

# 1 IEEE P1619.1™/D20 2 Draft Standard for Authenticated Encryption with 3 Length Expansion for Storage Devices

4 Prepared by the Security in Storage Working Group of the  
5 IEEE Computer Society Information Assurance Committee with joint sponsorship from the IEEE  
6 Computer Society Storage Systems Committee

7 Copyright © 2007 by the Institute of Electrical and Electronics Engineers, Inc.  
8 Three Park Avenue  
9 New York, New York 10016-5997, USA  
10 All rights reserved.

11 This document is an unapproved draft of a proposed IEEE Standard. As such, this document is subject to  
12 change. USE AT YOUR OWN RISK! Because this is an unapproved draft, this document must not be  
13 utilized for any conformance/compliance purposes. Permission is hereby granted for IEEE Standards  
14 Committee participants to reproduce this document for purposes of IEEE standardization activities only.  
15 Prior to submitting this document to another standards development organization for standardization  
16 activities, permission must first be obtained from the Manager, Standards Licensing and Contracts, IEEE  
17 Standards Activities Department. Other entities seeking permission to reproduce this document, in whole or  
18 in part, must obtain permission from the Manager, Standards Licensing and Contracts, IEEE Standards  
19 Activities Department.

20 IEEE Standards Activities Department  
21 Standards Licensing and Contracts  
22 445 Hoes Lane, P.O. Box 1331  
23 Piscataway, NJ 08855-1331, USA

24

25 **Abstract:** This standard specifies cryptographic and data authentication procedures for storage devices that  
26 support length-expansion, such as tape drives. Such procedures include the following cryptographic modes  
27 of operation for the AES block cipher: CCM, GCM, CBC-HMAC, and XTS-HMAC.

28 **Keywords:** authentication, data storage, CBC, CCM, cryptography, encryption, GCM, HMAC, security,  
29 tape drive, variable-length block, XTS  
30

- Deleted: ¶
- Deleted: ¶
- Deleted: A
- Deleted: D
- Deleted: S
- Deleted: C
- Deleted: E
- Deleted: Key-transform, S
- Deleted: T
- Deleted: D
- Deleted: V
- Deleted: L
- Deleted: B
- Deleted: and

## 1 Introduction

2 (This introduction is not part of IEEE P1619.1/D20, Draft Standard for Authenticated Encryption with  
3 Length Expansion for Storage Devices.)

4  
5 The problem of data storage protection has become increasingly important due to legislation that requires  
6 the encryption of sensitive information. To address this issue, the Security in Storage Working Group  
7 (SISWG) is developing standards for the protection of information on storage media. This standard  
8 provides strong data protection by specifying encryption with authentication and length expansion.

9  
10 This standard provides methods suitable for ensuring the privacy and integrity of stored data within  
11 applications requiring a high level of assurance. To this end, this standard specifies the Advanced  
12 Encryption Standard (AES) cipher as used in authenticated-encryption modes.

13  
14 There are many modes of non-cryptographic attacks that are outside the scope of this standard. See B.1 for  
15 a discussion.

## 16 Patents

17 Attention is called to the possibility that implementation of this standard may require use of subject matter  
18 covered by patent rights. By publication of this standard, no position is taken with respect to the existence  
19 or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying  
20 patents or patent applications for which a license may be required to implement an IEEE standard or for  
21 conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

## 22 Participants

23 When work on this draft standard began, the Security in Storage Working Group had the following officers:

24  
25 **James P. Hughes, *Chair***

26 **Serge Plotkin, *Vice-chair***

27 **Fabio Maino, *Secretary***

28  
29 At the time this draft standard was completed, the Security in Storage Working Group operated under the  
30 following sponsorship:

31  
32 Sponsor: **John L. Cole, *Information Assurance Standards Committee***

33 Co-Sponsor: **Curtis Anderson, *Storage Systems Standards Committee***

34  
35 At the time this draft standard was completed, the Security in Storage Working Group had the following  
36 officers:

37  
38 **Matthew V. Ball, *Chair***

39 **Eric A. Hibbard, *Vice-chair***

40 **Fabio Maino, *Secretary***

41  
42 At the time this draft standard was completed, the P1619.1 Task Group had the following membership:

43  
44 **Matthew V. Ball, *Chair, Technical Editor***

Deleted: standard

1	Gideon Avida	13	Larry D. Hofer	25	Scott Painter
2	Matthew V. Ball	14	Jon Holdman	26	David Peterson
3	Jon Buckingham	15	Walt Hubis	27	Serge Plotkin
4	William G. Colvin	16	James P. Hughes	28	David Sheehy
5	Russel Dietz	17	Glen Jaquette	29	Robert N. Snively
6	Robert Drennan	18	Curt Kolovson	30	Alexander (Sandy) Stewart
7	Hal Finney	19	Robert A. Lockhart	31	Greg Unruh
8	John Geldman	20	Fabio Maino	32	Douglas L. Whiting
9	Robert Griffin	21	Charlie Martin	33	Christopher L. Williams
10	Shai Halevi	22	Dalit Naor	34	Mike Witkowski
11	Laszlo Hars	23	Landon Curt Noll		
12	Eric A. Hibbard	24	Jim Norton		
35					
36					
37	Previous technical editors:				
38					
39	Shai Halevi				
40	James P. Hughes				
41	Glen Jaquette				
42					
43					
44	Special thanks to:				
45					
46	Brian Gladman				
47	David McGrew				
48	Michael Torla				
49	Danh Tran				

1	CONTENTS	
2	<u>1. Overview .....</u>	<u>1</u>
3	<u>1.1 Scope .....</u>	<u>1</u>
4	<u>1.2 Purpose .....</u>	<u>1</u>
5	<u>1.3 Description of clauses and annexes .....</u>	<u>1</u>
6	<u>2. Normative references.....</u>	<u>1</u>
7	<u>3. Definitions, acronyms, and abbreviations .....</u>	<u>2</u>
8	<u>3.1 Keywords.....</u>	<u>2</u>
9	<u>3.2 Definitions .....</u>	<u>3</u>
10	<u>3.3 Acronyms and abbreviations .....</u>	<u>5</u>
11	<u>3.4 Mathematical conventions .....</u>	<u>6</u>
12	<u>4. General concepts .....</u>	<u>6</u>
13	<u>4.1 Introduction .....</u>	<u>6</u>
14	<u>4.2 Components .....</u>	<u>8</u>
15	<u>4.3 Encryption routine .....</u>	<u>10</u>
16	<u>4.4 Decryption routine.....</u>	<u>11</u>
17	<u>5. Cryptographic modes.....</u>	<u>12</u>
18	<u>5.1 Overview .....</u>	<u>12</u>
19	<u>5.2 Counter mode with CBC-MAC (CCM).....</u>	<u>13</u>
20	<u>5.3 Galois/counter mode (GCM) .....</u>	<u>14</u>
21	<u>5.4 Cipher Block Chaining with HMAC-SHA (CBC-HMAC-SHA) .....</u>	<u>14</u>
22	<u>5.5 Tweakable block-cipher with HMAC (XTS-AES-256-HMAC-SHA-512).....</u>	<u>16</u>
23	<u>6. Cryptographic key management and initialization vector requirements.....</u>	<u>17</u>
24	<u>6.1 Random bit generator .....</u>	<u>17</u>
25	<u>6.2 Cryptographic key entry and export .....</u>	<u>18</u>
26	<u>6.3 Handling the cipher key.....</u>	<u>18</u>
27	<u>6.4 Cryptographic key wrapping on the storage medium .....</u>	<u>18</u>
28	<u>6.5 Initialization Vector (IV) requirements .....</u>	<u>19</u>
29	<u>6.6 Creating unique IVs within a self-contained group .....</u>	<u>20</u>
30	<u>Annex A (informative) Bibliography .....</u>	<u>22</u>
31	<u>Annex B (informative) Security concerns .....</u>	<u>24</u>
32	<u>B.1 Threat model .....</u>	<u>24</u>
33	<u>B.2 Lack of security when using passwords as cryptographic keys .....</u>	<u>24</u>
34	<u>B.3 Replay attacks .....</u>	<u>24</u>
35	<u>B.4 Passing plaintext to the host before checking the MAC .....</u>	<u>25</u>
36	<u>B.5 Checking for integrity of a cryptographic key .....</u>	<u>25</u>
37	<u>B.6 Avoiding collisions of initialization vectors .....</u>	<u>26</u>
38	<u>B.7 Examples of IV collision avoidance strategies .....</u>	<u>26</u>
39	<u>B.8 How many records to encrypt with one key? .....</u>	<u>28</u>

