# Advanced Access Content System (AACS)

## *Technical Overview (informative)*

*Intel Corporation*
*International Business Machines Corporation*
*Matsushita Electric Industrial Co., Ltd*
*Microsoft Corporation*
*Sony Corporation*
*Toshiba Corporation*
*The Walt Disney Company*
*Warner Bros.*

*July 21, 2004*

**Preliminary Draft, Subject to Change**

This page is intentionally left blank.

# Preface

## Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. IBM, Intel, Matsushita Electric Industrial Co., Ltd , Microsoft Corporation, Sony Corporation, Toshiba Corporation, The Walt Disney Company and Warner Bros. disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

This document is a preliminary draft and is subject to change without notice. Readers should not design products based on this document.



## Intellectual Property

Implementation of the specification described by this informative overview requires a license from AACS LA.

## Contact Information

The URL for the AACS LA web site is http://www.aacsla.com.

This page is intentionally left blank.

# Table of Contents

# List of Figures

This page is intentionally left blank.

# List of Tables

This page is intentionally left blank.

# Chapter 1
# Introduction

## 1.

## 1.1 Purpose and Scope

The Advanced Access Content System (AACS) technology provides an advanced, robust and renewable method for protecting audiovisual entertainment content, including high-definition content. This document, the *AACS Technical Overview*, describes the overall goals of AACS, cryptographic procedures that are common among its various defined uses, and an overview of its use for protection on pre-recorded and recordable optical storage media. The AACS Specification, when published, will be the normative document for implementing AACS, and will be comprised of separate books, providing content protection details specific to particular optical media types and formats. Those under development and finalization at or around the time of this publication are:

- Introduction and Common Cryptographic Elements
- Prerecorded Video Book (format-independent)
- Recordable Video Book (format-independent)

Books covering specific optical media formats are expected to be available in the future.

The use of the AACS specification and access to the intellectual property and cryptographic materials required to implement it will be the subject of a license. A license authority referred to as AACS LA is responsible for establishing and administering the content protection system based in part on the specification.

## 1.2 Objectives and Design Criteria

AACS is designed to meet the following general criteria:

- Meet the content owners' requirements for robustness and system renewability
    - Content encryption based on a published cryptographic algorithm.
    - Limit access to protected content to only licensed compliant implementations.
    - Support revocation of individual compromised devices' keys.
    - Limit output and recording of protected content to a list of approved methods.
- Suitable for implementation on both general-purpose computer and fixed-function consumer electronics platforms.
- Applicable to both audio and video content, including high-definition video.
- Applicable to various optical media formats.
- Transparent to authorized use by consumers.

To meet these general objectives, AACS is based in part on the following technical elements:

- Robust encryption of protected content using the AES cipher.
- Key management and revocation using advanced Media Key Block technology.

Additional specific criteria and technical elements apply to particular uses of AACS, as described in the specification.

## 1.3 Organization of this Document

This document is organized as follows:

- Chapter 1 provides an introduction.
- Chapter 2 describes core cryptographic functions, based on the AES cipher algorithm.
- Chapter 3 describes a cryptographic key management procedure using the Media Key Block.
- Chapter 4 provides an overview of the encryption and decryption processes.
- Chapter 5 describes procedures required for drive-host configurations
- Chapter 6 illustrates methods of utilizing online connections within AACS

## 1.4 References

This document references the following publications. When the publications are superceded by an approved revision, the revision shall apply.

AACS LA, *License agreement (not currently available)*

ANSI X9.31-1998, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA),* ANSI, September 9, 1998. (Informative)

National Institute of Standards and Technology (NIST), *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Special Publication 800-22, with revisions dated May 15, 2001.

National Institute of Standards and Technology (NIST), *Recommendation for Block Cipher Modes of Operation Methods and Techniques*, NIST Special Publication 800-38A, 2001 Edition

National Institute of Standards and Technology (NIST), *Secure Hash Standard*, FIPS Publication 180-2, August 1, 2002.

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

RSA Laboratories, *PKCS #1 (v2.1): RSA Cryptography Standard*, June 14, 2002.

## 1.5 Notation

### 1.5.1 Numerical Values

This document uses three different representations for numerical values. Decimal numbers are represented without any special notation. Binary numbers are represented as a string of binary (0, 1) digits followed by a subscript 2 (e.g., $1010_2$). Hexadecimal numbers are represented as a string of hexadecimal (0..9, A..F) digits followed by a subscript 16 (e.g., $A3C2_{16}$).

### 1.5.2 Bit and Byte Ordering

Certain data values or parts of data values are interpreted as an array of bits. Unless explicitly noted otherwise, bit positions within an n-bit data value are numbered such that the least significant bit is numbered 0 and the most significant bit is numbered n-1.

Unless explicitly noted otherwise, big-endian ordering is used for multiple-byte values, meaning that byte 0 is the most significant byte.

