — Engineering Guideline

SMPTE EG 42-2004, Material Exchange Format (MXF) — MXF Descriptive Metadata

ISO/IEC 14496-10:2005, Information Technology — Coding of Audio-Visual Objects — Part 10: Advanced Video Coding (AVC).

Bruce Schneier, Applied Cryptography (Second Edition), Wiley, 1996