1	<a href="#">Annex C (informative) Documentation Requirements summary .....</a>	<a href="#">30</a>
2	<a href="#">Annex D (informative) Test vectors .....</a>	<a href="#">32</a>
3	<a href="#">D.1 General.....</a>	<a href="#">32</a>
4	<a href="#">D.2 CCM test vectors .....</a>	<a href="#">33</a>
5	<a href="#">D.3 GCM test vectors .....</a>	<a href="#">34</a>
6	<a href="#">D.4 CBC-HMAC-SHA-1 Test Vectors.....</a>	<a href="#">37</a>
7	<a href="#">D.5 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vectors.....</a>	<a href="#">39</a>
8	<a href="#">D.6 XTS-AES-256-HMAC-512 Test Vectors.....</a>	<a href="#">42</a>

**Deleted:**

- 1. Overview . 1¶
- 1.1 Scope . 1¶
- 1.2 Purpose . 1¶
- 1.3 Description of clauses and annexes . 1¶
- 2. Normative references . 1¶
- 3. Definitions, acronyms, and abbreviations . 2¶
- 3.1 Keywords . 2¶
- 3.2 Definitions . 3¶
- 3.3 Acronyms and abbreviations . 5¶
- 3.4 Mathematical conventions . 6¶
- 4. General concepts . 6¶
- 4.1 Introduction . 6¶
- 4.2 Overview of roles . 7¶
- 4.3 Host . 9¶
- 4.4 Key manager . 12¶
- 4.5 Cryptographic unit . 13¶
- 4.6 Storage medium . 17¶
- 4.7 Documentation **Error! Bookmark not defined.**¶
- 5. Cryptographic modes . 17¶
- 5.1 Cryptographic mode selection . 17¶
- 5.2 Encryption . 18¶
- 5.3 Decryption . 19¶
- 5.4 Parameter limits . 20¶
- 5.5 Counter mode with CBC-MAC (CCM) . 21¶
- 5.6 Galois/counter mode (GCM) . 21¶
- 5.7 Cipher Block Chaining with HMAC-SHA (CBC-HMAC-SHA) . 22¶
- 5.8 Tweakable block-cipher with HMAC (XTS-AES-256-HMAC-SHA-512) . 23¶
- 6. Cryptographic key management and initialization vector requirements . 24¶
- 6.1 Random bit generator . 24¶
- 6.2 Cryptographic key entry and export . 25¶
- 6.3 Handling the user key and cipher key . 25¶
- 6.4 Cryptographic key wrapping on the storage medium . 26¶
- 6.5 Initialization Vector (IV) requirements . 27¶
- 6.6 Creating unique IVs within a self-contained group . 29¶
- Annex A (informative) Bibliography . 30¶
- Annex B (informative) Security concerns . 32¶
- B.1 Threat model . 32¶
- B.2 Lack of security when using passwords as cryptographic keys . 32¶
- B.3 Replay attacks . 32¶
- B.4 Passing plaintext to the host before checking the MAC . 33¶
- B.5 Checking for integrity of a cryptographic key . 33¶
- B.6 Avoiding collisions of initialization vectors . 34¶
- B.7 Examples of IV collision avoidance strategies . 34¶
- B.8 How many records to encrypt with one key? . 36¶
- Annex C (informative) Documentation Requirements summary . 38¶
- Annex D (informative) Test vectors . 32¶
- D.1 General . 32¶
- D.2 CCM test vectors . 33¶
- D.3 GCM test vectors . 34¶
- D.4 CBC-HMAC-SHA-1 Test Vectors . 37¶

# 1 Draft Standard for Authenticated Encryption with 2 Length Expansion for Storage Devices

## 3 1. Overview

### 4 1.1 Scope

5 This standard specifies requirements for cryptographic units that provide encryption and authentication for  
6 data contained within storage media. Full interchange requires additional format specifications (such as  
7 compression algorithms and physical data format) that are beyond the scope of this standard.

### 8 1.2 Purpose

9 This standard is suitable for encryption of data stored on tape because tape easily accommodates length-  
10 expanding ciphertext. In addition, this standard applies to other storage devices if these support storing  
11 extra metadata with each encrypted record. The algorithms of this standard are designed to ensure the  
12 confidentiality and integrity of stored data within systems requiring a high level of assurance.

### 13 1.3 Description of clauses and annexes

14 — Clause 1 provides an overview of this standard, including scope and purpose.

← Formatted: Bullets and Numbering

15 — Clause 2 lists the normative references that are essential for implementing this standard.

16 — Clause 3 gives definitions, acronyms, and abbreviations used in this standard.

17 — Clause 4 provides a description of the components that play roles in this standard.

18 — Clause 5 describes the cryptographic modes used by the cryptographic unit.

Deleted: 1

19 — Clause 6 describes cryptographic key management and initialization vector requirements.

20 — Annex A (informative) lists bibliographic references that are useful when implementing this standard.

21 — Annex B (informative) discusses several security issues that an implementer or user should understand.

Deleted: <#>Error! Reference source not found. provides several test vectors useful in verifying a cryptographic unit.¶

22 — Annex C (informative) provides a summary of documentation requirements.

23 — Annex D (informative) provides several test vectors useful in verifying a cryptographic unit.

Deleted: ¶  
Annex D describes an optional format for interchange of encrypted data.

## 24 2. Normative references

25 The following referenced documents are indispensable for the application of this document. For dated  
26 references, only the edition cited applies. For undated references, the latest edition of the referenced  
27 document (including any amendments or corrigenda) applies.

- 1  
2  
3 | ~~McGrew, David and John Viega, The Galois/Counter Mode of Operation (GCM), May 31, 2005<sup>2</sup>.~~
- 4 NIST FIPS 180-2, Federal Information Processing Standard (FIPS) 180-2 (August 1, 2002), Announcing  
5 the Secure Hash Standard (SHA).
- 6 NIST FIPS 197, Federal Information Processing Standard (FIPS) 197 (November 26, 2001), Announcing  
7 the Advanced Encryption Standard (AES).  
8
- 9 NIST FIPS 198a, Federal Information Processing Standard (FIPS) 198 (March 2002 updated April 8,  
10 2002), The Keyed-Hash Message Authentication Code (HMAC).  
11
- 12 NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation - Methods and  
13 Techniques.  
14
- 15 NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM  
16 Mode for Authentication and Confidentiality.<sup>3</sup>  
17
- 18 NIST Draft Special Publication 800-38D (April 20, 2006), Recommendation for Block Cipher Modes of  
19 Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication.  
20
- 21 | IEEE P1619<sup>TM</sup> (Draft ~~16, May~~ 2007), Draft Standard for Cryptographic Protection of Data on Block-  
22 Oriented Storage Devices.