### 1.5.3 Operations

The following notation will be used for bitwise and arithmetic operations:

| | |
|---|---|
| $[x]_{msb\_z}$ | The most significant z bits of x. |
| $[x]_{lsb\_z}$ | The least significant z bits of x. |
| $[x]_{y:z}$ | The inclusive range of bits between bit y and bit z in x. |
| ~x | Bit-wise inversion of x. |
| x \|\| y | Ordered concatenation of x and y. |
| x ⊕ y | Bit-wise Exclusive-OR (XOR) of two strings x and y. |
| x + y | Modular addition of two strings x and y. |
| x × y | Multiplication of x and y. |
| x – y | Subtraction of y from x. |

The following assignment and relational operators will be used:

| | |
|---|---|
| = | Assignment |
| == | Equal to |
| != | Not equal to |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |

## 1.6 Abbreviations and Acronyms

The following is an alphabetical list of abbreviations and acronyms used in this document:

| | |
|---|---|
| AACS | Advanced Access Content System |
| AACS LA | AACS Licensing Administrator, LLC |
| AES | Advanced Encryption Standard |
| CBC | Cipher Block Chaining |
| CE | Consumer Electronics |
| ECB | Electronic Codebook |
| FIPS | Federal Information Processing Standards |
| MKB | Media Key Block |
| NIST | National Institute of Standards and Technology |
| PC | Personal Computer |

This page is intentionally left blank.

# Chapter 2
# Common Cryptographic Functions

## 2. Introduction

This chapter describes common cryptographic functions upon which protection mechanisms described in the AACS specification are based. The functions are described here in isolation; their specific uses as part of encryption, key management and renewability mechanisms are described elsewhere in this document, as well as in other books of the specification.

### 2.1 AES Symmetric Block Cipher Algorithm

Symmetric cryptographic functions are based on the Advanced Encryption Standard (AES) block cipher algorithm, as specified in FIPS Publication 197 (see reference in Section 1.4). Unless otherwise specified, the AES algorithm is used with data blocks of 128 bits and keys with lengths of 128 bits.

#### 2.1.1 ECB Mode (AES-128E and AES-128D)

For purposes such as management of cryptographic keys, the AES cipher is used with the Electronic Codebook (ECB) mode of operation specified in NIST Special Publication 800-38A (see reference in Section 1.4).

Hereafter, encryption using the AES algorithm in ECB mode as just described is represented by the function

AES-128E(k, d)

where d is a 128-bit data value to be encrypted, k is a 128-bit key, and AES-128E returns the 128-bit encryption result.

Decryption using the AES algorithm in ECB mode as described above (using either the “inverse cipher” or “equivalent inverse cipher” specified in FIPS Publication 197) is represented by the function

AES-128D(k, d)

where d is a 128-bit data value to be decrypted, k is a 128-bit key, and AES-128D returns the 128-bit decryption result.

#### 2.1.2 CBC Mode (AES-128CBCE and AES-128CBCD)

For purposes such as encryption and decryption of protected content, the AES cipher is used with the Cipher Block Chaining (CBC) mode of operation specified in NIST Special Publication 800-38A (see reference in Section 1.4).

Hereafter, encryption using the AES algorithm in CBC mode as just described is represented by the function

AES-128CBCE(k, d)

where d is a frame of data to be encrypted, k is a 128-bit key, and AES-128CBCE returns the encrypted frame.

Decryption using the AES algorithm in CBC mode as described above (using either the “inverse cipher” or “equivalent inverse cipher” specified in FIPS Publication 197) is represented by the function

AES-128CBCD(k, d)

where d is a frame of data to be decrypted, k is a 128-bit key, and AES-128CBCD returns the encrypted frame.

The size of the frame of data to be encrypted or decrypted (i.e. how often a new CBC chain is started) depends on the particular application, and is defined for each in the corresponding books of the specification. Unless otherwise specified, the Initialization Vector used at the beginning of a CBC encryption or decryption chain is a constant, $iv_0$, specified by AACS LA.

### 2.1.3 AES-based One-way Function (AES-G)

AACS uses a cryptographic one-way function based on the AES algorithm. This function is referred to as the AES-based One-way Function, and is represented by

AES-G($x_1$, $x_2$)

where $x_1$ and $x_2$ are 128-bit input values, and AES-G($x_1$, $x_2$) returns the 128-bit result.

Figure 2-1 depicts the AES-based One-way Function.

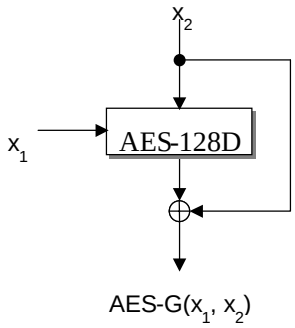


**Figure 2-1 – AES-based One-way Function**

The AES-based One-way Function result is calculated as

AES-G($x_1$, $x_2$) = AES-128D($x_1$, $x_2$) $\oplus$ $x_2$.

### 2.1.4 AES-based Hashing Function (AES-H)

For the purpose of processing data to produce a condensed representation, a hashing procedure based on the AES algorithm is used. This procedure, referred to as the AES-based Hashing Function, is represented by

AES-H(x)

where x is input data of arbitrary length, and AES-H(x) returns the corresponding 128-bit hash value.

Prior to hashing, the data to be hashed (x) is padded by first appending a single '1' bit, and then appending zero or more '0' bits until the length of the padded data (x') is a multiple of 128 bits. By way of example, a 126-bit data value x is padded to a 128-bit value x' ending in $10_2$, a 127-bit data value x is padded to a 128-bit value x' ending in $1_2$, and a 128-bit data value x is padded to a 256-bit value x' ending in $80000000000000000000000000000000_{16}$.

The padded data x' is divided into n 128-bit blocks, represented as $x'_1, x'_2, \ldots, x'_n$, which are used in calculating the hash as shown in Figure 2-2.

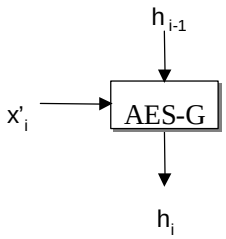


**Figure 2-2 – AES-based Hashing Function**

A 128-bit confidential initial value $h_0$ is provided to licensees by AACS LA. For i from 1 to n, the 128-bit value $h_i$ is calculated as

$$h_i = \text{AES-G}(x'_i , h_{i-1}).$$

The value $h_n$ is the result of the hashing function, i.e., AES-H(x) = $h_n$.