Deleted: ISO/IEC 14165-252, Fibre Channel Framing and Signaling-2 (FC-FS-2), ANSI INCITS 424-2006<sup>4</sup>.

Deleted: 14

Deleted: March

### 23 3. Definitions, acronyms, and abbreviations

#### 24 3.1 Keywords

25 For the purposes of this standard, the following terms are keywords.

26  
27 **can:** A keyword indicating a capability (*can* equals *is able to*).

28  
29 **recommended:** *See should*.

30  
31 **required:** *See shall*.

32  
33 **may:** A keyword indicating a course of action permissible within the limits of this standard (*may* equals *is*  
34 *permitted to*).

35  
36 **must:** A keyword used only to describe an unavoidable situation that does not constitute a requirement for  
37 compliance to this standard.

38  
39 **shall:** A keyword indicating a mandatory requirement strictly to be followed in order to conform to this  
40 standard and from which no deviation is permitted. (*shall* equals *is required to*).

41  
42 **shall not:** a phrase indicating an absolute prohibition of this standard.

43  
44 **should:** A keyword indicating that among several possibilities one is recommended as particularly suitable,  
45 without mentioning or excluding others; or that a certain course of action is preferred but not necessarily

<sup>2</sup> Available from the World Wide Web site <<http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-revised-spec.pdf>>

<sup>3</sup> Available from the NIST World Wide Web site <<http://csrc.nist.gov/publications/nistpubs/index.html>>

1 required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should*  
 2 equals *is recommended to*).

3  
 4 **will:** A keyword providing a statement of fact that does not constitute a requirement for conformance to this  
 5 standard.

6 **3.2 Definitions**

7 For the purposes of this draft standard, the following terms and definitions apply. *The Authoritative*  
 8 *Dictionary of IEEE Standards, Seventh Edition*, should be referenced for terms not defined in this clause.  
 9

10 **3.2.1 additionally authenticated data (AAD):** Information passed into an authenticated encryption routine  
 11 that is authenticated but not encrypted.

12 **3.2.2 advanced encryption standard (AES):** The block cipher defined by NIST FIPS 197. *See also:*  
 13 block cipher.

14 **3.2.3 block cipher:** A cryptographic primitive that uses a cipher key to create a pseudo-random  
 15 permutation of a fixed-size bit string. *See also:* **cipher key; plaintext; ciphertext.**

Deleted: 3.2  
 Formatted: IEEEStdts  
 DefTerms+Numbers, Do not check  
 spelling or grammar

16 **3.2.4 CBC-HMAC-SHA:** A family of cipher blocking chaining (CBC) modes (see NIST SP 800-38A)  
 17 with a keyed-hash message authentication code (see NIST FIPS 198a) from a member of the secure hash  
 18 algorithm family (see NIST FIPS 180-2).

19 NOTE—See 5.4.

20 **3.2.5 CBC-IV:** The IV input for the CBC modes, according to NIST Special Publication 800-38A. *See*  
 21 *also:* **CBC-HMAC-SHA; initialization vector.**

22 NOTE—See 5.4.

23 **3.2.6 cipher block chaining (CBC):** A cryptographic mode of operation in which the ciphertext output  
 24 from each cipher block feeds into the following cipher block (see NIST SP 800-38A).

25 **3.2.7 cipher key:** A bit string that controls the pseudo-random permutation of an encryption or decryption  
 26 routine. *See also:* **cryptographic key; block cipher.**

Deleted: 3.2  
 Formatted: IEEEStdts  
 DefTerms+Numbers, Do not check  
 spelling or grammar  
 Deleted: user key;

27 **3.2.8 collision:** An event where two independent variables have the same value in a particular context. *See*  
 28 *also:* **initialization vector; plaintext; ciphertext.**

29 **3.2.9 counter mode (CTR):** A cryptographic mode of operation defined by NIST SP 800-38A in which the  
 30 ciphertext is the bitwise exclusive-or of the plaintext with an encrypted counter.

31 **3.2.10 counter mode encryption with cipher block chaining message authentication code (CCM):** A  
 32 cryptographic mode of operation that provides confidentiality with counter mode and integrity with a  
 33 message authentication code that uses cipher block chaining (see NIST SP 800-38C).

34 NOTE—See 5.2

35 **3.2.11 cryptographically-sound random-bit generator (RBG):** A device or algorithm that outputs a  
 36 sequence of binary bits that appears to be statistically independent and unbiased. In particular, a RBG  
 37 generates numbers that are highly unpredictable, and knowledge of any particular output from a RBG does  
 38 not reveal any information about other data generated by the RBG.

Formatted: IEEEStdts  
 DefTerms+Numbers, Do not check  
 spelling or grammar  
 Deleted: 3.2

39 NOTE—See 6.1

1 **3.2.12 cryptographic hash function:** A hash function that generates a hash value from an input and that  
 2 has the following properties: 1) it is computationally difficult to compute the inverse (i.e. compute the input  
 3 from the hash value); 2) it is computationally difficult to find two different inputs that have the same hash  
 4 value; 3) it is computationally difficult to find an input whose hash value is a particular value.

5 **3.2.13 cryptographic key:** A bit string used as an input into cryptographic primitives. *See also:* **block**  
 6 **cipher; cipher key.**

Formatted: IEEEStd  
DefTerms+Numbers, Do not check  
spelling or grammar

Deleted: 3.2

Deleted: ; user key

7 **3.2.14 cryptographic mode:** *See* **cryptographic mode of operation.**

8 **3.2.15 cryptographic mode of operation:** An algorithm that includes a block cipher used in a particular  
 9 configuration and uses a cipher key to convert plaintext into ciphertext and vice versa. *See also:* **block**  
 10 **cipher; cipher key; plaintext; ciphertext.**

11 **3.2.16 cryptographic unit:** Any set of software, firmware, or hardware that can perform a cryptographic  
 12 operation.

13 **3.2.17 decryption:** The act of producing plaintext from ciphertext. *Contrast:* **encryption.** *See also:*  
 14 **plaintext; ciphertext; cryptographic key; cryptographic mode of operation.**

15 **3.2.18 decryption routine:** An instantiation of a cryptographic mode of operation that converts ciphertext  
 16 into plaintext.

17 **3.2.19 encrypted record:** A collection of fields that includes the output of an encryption operation or  
 18 authentication operation (e.g., ciphertext, message authentication code), and optionally contains other  
 19 information needed for a subsequent decryption operation (e.g., additionally authenticated data,  
 20 initialization vector). *See also:* **ciphertext; initialization vector; message authentication code;**  
 21 **additionally authenticated data.**

Formatted: IEEEStd  
DefTerms+Numbers, Do not check  
spelling or grammar

Deleted: 3.2

22 **3.2.20 encryption:** The act of producing ciphertext from plaintext. *Contrast:* **decryption.** *See also:*  
 23 **plaintext; ciphertext; cryptographic key; cryptographic mode of operation.**

24 **3.2.21 encryption routine:** An instantiation of a cryptographic mode of operation that converts plaintext  
 25 into ciphertext.

26 **3.2.22 encryption session:** An interval in which a cryptographic unit generates a set of self-consistent  
 27 variables, such as unique initialization vectors, for encryption operations. *See also:* **encryption;**  
 28 **initialization vector; cryptographic unit.**

Formatted: IEEEStd  
DefTerms+Numbers, Do not check  
spelling or grammar

Deleted: 3.2

29 NOTE—See 6.5.3.

30 **3.2.23 Galois/counter mode (GCM):** A cryptographic mode of operation that provides confidentiality  
 31 through counter mode encryption and integrity through a message authentication code that uses Galois field  
 32 arithmetic (See NIST SP 800-38D).