### 2.1.5 AACS Message Authentication Code (AACS-MAC)

AACS will select and utilize a cryptographic message authentication code.

## 2.2 Random/Pseudorandom Number Generation

This section describes a random/pseudorandom number generator based on a design described in ANSI X9.31 (see reference in Section 1.4), for use in generating values such as cryptographic keys. Unless specifically noted otherwise, the generator illustrated in Figure 2-3 and described below, or a design of equal or higher quality that passes the tests described in NIST Special Publication 800-22 when using the default parameters and other recommendations provided therein (see reference in Section 1.4), shall be used.

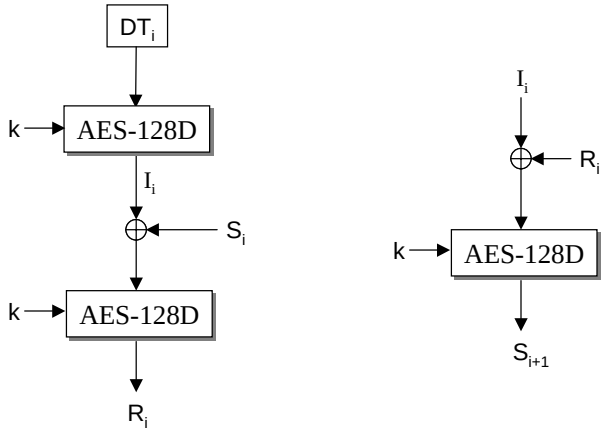


**Figure 2-3 – Random/pseudorandom Number Generator Example**

In the figure, k is a 128-bit constant value, DT is a 128-bit value that is updated on every iteration (such as a date/time vector or monotonic counter) and S is a seed value.

In one alternative, the initial seed value $S_0$ shall be obtained from an unpredictable source of run-time entropy, in which case the fixed value k need not be unique per licensed product, and DT must be ensured to be non-repeating only until another initial seed value ($S_0$) is obtained.

In another alternative, the combination of fixed value k and initial seed value $S_0$ shall be unpredictable and unique per licensed product, and S shall be maintained in a non-volatile register, in which case a source of entropy is not required and DT must be ensured to be non-repeating only until the next time the licensed product is re-started.

For both alternatives, 128-bit random values $R_i$ (i=0,1,…) are generated via the following calculations:

$I_i$ = AES-128D(k, $DT_i$),

$R_i$ = AES-128D(k, $I_i \oplus S_i$), and

$S_{i+1}$ = AES-128D(k, $I_i \oplus R_i$).

Unless explicitly noted otherwise, the values k and S shall be treated as highly confidential as described in the license agreement.

## 2.3 Digital Signatures

All digital signatures in AACS utilize the RSASSA-PSS digital signature scheme defined in PKCS#1(v2.1) (see reference in Section 1.4). This standard requires the implementer to fix two primitives: a hash function, and a mask generation function. AACS uses SHA-1 defined in FIPS 180-2 (see reference Section 1.4) as the hash function and MGF1 defined in section B.2.1 of the PKCS#1(v2.1) standard as the mask generation function. The RSAVP1 operations will use RSA with a 2048-bit modulus and a public exponent (e) of 65537.

AACS devices are not required to generate digital signatures.

# Chapter 3
# Media Key Block

## 3. Introduction

This chapter describes an advanced cryptographic key management procedure, depicted in Figure 3-1, which uses a Media Key Block, based on the subset-difference tree method, to provide system renewability in the form of device revocation. The procedure is described here in isolation; its use as part of the overall protection system is described elsewhere in this document.

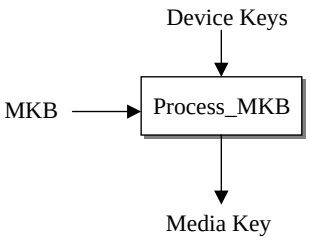


**Figure 3-1 – Common AACS Cryptographic Key Management Procedure**

Device Keys ($K_{d\_0}$,$K_{d\_1}$,…,$K_{d\_n-1}$) are used to decrypt one or more elements of a Media Key Block (MKB), in order to extract a secret Media Key ($K_m$). Table 3-1 lists the elements involved in this process, along with their sizes.

**Table 3-1 – Common Cryptographic Key Management Elements**

| Key or Variable | Size |
|---|---|
| Device Keys ($K_{d\_0}$,$K_{d\_1}$,…,$K_{d\_n-1}$) | 128 bits each |
| Media Key Block (MKB) | Variable, multiple of 4 bytes |
| Media Key ($K_m$) | 128 bits |

The remainder of this section describes this cryptographic key management procedure in detail.

## 3.1 Device Keys

Each compliant device or application is given a set of secret Device Keys when manufactured. The actual number of keys may be different in different media types. These Device Keys, referred to as $K_{di}$ (i=0,1,…,n-1), are provided by AACS LA, and are used by the device or application to process the MKB to calculate $K_m$. The set of Device Keys may either be unique per device, or used commonly by multiple devices. The license agreement describes details and requirements associated with these two alternatives. A device shall treat its Device Keys as highly confidential, as defined in the license agreement.

## 3.2 Media Key Block (MKB)

The Media Key Block (MKB) enables system renewability. The MKB is generated by AACS LA, and allows all compliant devices, each using their set of secret Device Keys, to calculate the same $K_m$. If a set of Device Keys is compromised in a way that threatens the integrity of the system, an updated MKB can be released that causes a device with the compromised set of Device Keys to be unable to calculate the correct $K_m$. In this way, the compromised Device Keys are "revoked" by the new MKB.

### 3.2.1 Calculation of Subsidiary Device Keys and Processing Keys

For the purpose of processing an MKB to calculate $K_m$, Device Keys are used to calculate subsidiary Device Keys and Processing Keys using the AES-G3 function depicted in Figure 3-2.

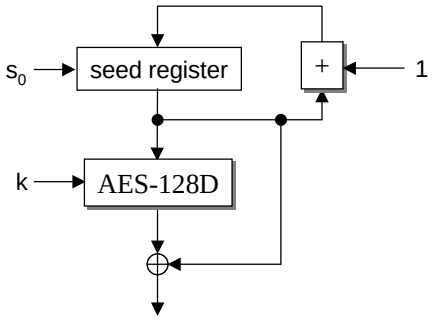


**Figure 3-2 – Triple AES Generator (AES-G3)**

A 128-bit input Device Key (which may be a subsidiary Device Key) is denoted 'k' in this diagram. This loop is executed three times to produce 384 output bits, incrementing the seed register by one each time. The output of AES-128D is XORed with the seed register's output at each step. For each AES-G3 calculation, the seed register is initialized by the 128-bit value "$s_0$", which is provided by the licensing agency. A device shall keep the value $s_0$ confidential, as defined in the license agreement.

The 384 output bits are interpreted as follows:

1. The first 128 bits is the subsidiary Device Key for the left child of the current node, or it is ignored if the Device Key 'k' is a leaf Device Key, AES-128D(k, $s_0$) $\oplus$ $s_0$.
2. The second 128 bits is the Processing Key, AES-128D(k, $s_0$+1) $\oplus$ ($s_0$+1).
3. The third 128 bits is the subsidiary Device Key for the right child of the current node, or it is ignored if the Device Key 'k' is a leaf Device Key, AES-128D(k, $s_0$+2) $\oplus$ ($s_0$+2).

By using the AES-G3 function in this way, the device can calculate all of the subsidiary Device Keys it needs from the few Device Keys it stores at manufacturing time.

### 3.2.2 Storing Device Keys

Each device is given its Device Keys and a 31-bit number *d* called the *device node number*. For each Device Key, there is an associated number called the *uv* number, and two bit masks, called $m_u$ and $m_v$. The *uv* number denotes the position in the tree associated with the Device Key. It is a path from the root to that node in the tree, 0 indicating 'left' and 1 indicating 'right', starting with the most significant bit. The bit masks indicate "don't care" bits in the *uv* number; if a bit is zero, that corresponding bit in the *uv* number is "don't care"; i.e., the path ends at this point. The masks are always a single sequence of 1-bits followed by a single sequence of 0-bits. The device node number, *uv* number, and masks denote nodes within a binary tree, where *u* is an ancestor of *v*. These masks represent the depth of the respective nodes, u and v, from the root of the tree. As a result, the $m_u$ mask always has more 0 bits than the $m_v$ mask. The subset-difference is the subtree rooted at node *u* minus the subtree rooted at node *v*.

### 3.2.3 Calculation of Media Key

The Media Key Block includes two major parts: the subset-difference identification part, and the key data part. For each subset-difference included in the identification part, there are 16 bytes of key data in the key data part. The key data is one-for-one with the identified subset-differences. For example, the 23rd subset-difference is associated with the 23rd section of the Media Key Data field; that is, it begins at offset (23-1)*16 from the start of the Media Key Data field of the Media Key Data Record.

Subset-differences are encoded as *uv* numbers and two masks, a "u" mask $m_u$ and a "v" mask $m_v$. A subset-difference applies to a device if the *u* node is on a path from the device's node to the root of the tree, but the *v* node is not. This is simple to calculate using the *uv* number, the appropriate mask, and the device node number *d*. A device is on a path to a *uv* number with mask *m* if and only if:

$$(d \,\&\, m) == (uv \,\&\, m)$$

Thus, a subset-difference applies if and only if:

$$((d \,\&\, m_u) == (uv \,\&\, m_u)) \text{ and } ((d \,\&\, m_v) != (uv \,\&\, m_v))$$

To put this test into words, this tests that the device is in the subset (its node is in the subtree rooted in *u*), and it is not in the difference (its node is not in the subtree rooted in *v*).

The device searches through the Explicit Subset-Difference Record fields, looking at the identified subset-differences, until it finds the one that applies to it. At that point the device will either have the Device Key, or will be able to derive the subsidiary Device Key, associated with that subset-difference. It finds the appropriate stored Device Key as follows: assuming the Explicit Subset-Difference Record value is uv, $m_u$, and $m_v$, and the stored Device Key has uv', $m'_u$, and $m'_v$, the appropriate Device Key is the one that meets the following condition:

$$(m_u == m'_u) \text{ and } ((uv \,\&\, m'_v) == (uv' \,\&\, m'_v))$$

If $m'_v$ equals $m_v$, the starting Device Key is the final Device Key, and can be used directly to derive the Processing Key, as described above. Usually, however, the starting Device Key's node is further up in the tree, and the actual Device Key will have to be derived. The device does that as follows:

1. *Initialization*. m = the stored v mask $m'_v$. D = the starting Device Key.
2. Use AES-G3 on D, as described above, to determine a left subsidiary Device Key, a Processing Key, and a right subsidiary Device Key.
3. Look at the most significant zero bit in *m*. If the corresponding bit in the incoming *uv* number is 0, D = left subsidiary Device Key from step 2. Otherwise, D = right subsidiary Device Key from step 2.
4. *Iteration*. Arithmetic shift *m* right one bit. If it does not equal the incoming v mask $m_v$, repeat starting at step 2.