33 NOTE—See 5.3.

34 **3.2.24 host record:** a string of plaintext passed to the cryptographic unit from the host. *See also:* **plaintext**  
 35 **record; cryptographic unit; host.**

Formatted: IEEEStd  
DefTerms+Numbers, Do not check  
spelling or grammar

Deleted: 3.2

36 **3.2.25 initialization vector (IV):** An input into an encryption or decryption algorithm that needs not be  
 37 secret, but has a high probability of being unique when used with a particular cipher key. *See also:*  
 38 **encryption; decryption; encryption session; cipher key.**

39 NOTE—See 6.5.

1 **3.2.26 key encrypting key (KEK):** A cryptographic key used to encrypt another cryptographic key. *See*  
 2 *also: cryptographic key; encryption.*

3 **3.2.27 keyed-hash message authentication code (HMAC):** A message authentication code defined by  
 4 NIST FIPS 198a that includes a secret hash key.

5 **3.2.28 key manager:** Any device or person that controls the creation, archiving, and destruction of  
 6 cryptographic keys. *See also: cryptographic key.*

Deleted: 3.2  
 Formatted: IEEEStd  
 DefTerms+Numbers, Do not check  
 spelling or grammar

7 NOTE—See 4.2.3

8 **3.2.29 message authentication code (MAC):** A cryptographic checksum that is used to detect intentional  
 9 modifications and errors in an encrypted record, and cannot be efficiently forged without knowledge of the  
 10 cryptographic key used in the MAC algorithm. *See also: encrypted record.*

11 **3.2.30 nonce:** A bit string that has a low probability of matching any other nonce in a particular context.  
 12 *See also: initialization vector.*

Deleted: 3.2.<#>name address  
 authority (NAA): A format for a  
 globally unique number that includes an  
 IEEE OUI as specified by ISO/IEC  
 14165-252.¶

13 **3.2.31 plaintext:** Information that has not been obscured through a cryptographic transformation.  
 14 *Contrast: ciphertext.*

Formatted: IEEEStd  
 DefTerms+Numbers, Do not check  
 spelling or grammar

15 **3.2.32 plaintext record:** A string of plaintext passed to an encryption routine to produce an encrypted  
 16 record. *See also: plaintext; cryptographic unit; encryption; encrypted record; host record.*

Deleted: 3.2  
 Formatted: Bullets and Numbering

17 **3.2.33 random bit generator (RBG):** *See cryptographically-sound random-bit generator.*

Deleted: 3.2.<#>organizationally  
 unique identifier (OUI): A 3-byte value  
 that uniquely identifies a particular entity  
 by means of a registry maintained by the  
 IEEE.¶

18 **3.2.34 random nonce:** A nonce that completely consists of the output from a random bit generator. *See*  
 19 *also: nonce; cryptographically-sound random-bit generator.*

Formatted: IEEEStd  
 DefTerms+Numbers, Do not check  
 spelling or grammar

20 **3.2.35 secure hashing algorithm (SHA):** A family of cryptographic hash functions defined by NIST FIPS  
 21 180-2. *See also cryptographic hash function.*

Deleted: 3.2  
 Formatted: Bullets and Numbering

22 **3.2.36 secure hashing standard (SHS):** *See NIST FIPS 180-2.*

Formatted: Bullets and Numbering  
 Formatted: Bullets and Numbering

23 **3.2.37 self-contained group:** A set of cryptographic units that generate and use IVs in a consistent manner.

Formatted: Bullets and Numbering  
 Deleted:

24 NOTE—See 6.6.

Deleted:  
 Formatted: Bullets and Numbering

26 **3.3 Acronyms and abbreviations**

27 AAD additional authenticated data  
 28 AES advanced encryption standard  
 29 CBC cipher block chaining  
 30 CCM counter mode encryption with cipher block chaining message authentication code  
 31 CTR counter mode  
 32 FIPS Federal Information Processing Standards  
 33 FIPS PUB Federal Information Processing Standards Publication  
 34 GCM Galois/counter mode  
 35 HMAC keyed-hash message authentication code  
 36 IEEE Institute of Electrical and Electronics Engineers  
 37 IV initialization vector  
 38 KEK key encrypting key  
 39 MAC message authentication code

Formatted: Bullets and Numbering  
 Formatted: IEEEStd  
 DefTerms+Numbers, Do not check  
 spelling or grammar

Deleted: 3.2

Formatted: Bullets and Numbering

Formatted: IEEEStd  
 DefTerms+Numbers, Do not check  
 spelling or grammar

Deleted: 3.2

Formatted: Bullets and Numbering

Deleted: 3.2.<#>user key: A  
 cryptographic key supplied by the key  
 manager. *See also: cipher key; key  
 manager.*

1	NIST	National Institute of Standards and Technology
2	RBG	(cryptographically-sound) random-bit generator
3	SHA	secure hashing algorithm
4	SHS	secure hashing standard
5	SP	(NIST) special publication
6	XOR	exclusive OR

Deleted: NAA name address authority¶

Deleted: OUI . organizationally unique identifier¶

Deleted:

Deleted:

7 **3.4 Mathematical conventions**

8 **3.4.1 General**

9 The following subclauses contain definitions for mathematical concepts as used in this standard.

10 **3.4.2 Numerical representation**

11 This standard uses decimal, binary, and hexadecimal numbers. For clarity, decimal numbers generally  
 12 represent counts, and binary or hexadecimal numbers describe bit patterns or raw binary data.

13  
 14 Binary numbers are represented by a string of one or more binary digits, followed by the subscript 2.  
 15 Unless otherwise stated, hexadecimal numbers are a string of the characters 0-9 and 'a'-'f', followed by the  
 16 subscript 16. Thus, the decimal number 26 is represented as 00011010<sub>2</sub> in binary and 1a<sub>16</sub> in hexadecimal.  
 17 Decimal numbers do not have a suffix or subscript.

18 **3.4.3 Concatenation**

19 The concatenation operator (||), represented as two vertical pipes, joins two bit strings such that the left  
 20 operand occupies the lower addresses of the result, and the right operand occupies the upper addresses. If  
 21 the result is interpreted as an integer, the left operand contains the most-significant bits and the right  
 22 operand contains the least-significant bits.

23  
 24 NOTE—This is consistent with the big-endian convention.