Once the device has the correct Device Key D, it calculates a Processing Key K using AES-G3 as described above. Using that Processing Key K and the appropriate 16 bytes of encrypted key data C, the device calculates the 128-bit Media Key $K_m$ as follows:

$K_m = AES\text{-}128D(K, C) \oplus (00000000_{16} \,||\, uv)$

The appropriate encrypted key data C is found in the Media Key Data Record in the Media Key Block.

A device may discover, while processing the Media Key Block, that none of the subset-differences identified in the block apply to it. In that case, the device shall conclude that it is revoked. Device behavior in this situation is implementation defined. As an example, a device could exhibit a special diagnostic code, as information to a service technician.

### 3.2.4 Media Key Block Format

A Media Key Block is formatted as a sequence of contiguous Records. Each Record begins with a one-byte Record Type field, followed by a three-byte Record Length field. The Record Type field value indicates the type of the Record, and the Record Length field value indicates the number of bytes in the Record, including the Record Type and the Record Length fields themselves. Record lengths are always multiples of 4 bytes.

Using its Device Keys, a device calculates $K_m$ by processing Records of the MKB one-by-one, in order, from first to last. The device must not make any assumptions about the length of Records, and must instead use the Record Length field value to go from one Record to the next. If a device encounters a Record with a Record Type field value it does not recognize, that is not an error; it shall ignore that Record and skip to the next. Likewise, if a Record Length indicates a record is longer than the device expects, that is also not an error; it shall ignore the additional record data.

The following subsections describe the currently defined Record types, and how a device processes each. All multi-byte integers, including the length field, are “Big Endian”; in other words, the most significant byte comes first in the record.

#### 3.2.4.1 Verify Media Key Record

**Table 3-2 – *Verify Media Key* Record Format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Record Type: $81_{16}$ | | | | | | | |
| 1 | Record Length: $000014_{16}$ | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | Verification Data ($D_v$) | | | | | | | |
| … | | | | | | | | |
| 19 | | | | | | | | |

A properly formatted MKB shall have exactly one *Verify Media Key* Record as its first Record. Bytes 4 through 19 of the Record contain the ciphertext value

$C_v = AES\text{-}128E (K_m, 0123456789ABCDEF_{16} \,||\, XXXXXXXXXXXXXXXX_{16})$

where $XXXXXXXXXXXXXXXX_{16}$ is an arbitrary 8-byte value, and $K_m$ is the correct final Media Key value.

The presence of the *Verify Media Key* Record in an MKB is mandatory, but the use of the Record by a device is not mandatory. The device may use the *Verify Media Key* Record to verify the correctness of a given MKB, or of its processing of it. If everything is correct, the device should observe the condition:

$[AES\_128D({}_vK_m, C)]_{msb\_64} == 0123456789ABCDEF_{16}$

where $K_m$ is the Media Key value.

### 3.2.4.2 Type and Version Record

**Table 3-3 – *Type and Version* Record Format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Record Type: $10_{16}$ | | | | | | | |
| 1 | Record Length: $00000C_{16}$ | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | MKBType: $00031003_{16}$ | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | Version Number | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |

Devices, except for recording devices which are writing Media Key Block Extensions (see 3.2.5), may ignore this record. Recording devices shall verify the signature (see End of Media Key Block record) and use the Version Number in this record to determine if a new Media Key Block Extension is, in fact, more recent than the Media Key Block Extension that is currently on the media. The Version Number is a 32-bit unsigned integer. Each time the licensing agency changes the revocation, it increments the version number and inserts the new value in subsequent Media Key Blocks. Thus, larger values indicate more recent Media Key Blocks. The Version Numbers begin at 1; 0 is a special value used for test Media Key Blocks.

Any device depending on the Version Number shall check the signature on the Media Key Block (see 3.2.4.6) to determine if the Version Number has been tampered with.

For AACS applications, the type field is always $00031003_{16}$. Devices shall ignore this.

### 3.2.4.3 Explicit Subset-Difference Record

**Table 3-4 – *Explicit Subset-Difference* Record Format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Record Type: $04_{16}$ | | | | | | | |
| 1 | Record Length | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | U Mask (0) | | | | | | | |
| 5 | UV Number (0) | | | | | | | |
| … | | | | | | | | |
| 8 | | | | | | | | |
| 9 | U Mask (1) | | | | | | | |
| 10 | UV Number (1) | | | | | | | |
| ... | | | | | | | | |
| 13 | | | | | | | | |
| 14 | . | | | | | | | |
| . | . | | | | | | | |
| . | . | | | | | | | |
| Length-1 | | | | | | | | |

In this record, each subset-difference is encoded with 5 bytes. The mask for u is given by the first byte. That byte is treated as a number, the number of low-order 0-bits in the mask. For example, the value $01_{16}$ denotes a mask of $FFFFFFFE_{16}$; value $0A_{16}$ denotes a mask of $FFFFFC00_{16}$.

The last 4 bytes are the *uv* number, most significant byte first.