25  
 26 *Example:*

27  
 28 0011<sub>2</sub> || 1010<sub>2</sub> = 00111010<sub>2</sub>

29 **4. General concepts**

30 **4.1 Introduction**

31 This standard describes elements of an architecture that is suitable for the cryptographic confidentiality and  
 32 integrity of stored data. This architecture includes a model of several components within a typical system  
 33 that securely archives and restores information. These components are as follows:

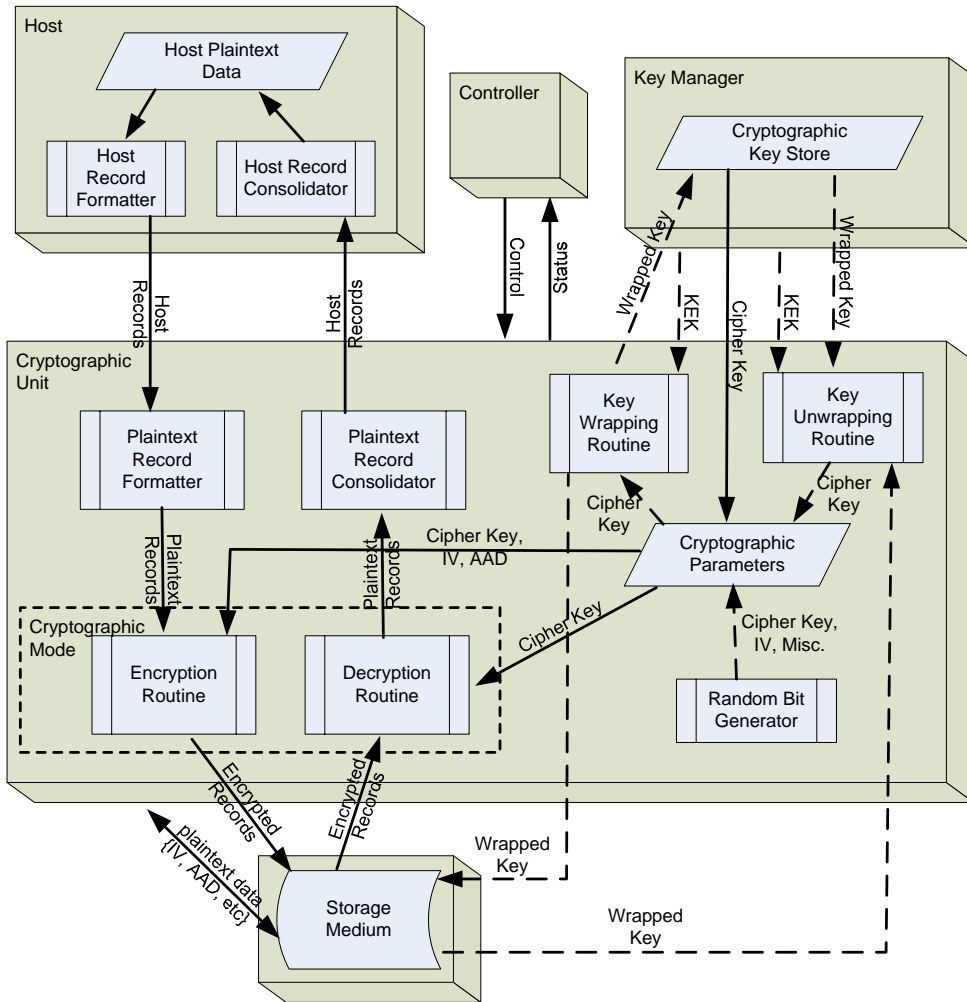
- 34  
 35 — A cryptographic unit that performs data formatting, encryption, decryption, and simple cryptographic  
 36 key management (see 4.2.4).
- 37  
 38 — A controller that controls the overall operation of the cryptographic unit and processes status from the  
cryptographic unit (see 4.2.1).
- 39  
 40 — A host that provides plaintext data, in the form of host records, to the cryptographic unit and receives  
plaintext data from the cryptographic unit (see 4.2.2).
- 41  
 42 — A key manager that provides cipher keys and key encryption keys (KEK) to the cryptographic unit, and  
that securely maintains the lifecycle of these cryptographic keys (see 4.2.3).

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

1 — A storage medium that provides non-volatile storage of information produced by the cryptographic  
 2 unit (see 4.2.5).  
 3  
 4 This standard provides requirements only for the cryptographic unit.  
 5  
 6 An implementer of this standard shall provide documentation to the end-user with the cryptographic unit.  
 7 This documentation may be in any form (e.g., electronic, printed on paper) that is easily accessible by the  
 8 end-user. Documentation shall include all the required text as specified throughout this standard. This  
 9 information facilitates interchange and allows the user to evaluate the security of the cryptographic unit.  
 10 See Annex C for a summary of requirements.

11  
 12 Figure 1 shows a high-level depiction of the interactions among these four components and of  
 13 subcomponents contained within each component.  
 14



15  
 16

Figure 1—Model showing interactions of components

1 Some of these components may exist within one physical or logical device, depending on the topology of  
2 the complete solution. There may also be multiple instantiations of each component.

3  
4 Subclause 4.2 describes in detail each of the components within Figure 1.

## 5 4.2 Components

### 6 4.2.1 Controller

7 The controller is any person or device that controls the overall operation of the cryptographic unit. This  
8 includes sending commands to the cryptographic unit and processing status from the cryptographic unit.  
9 There may be multiple controllers controlling a particular cryptographic unit. A controller may be part of  
10 another component such as a host or key manager.

### 11 4.2.2 Host

12 The host provides host records to the cryptographic unit for encryption, and receives host records from the  
13 cryptographic unit after decryption. A host record contains plaintext data and may be any size that the  
14 cryptographic unit allows.

15  
16 A typical host includes routines to convert arbitrary host plaintext data into host records and vice versa.  
17 Such host records may be variable-length, depending on the capabilities of the cryptographic unit. In  
18 Figure 1 these routines are as follows:

19  
20 — **Host Record Formatter** – A routine that converts arbitrary host plaintext data into host records for the  
21 cryptographic unit

22 — **Host Record Consolidator** – A routine that consolidates host records from the cryptographic unit into  
23 host plaintext data

24  
25 It is not required for a host to implement these functions. The host needs only to present host records to the  
26 cryptographic unit, and accept host records from a cryptographic unit.

#### 27 Examples:

28  
29 — If the cryptographic unit is a tape drive, then the host might be a computer running a backup  
30 application in which the backup application takes arbitrary host plaintext data in the form of files and  
31 consolidates them into backup sets, breaks these backup sets into variable-length blocks, and sends the  
32 blocks as host records to the cryptographic unit.

33 — If the cryptographic unit is a disk drive, then the host might be an operating system that formats files  
34 into fixed-size sectors (typically 512 bytes) and uses these sectors as host records when sending data to  
35 and receiving data from the cryptographic unit.

### 36 4.2.3 Key manager

37 The key manager is responsible for the life-cycle (e.g., generation, archiving, and destruction) of the  
38 cryptographic keys used by the cryptographic unit. Such a cryptographic key may be a cipher key or a key  
39 encryption key (KEK) (see 6.3). For purposes of this standard, the key manager is modeled as maintaining  
40 these cryptographic keys within a key store.

### 41 4.2.4 Cryptographic unit

#### 42 4.2.4.1 Overview

43 A cryptographic unit is any combination of software, firmware, or hardware that is capable of handling  
44 plaintext and ciphertext using at least one of the cryptographic modes specified in 1.1.  
45 The cryptographic unit shall contain the following subcomponents:

**Deleted:** ¶  
**<#>Overview of roles¶**  
To implement this standard, it is necessary to understand the following roles and their interactions:¶  
¶  
**<#>Host** (see 4.3)¶  
**<#>Key manager** (see 4.4)¶  
**<#>Cryptographic unit** (see 4.5)¶  
**<#>Storage medium** (see 4.6)¶  
¶

**Deleted:** roles

**Deleted:** component

**Formatted:** Bullets and Numbering

**Formatted:** Bullets and Numbering

**Deleted:** The host is any device that controls the overall operation of the cryptographic unit.

**Deleted:** ¶

**Formatted:** Bullets and Numbering

**Formatted:** Bullets and Numbering

**Deleted:** Multiple hosts may control a cryptographic unit.¶

**Deleted:** user key,

**Formatted:** Bullets and Numbering

**Formatted:** Bullets and Numbering

**Deleted:** Table 1

**Deleted:** by this standard (see Clause 1)

**Deleted:** .¶

- 1 — [Plaintext record formatter \(see 4.2.4.2\)](#)
- 2 — [Plaintext record consolidator \(see 4.2.4.3\)](#)
- 3 — [Encryption routine \(see 4.3\)](#)
- 4 — [Decryption routine \(see 4.4\)](#)
- 5 — [Cryptographic parameters \(see 4.2.4.4\)](#)