The mask for *v* is given by the first lower-order 1-bit in the *uv* number. That bit, and all lower-order 0-bits, are zero bits in the mask. The following C code fragment illustrates one way to calculate the *v* mask from the *uv* value:

```
long v_mask = 0xFFFFFFFF;
while ((uv & ~v_mask) == 0) v_mask <<= 1;
```

The length of this record will always be a multiple of 4 bytes. Thus, there may be unused bytes at the end of the record.

### 3.2.4.4 Subset-Difference Index Record

**Table 3-5 – *Subset-Difference Index* Record Format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Record Type: $07_{16}$ | | | | | | | |
| 1 | Record Length | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | Span (number of devices) | | | | | | | |
| … | | | | | | | | |
| 7 | | | | | | | | |
| 8 | Offset 0 | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | Offset 1 | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | . | | | | | | | |
| … | . | | | | | | | |
| Length-1 | . | | | | | | | |

This is a speed-up record which can be ignored by devices not wishing to take advantage of it. It is a lookup table which allows devices to quickly find their subset-difference in the Explicit Subset-Difference record, without processing the entire record. This Subset-Difference Index record is always present, and always precedes the Explicit Subset-Difference record in the MKB, although it does not necessarily immediately precede it.

This record contains a "span", the number of devices per index offset, and a number of 3-byte offsets. These offsets refer to the offset within the following Explicit Subset-Difference record, with 0 being the start of the record. Devices whose device number is between 0 and span-1 should begin processing the Explicit Subset-Difference record at Offset 0. Devices whose number is between span and 2*span-1, should begin processing the Explicit Subset-Difference record at Offset 1, and so on. Equivalently, if a device's number is D, it can find its offset within the Explicit Subset-Difference record at offset 3*D/span + 8 in this record.

The length of this record will always be a multiple of 4 bytes. Thus, there may be unused bytes at the end of the record.

### 3.2.4.5 Media Key Data Record

**Table 3-6 – *Media Key Data* Record Format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Record Type: $05_{16}$ | | | | | | | |
| 1 | Record Length | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | Media Key Data (0) | | | | | | | |
| … | | | | | | | | |
| 19 | | | | | | | | |
| 20 | Media Key Data (1) | | | | | | | |
| ... | | | | | | | | |
| 35 | | | | | | | | |
| 36 | . | | | | | | | |
| . | . | | | | | | | |
| . | . | | | | | | | |
| Length-1 | | | | | | | | |

This record gives the associated encrypted media key data for the subset-differences identified in the Explicit Subset-Difference Record. Each subset difference has its associated 16 bytes in this record, in the same order it is encountered in the subset-difference record. This 16 bytes is the ciphertext value C in the media key calculation in section 3.2.3

The Explicit Subset-Difference Record always precedes this record, although it may not immediately precede it.

The length of this record is always a multiple of 4 bytes.

### 3.2.4.6 End of Media Key Block Record

**Table 3-7 – *End of Media Key Block* Record Format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Record Type: $02_{16}$ | | | | | | | |
| 1 | Record Length | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | Signature Data | | | | | | | |
| … | | | | | | | | |
| Length-1 | | | | | | | | |

A properly formatted MKB shall contain an *End of Media Key Block* Record. When a device encounters this Record it stops processing the MKB, using whatever $K_m$ value it has calculated up to that point as the final $K_m$ for that MKB (pending possible checks for correctness of the key, as described previously).

The End of Media Key Block record contains the AACS LA's signature on the data in the Media Key Block up to, but not including, this record. Devices depending on the Version Number in the *Type and Version* Record must verify the signature. Other devices may ignore the signature data. If any device determines that the signature does not verify or is omitted, it must refuse to use the Media Key.

The length of this record is always a multiple of 4 bytes.

## 3.2.5 MKB Extension

In processing recordable media (as opposed to pre-recorded media) there may actually be two Media Key Blocks: the Base Media Key Block and a Media Key Block Extension. The MKB Extension is a simple read/write file, with a name defined in the specific AACS book that covers the given media type (and/or application type). Each application could define its own MKB Extension, so a single piece of media might have multiple MKB Extensions. The purpose of the MKB Extension is to provide more recent revocation information than might be present in the Base MKB.

Writing of an MKB Extension onto the media by a recorder may not be required in all cases. However, all recorders and players must recognize and use an MKB Extension, if one is present, as part of the Media Key calculation. The device reads and processes both the Base MKB and MKB Extension, and does a bit-for-bit XOR of the two resulting keys. The result is the final Media Key.

In general, if a device writes an MKB Extension, it must update the encrypted title keys of all existing content on the media that correspond to that particular MKB Extension.

A device must use the Version Number in the MKB to determine if it will write or replace an MKB Extension on a piece of media. It must check the signature on both the new MKB Extension, and the existing MKB Extension (or Base MKB, where an MKB Extension is not already present). As indicated in section3.2.4.6, a device shall not use a Media Key corresponding to an MKB with an invalid signature for any purpose. Therefore, if the signature of the new MKB Extension is omitted or does not successfully verify, the new MKB Extension shall not be written onto the media. If the signature of the existing MKB Extension is omitted or does not verify, it is the device's choice whether to write a new MKB Extension. However, the device shall not update title keys of any existing content on the media protected using the existing (tampered) MKB Extension, nor shall it protect any new content using that existing MKB Extension.

This page is intentionally left blank.

# Chapter 4
# Content Encryption and Decryption

## 4. Introduction

This chapter describes the procedures for encryption and decryption of pre-recorded video content protected by AACS. Figure 4-1presents an overview, and the remainder of the chapter describes the encryption and decryption processes.