← - - - Formatted: Bullets and Numbering

- 6
- 7 [The cryptographic unit may contain the following subcomponents:](#)
- 8 — [Random bit generator \(see 6.1\)](#)
  - 9 — [Key wrapping routine \(see 6.4\)](#)
  - 10 — [Key unwrapping routine \(see 6.4\)](#)

← - - - Formatted: Bullets and Numbering

11 [The cryptographic unit may contain other subcomponents that do not affect the cryptographic functionality](#)

12 [of the cryptographic unit:](#)

← - - - Formatted: Bullets and Numbering

13 **4.2.4.2 Plaintext record formatter**

14 [The plaintext record formatter is a routine that converts host records into plaintext records that pass into the](#)

15 [encryption routine.](#)

16

17 [The cryptographic unit receives host records from the host as a basic unit of data for encryption. When](#)

18 [performing encryption, the cryptographic unit shall use the plaintext record formatter to format the host](#)

19 [records into plaintext records.](#)

20

21 [To minimize buffering requirements and latency, the cryptographic unit may define a maximum size for the](#)

22 [plaintext records that is smaller than the maximum host record size allowed by the cryptographic unit. The](#)

23 [plaintext record formatter may split the host record into multiple plaintext records with optional padding or](#)

24 [reformatting.](#)

25

26 [The cryptographic unit may apply padding or perform reversible transforms \(such as compression\) to the](#)

27 [data within the host records to form the plaintext records.](#)

28

29 [The cryptographic unit shall include sufficient information within the AAD, IV, or plaintext record to allow](#)

30 [the plaintext record consolidator \(see 4.2.4.3\) to unambiguously reconstruct the original set of host records,](#)

31 [even after malicious tampering. To help fulfill this requirement, the cryptographic unit should use ordering](#)

32 [verification to detect tampering or reordering of the encrypted records \(see 4.4.3\).](#)

33

34 [Documentation shall describe how the plaintext record formatter generates plaintext records from host](#)

35 [records.](#)

← - - - Formatted: Bullets and Numbering

36 **4.2.4.3 Plaintext record consolidator**

37 [The plaintext record consolidator is a routine that converts plaintext records received from the decryption](#)

38 [routine into host records that the cryptographic unit passes to the host.](#)

39

40 [The plaintext record consolidator shall only use information that the decryption routine cryptographically](#)

41 [verified using a message authentication code \(MAC\).](#)

42

43 [If the plaintext record contains padding or reversible transforms, then the plaintext record consolidator shall](#)

44 [verify the correctness of these formats. If the format is incorrect, then the cryptographic unit shall send the](#)

45 [special signal FAIL to the controller and should not return any host records.](#)

46

47 [Documentation shall describe how the plaintext record consolidator generates host records from plaintext](#)

48 [records.](#)

1 **4.2.4.4 Cryptographic parameters**  
 2 A cryptographic parameter is a value that affects the cryptographic confidentiality or integrity of encrypted  
 3 information. The cryptographic shall securely maintain all cryptographic parameters from unauthorized  
 4 disclosure and/or modification.

6 The cryptographic module shall protect the following cryptographic parameters from unauthorized  
 7 modification:  
 8 — additionally authenticated data (AAD)  
 9 — initialization vector (IV)  
 10 — any asymmetric public key (e.g., asymmetric public KEK)

12 Additionally, the cryptographic module shall protect the following cryptographic parameters from both  
 13 unauthorized modification and disclosure:  
 14 — cipher keys  
 15 — seed keys for random bit generators  
 16 — any asymmetric private key (e.g., asymmetric public KEK)  
 17 — any symmetric KEK

18 Documentation shall describe all cryptographic parameters used by the cryptographic unit.

19 **4.2.5 Storage medium**

20 The storage medium is any device or material that is able to store non-volatile data.

22 The controller may configure the cryptographic unit to write a particular plaintext record to the storage  
 23 medium either with encryption or without encryption. In this case, the cryptographic unit may mix both  
 24 encrypted records and plaintext records on the storage medium. The cryptographic unit may write  
 25 additional information without encryption to the storage medium, assuming that such information does not  
 26 reveal cryptographic keys or plaintext that was intended to be encrypted. The cryptographic unit shall not  
 27 write information to the storage medium that compromises the cryptographic confidentiality or integrity of  
 28 any encrypted information on the storage medium.

30 **4.3 Encryption routine**

31 **4.3.1 Overview**

32 The encryption routine takes formatted plaintext records as input and produces encrypted records as output.  
 33 The combination of an encryption routine and decryption routine forms a cryptographic mode of operation  
 34 (see Clause 5). Subclause 4.3 describes the traits of an encryption routine that are common across all  
 35 cryptographic modes.

36 **4.3.2 Inputs**

37 The encryption routine requires the following inputs (See 5.1 for limits):

- 38 a) A secret cipher key.
- 39 b) An initialization vector (IV).
- 40 c) Length of the IV.
- 41 d) Plaintext record.
- 42 e) Length of the plaintext record.
- 43 f) Additional authenticated data (AAD).
- 44 g) Length of the AAD.
- 45

Formatted: Bullets and Numbering

Formatted: IEEEStd's Unordered List

Formatted: IEEEStd's Unordered List

Deleted: *Examples of cryptographic unit embodiments:*  
 ¶  
 <#>A tape drive¶  
 <#>A disk drive that emulates a tape drive or supports length-expansion¶  
 <#>A software application running on a host computer¶  
 <#>A bridge device that sits between the host and the drive¶

Formatted: Bullets and Numbering

Deleted: ¶  
*Examples of storage media:*  
 ¶  
 <#>Magnetic tape cartridge¶  
 <#>Hard disk¶  
 <#>Flash memory¶  
 <#>Holographic or optical storage¶  
 <#>Cryptographic modes¶  
 <#>Cryptographic mode selection¶  
 The cryptographic unit shall support at least one of the cryptographic modes shown in Table 1:¶  
 ¶  
 <#>—Cryptographic modes¶ (... [2])

Formatted: Bullets and Numbering

Deleted: ¶  
 <#>The cryptographic unit receives host records from the host as a basic unit of data for encryption. When performing encryption, the cryptographic unit shall format the host records into plaintext records that are input into the encryption routine. Documentation shall describe the mapping between host records and plaintext records.¶  
 <#>¶  
 <#>To minimize buffering requirements and latency, the cryptographic unit may define a maximum size for the plaintext records that is smaller than the maximum host record size allowed by the cryptographic unit. If a particular host record is larger than the maximum plaintext record length, then the cryptographic unit may split the host record into multiple plaintext records with optional padding or reformatting. When splitting a host record into multiple plaintext records, the cryptographic unit shall include sufficient information within the AAD, IV, or plaintext record to allow the decryption routine to unambiguously reconstruct the original host record. To help fulfill this requirement, the cryptographic unit should use ordering (... [3])

Formatted: Bullets and Numbering

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49

### 4.3.3 Outputs

The encryption routine produces two outputs that are included within an encrypted record:

- a) A ciphertext record that has the same length as the plaintext record.
- b) A message authentication code (MAC) with length specified by the cryptographic mode.

An encrypted record may additionally contain the IV and AAD. If an encrypted record does not contain both the IV and AAD, then there shall be sufficient information on the storage medium or in the cryptographic unit to allow reconstruction of the complete IV and AAD. The AAD and IV may contain any other information that does not compromise the cryptographic confidentiality and integrity of the encrypted record (e.g., information to support order verification as given in 4.4.3).