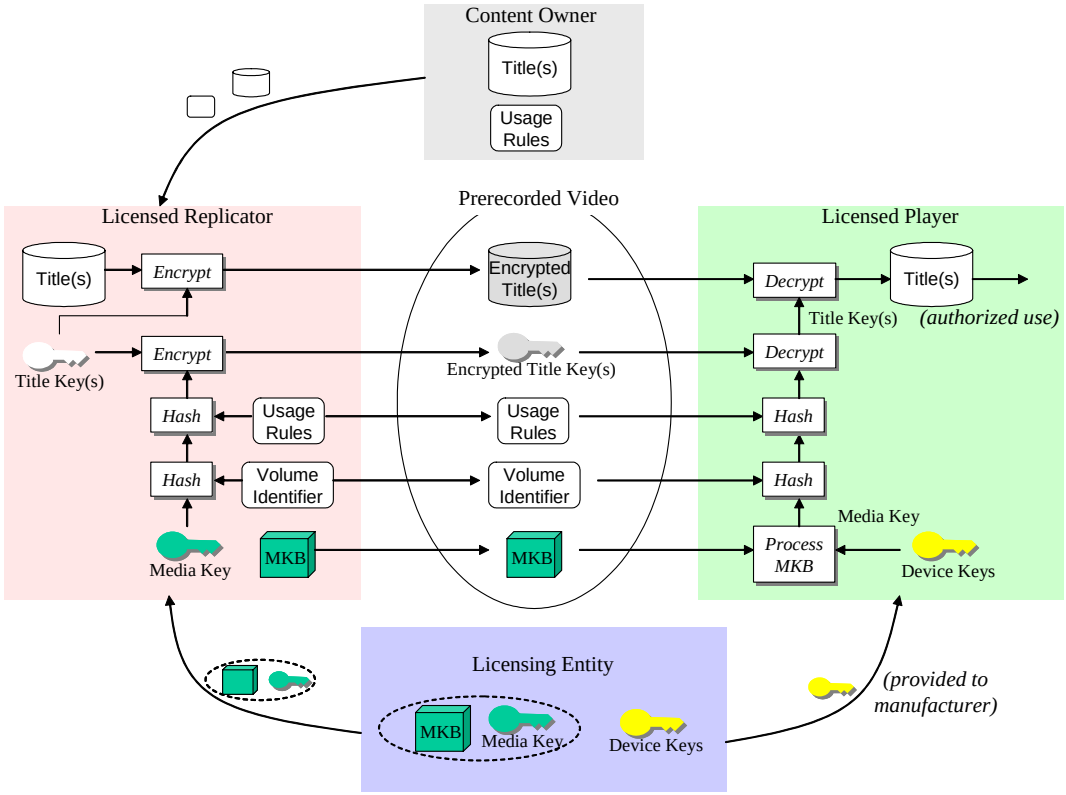


**Figure 4-1– Encryption and Decryption Overview**

### 4.1 Content Encryption (General)

The owner of content that is to be protected provides the content in the form of one or more titles, and their associated Usage Rules, to the licensed replicator.

The licensed replicator shall select a secret, random Title Key for each title to be protected. Each Title Key shall be used to encrypt the content of its corresponding title, as specified for each supported content format elsewhere in the specification. At the replicator's discretion, a given title may be encrypted using the same Title Key for all instances of pre-recorded media, or different Title Keys may be used for different instances.

The licensed replicator shall also assign a secret, unpredictable (e.g., random) identifier to the protected title or set of protected titles to be included together on a pre-recorded medium. This identifier, referred to as the Volume Identifier, is used as a safeguard against "bit-by-bit copying" of protected titles, and is therefore stored on the pre-recorded medium in a manner that cannot be duplicated by consumer recorders, as specified for each

supported storage format elsewhere in the specification. At the licensed replicator's discretion, the same Volume Identifier may be used for all instances of pre-recorded media containing a given protected title or set of protected titles, or different values may be assigned for different instances.

For each protected title or set of protected titles to be included together on a pre-recorded medium, the licensing entity provides to the licensed replicator a Media Key Block (MKB), and a corresponding secret Media Key. The Media Key Block will enable all compliant devices, each using their set of secret Device Keys, to calculate the same Media Key as described in the *Introduction and Common Cryptographic Elements* book of the specification. If a set of Device Keys is compromised in a way that threatens the integrity of the system, an updated MKB can be released that will cause a device with the compromised set of Device Keys to calculate a different Media Key than is computed by the remaining compliant devices. In this way, the compromised Device Keys are "revoked" by the new MKB

For each protected title, the licensed replicator calculates a cryptographic hash of the Media Key , the Volume ID and the title's Usage Rules, and uses the result to encrypt the title's Title Key. The encrypted content, Encrypted Title Key(s), Usage Rules and MKB are stored on the pre-recorded medium as specified for each supported storage/content format elsewhere in the specification.

## 4.2 Content Decryption (General)

The licensing entity provides a set of n secret Device Keys, denoted $K_{d\_0}$,$K_{d\_1}$,…,$K_{d\_n-1}$, to the licensed manufacturer for inclusion into each compliant device or application produced. Device Key sets may either be unique per licensed product, or used commonly by multiple products; the license agreement describes the details and requirements associated with these two alternatives. A licensed product shall treat its Device Keys as highly confidential, as defined in the license agreement.

The licensed product reads the MKB from the pre-recorded medium, and uses its Device Keys to process the MKB and thereby calculate the Media Key, as described in the *Introduction and Common Cryptographic Elements* book of the specification. If the given set of Device Keys has not been revoked, then the calculated Media Key will be the same Media Key that was used by the licensed replicator as described above.