The cryptographic unit shall write the encrypted record to the storage medium.

When performing encryption, the cryptographic unit shall not write the plaintext or cipher key to the storage medium unencrypted. The cryptographic unit may write a cryptographically wrapped version of the cipher key to the storage medium (see 6.4).

## 4.4 Decryption routine

### 4.4.1 Overview

The decryption routine uses a cipher key to convert encrypted records from the storage medium into plaintext records for the plaintext record consolidator.

The following subclauses describe the requirements for decryption that are common among all the cryptographic modes specified in this standard.

### 4.4.2 Decryption inputs

The decryption routine requires the following inputs:

- a) A secret cipher key.
- b) Initialization vector (IV).
- c) Length of the IV.
- d) Ciphertext.
- e) Length of the ciphertext.
- f) Additional authenticated data (AAD).
- g) Length of the AAD.
- h) A message authentication code (MAC) with length determined by the cryptographic mode.

During decryption, the cryptographic unit shall always validate the MAC. The cryptographic unit should validate the MAC before sending any plaintext to the host. Documentation shall disclose whether the cryptographic unit validates the MAC before returning any plaintext.

If the cryptographic unit validates the MAC before returning plaintext, then it shall not return plaintext to the host if the MAC validation fails. Instead, the cryptographic unit shall return the special signal *FAIL*.

If the cryptographic unit returns plaintext to the host before validating the MAC, then the cryptographic unit shall subsequently validate the MAC. If this MAC validation fails, then the cryptographic unit shall return the special signal *FAIL*. If this MAC validation passes, then the cryptographic unit shall return the special signal *PASS*.

Formatted: Bullets and Numbering

Deleted: C

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

1 The host should not act on any plaintext from the cryptographic unit until receiving a complete host record  
 2 and the special signal *PASS*.

3  
 4 Documentation shall define the special signal *FAIL* and describe how the host receives such a signal. The  
 5 special signal *FAIL* should identify the records that failed the MAC validation.

6  
 7 Documentation shall define the special signal *PASS* and describe how the host receives such a signal.

8 NOTE—See B.4 for a discussion on the security concerns of returning plaintext before validating the MAC.

Formatted: Bullets and Numbering

9 **4.4.3 Verification of correct ordering**

10 During decryption, a cryptographic unit should check that each IV or AAD is consistent with the preceding  
 11 IV or AAD based on the documented mechanism used for creating the IV or AAD (see B.3).

12  
 13 If a cryptographic unit is performing ordering-verification and detects an inconsistent IV or AAD, then the  
 14 cryptographic unit shall return the special signal *FAIL* and should not return any plaintext from the  
 15 inconsistent encrypted records.

Deleted: tampered with

16 NOTE—It is not best practice to return plaintext that might be maliciously altered.

17  
 18 If a cryptographic unit supports ordering-verification, then documentation shall specify the methods for  
 19 enabling or disabling this functionality, and shall specify how the cryptographic unit notifies the host of  
 20 inconsistent IV or AAD ordering.

Formatted: Bullets and Numbering

21 **4.4.4 Verification-only mode**

22 The cryptographic unit may support a verification-only mode, where it only validates the MAC and returns  
 23 a *PASS* or *FAIL* signal to the controller, but does not return any host records. The cryptographic key used  
 24 to validate the MAC should have the same level of protection as the cipher key.

Deleted: host

Formatted: Bullets and Numbering

25 **5. Cryptographic modes**

26 **5.1 Overview**

Deleted: Parameter limits

Formatted: Bullets and Numbering

27 This Clause describes the cryptographic modes of operation (i.e. cryptographic modes) allowed by this  
 28 standard when the cryptographic unit is operating in a compliant mode. The cryptographic unit shall  
 29 support at least one of the cryptographic modes shown in Table 1.

Formatted: Bullets and Numbering

30 **Table 1—Cryptographic modes**

Family	Name	Description	Ref.
CCM	CCM-128-AES-256	Counter with 128-bit cipher block chaining MAC	5.2
GCM	GCM-128-AES-256	Galois/Counter Mode with 128-bit MAC	5.3
CBC	CBC-AES-256-HMAC-SHA-1	Cipher block chaining with 160-bit HMAC	5.4
	CBC-AES-256-HMAC-SHA-256	Cipher block chaining with 256-bit HMAC	5.4
	CBC-AES-256-HMAC-SHA-512	Cipher block chaining with 512-bit HMAC	5.4
XTS	XTS-AES-256-HMAC-SHA-512	Tweakable block cipher with 512-bit HMAC	5.5

31 The cryptographic unit shall operate these cryptographic modes within the parameter limits given in Table  
 32 2:  
 33

Table 2—Parameter limits for encryption modes

Cryptographic Mode	Parameter limits, in bytes					
	Cipher key	IV	AAD	Plaintext record	Maximum total plaintext	MAC
CCM-128-AES-256	32	12	0 to $2^{64}-1$	0 to $2^{24}-1$	$2^{64}-1$	16
GCM-128-AES-256	32	12 or 16 to $2^{61}-1$	0 to $2^{61}-1$	0 to $2^{36}-32$	$2^{68}-16$	16
CBC-AES-256-HMAC-SHA-1	52*	16	0 to $2^{64}-4$ in multiples of 4	0 to $2^{64}-16$ in multiples of 16	$2^{64}-1$	20
CBC-AES-256-HMAC-SHA-256	64*	16			$2^{64}-1$	32
CBC-AES-256-HMAC-SHA-512	96*	16			$2^{64}-1$	64
XTS-AES-256-HMAC-SHA-512	128*	16	0 to $2^{125}-128$	0 or 16 to $2^{68}-1$	$2^{68}-1$	64

\* Includes both AES key and MAC key

All lengths shall be an integer number of bytes (i.e. multiples of 8 bits).

A cryptographic unit may impose parameter limits that are more restrictive than those in Table 2. Documentation shall specify the parameter limits for the cryptographic unit, if different from those in Table 2.

**5.2 Counter mode with CBC-MAC (CCM)**

A cryptographic unit that supports the CCM-128-AES-256 mode shall use the algorithm specified by NIST Special Publication 800-38C<sup>4</sup> (hereafter referenced as the *CCM document*) with the following specifications:

- a) Block cipher algorithm: AES with a 256-bit cipher key (see NIST FIPS 197).
- b) Counter generation function as specified in Appendix A of the CCM document.
- c) Formatting function as specified in Appendix A of the CCM document.
- d) MAC length:  $Tlen=128$  bits.
- e) Nonce length: exactly 96 bits ( $n=12$  bytes). The IV input to the encryption procedure corresponds to the nonce  $N$  required by the CCM algorithm.
- f) The decryption allowance given by 4.4.
- g) Generate IVs according to 6.5

The data length shall be represented using three bytes, which is parameterized by setting  $t=16$ , and  $q=3$  (see A.2.1 within the CCM document). Table 3 shows the format of block  $B_0$ .

Table 3—Formatting of  $B_0$

<b>Byte Number:</b>	0	1...12	13...15
<b>Value:</b>	0y111010 <sub>2</sub>	Initialization Vector (IV)	Byte-length of plaintext

**Deleted:** A cryptographic unit may impose parameter limits that are more restrictive than those in Table 2. Documentation shall specify the parameter limits for the cryptographic unit, if different from those in Table 2.  
**Formatted:** Bullets and Numbering

<sup>4</sup> For information on references, see Clause 2.