For each protected title the licensed product then calculates a cryptographic hash of the calculated Media Key , the Volume Identifier and the title's Usage Rules, and uses the result to decrypt the title's Encrypted Title Key. The resulting Title Key is then used to decrypt the title, as specified for each supported format elsewhere in the specification.

In a system using a drive-host configuration (e.g., a PC), the Volume Identifier is accessed using a drive authentication protocol, as described in the *Introduction and Common Cryptographic Elements* book of the specification.

# Chapter 5
# Additional Procedures for Drive-Host Configurations

## 5. Introduction

The AACS specification is applicable to a PC based system. In such a system, a drive and PC host act together as the Recording Device and/or Playback Device for AACS protected content. Note that a new, robust and renewable form of drive authentication is introduced; recording or playback of AACS protected content is *not* permitted using drives that only support authentication associated with the Content Scramble System (CSS) for DVD-Video. The procedure for recording or playback of the content is the same as described in relevant documents of the AACS specification, except for additional steps that are required to read the Volume Identifier and to enable the host to verify the integrity of the Pre-recorded Media Serialization Number from the medium in the case of pre-recorded media, to enable the host to verify the integrity of the Media Identifier values it receives from the drive in the case of recordable media, and to manage the Protected Area Data of recordable media securely between the drive and the host. Figure 5-1 illustrates this procedure for reading the Volume Identifier.

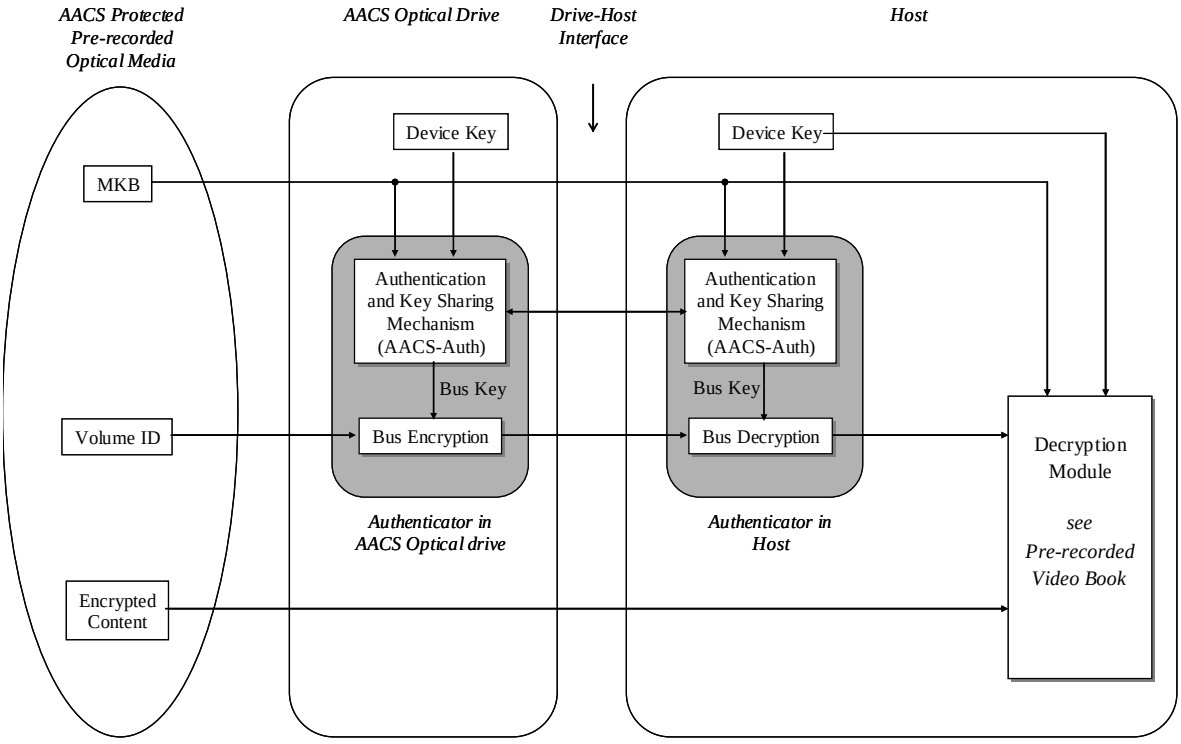


**Figure 5-1 – Reading of Volume Identifier in a Drive-Host Configuration**

# Chapter 6
# Uses of On-line Connections

## 6. Introduction

This section provides an overview of the use of an on-line connection to enable certain enhanced uses of AACS-protected content. AACS defines several different modes for using AACS content with on-line connections:

- "AACS Network Download Content". This online content is intended to be recorded on AACS-protected media. An online transaction serves to bind the content to a particular piece of media.
- "AACS Online Enabled Content". This content is pre-recorded on pre-recorded media, or part of the initial download in AACS Network Download Content, but only made playable by an online transaction.
- "AACS Streamed Content". This is stream content logically associated with pre-recorded or AACS Network Download Content, but delivered on demand across the Internet.

In addition, content owners can authorize uses of AACS content in other, unspecified ways as part of an online transaction. As a rule, if the transaction requires the use of AACS-defined keys such as device keys or media keys, the AACS specification applies, and the specification defines a minimum support level for that transaction in every AACS device. If, on the other hand, the transaction is accomplished using non-AACS keys such as title keys, the transaction is outside of the scope of the AACS specification, and it is not mandatory for AACS devices to support it.