1 In Table 3, the variable  $y$  shall equal a binary '0' if the AAD length is zero and a binary '1' if the AAD  
 2 length is non-zero. For example, the first byte of  $B_0$  has the binary value  $00111010_2$  if there is no AAD and  
 3  $01111010_2$  if there is AAD.

4  
 5 The *Flags* field within the *counter blocks* shall contain the binary value  $00000010_2$  (see A.3 within the  
 6 CCM document). All other parameters shall be as specified in Appendix A of the CCM document.

7 NOTE—The CCM document does not allow any plaintext to be returned if the MAC validation fails. This standard  
 8 makes an exception for this case.

Formatted: Bullets and Numbering

### 9 **5.3 Galois/counter mode (GCM)**

10 A cryptographic unit that supports GCM-128-AES-256 shall use the GCM algorithm specified in NIST SP  
 11 800-38D, with the following parameters:

- 12 a) Block cipher algorithm: AES with a 256-bit cipher key (see NIST FIPS 197).
- 13 b) Tag length: 128 bits.
- 14 c) IV computation algorithm as specified by McGrew and Viega's "The Galois/Counter Mode of  
 15 Operation" and with further requirements in 6.5.
- 16 d) The decryption allowance as given by 4.4.

17  
 18 The length of the IV shall be either 12 bytes or at least 16 bytes.

19  
 20 NOTE—Using an IV with more than 16 bytes does not add more security because a long IV is distilled back to 16  
 21 bytes before use.

22 NOTE—NIST SP 800-38D does not allow any plaintext to be returned if the MAC validation fails. This standard  
 23 makes an exception for this case.

Formatted: Bullets and Numbering

### 24 **5.4 Cipher Block Chaining with HMAC-SHA (CBC-HMAC-SHA)**

25 A cryptographic unit that supports the CBC-HMAC mode shall use the algorithm specified by NIST  
 26 Special Publication 800-38A and a HMAC as specified in NIST FIPS 198 with the following  
 27 specifications:

- 28 a) Block cipher algorithm: AES with a 256-bit (32-byte) cipher key (see NIST FIPS 197).
- 29 b) HMAC using SHA-1, SHA-256 or SHA-512
- 30 c) MAC length:  $Tlen=160$  bits (20 bytes), 256 bits (32 bytes), or 512 bits (64 bytes)
- 31 d) MAC key length shall be equal to the MAC length ( $Tlen$ )
- 32 e) Plaintext record length shall be a multiple of 16 bytes (see 1.1.1 for a discussion on padding)
- 33 f) AAD length shall be a multiple of 4 bytes
- 34 g) IV generation according to one of the following methods:
  - 35 1) Set the value of the IV entirely to the output of an RBG for each new encrypted record,  
 36 according to 6.5.2.
  - 37 2) Generate nonce IVs according to 6.5.3, and use the nonce IV as a nonce as specified in SP  
 38 800-38A, Appendix C. Specifically, the value of the IV in this case shall be encrypted  
 39 using the forward encryption algorithm of the AES-256 block cipher before being passed to  
 40 the CBC-HMAC-SHA procedure.
  - 41
  - 42 h) IV length of 16 bytes.
  - 43 i) The decryption allowance given by 4.4.

1 NOTE—Even though the plaintext record is required to be a multiple of 16 bytes, the host record may be any size,  
2 provided that the cryptographic unit provides padding.

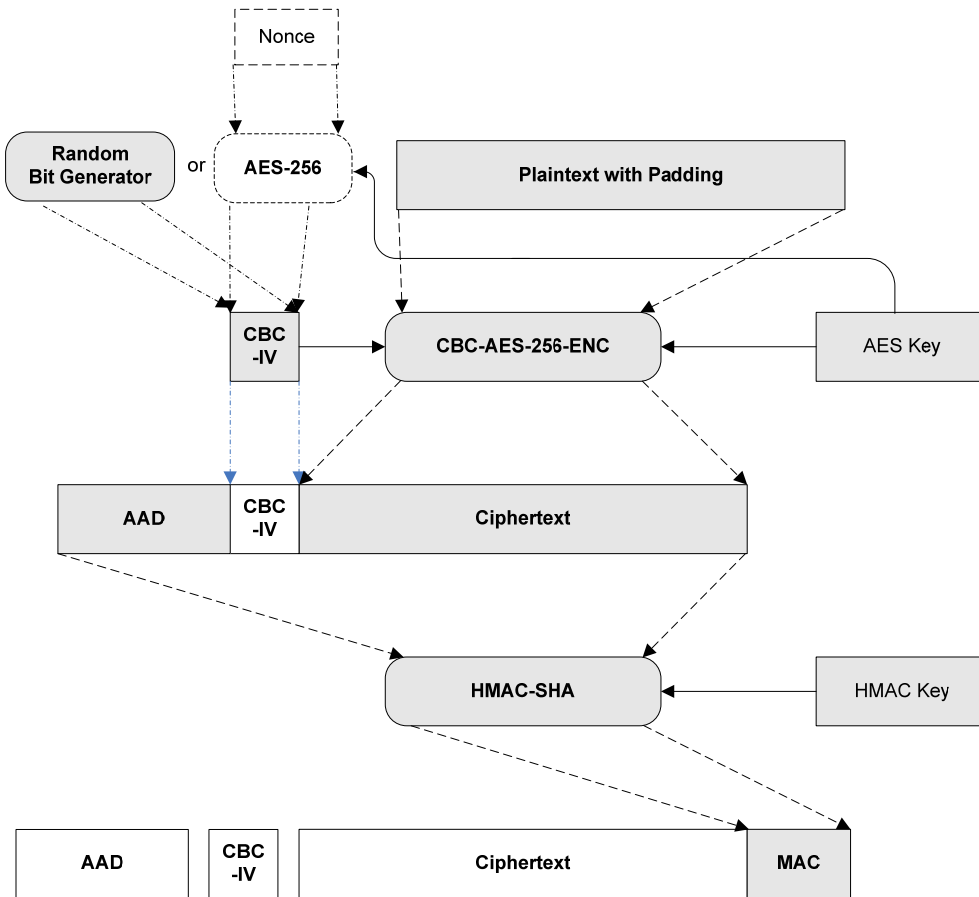
3  
4 The cryptographic unit shall compute the HMAC over the concatenation of the AAD, CBC-IV, and  
5 ciphertext. If the AAD has variable-length, then there shall be sufficient information within the AAD to  
6 allow the decryption routine to unambiguously determine where the AAD ends and the ciphertext starts.

7  
8 If the cryptographic unit supports CBC-HMAC-SHA, then documentation shall describe the format of the  
9 AAD and the method used to determine where the AAD ends and the CBC-IV starts. During decryption,  
10 the cryptographic unit shall use this method to determine where the AAD ends and shall send the special  
11 signal FAIL to the host if the AAD does not adhere to the documented format.

12 NOTE—It is possible to fulfill the previous requirement by including the length of the AAD within a fixed-length field  
13 at the beginning of the AAD.

14  
15 The cipher key length shall be 416 bits when using CBC-HMAC-SHA-1, 512 bits when using CBC-  
16 HMAC-SHA-256, and 768 bits when using CBC-HMAC-SHA-512. The cryptographic unit shall use the  
17 first 256-bits of the cipher key as the AES key in the encryption and decryption routines. The  
18 cryptographic unit shall use the remaining bits of the cipher key as the HMAC key in the MAC generation  
19 and verification routines.

20  
21 Figure 2 shows the CBC-HMAC encryption routine.  
22



**Figure 2**—Depiction of an CBC-HMAC-SHA encryption routine

Formatted: Bullets and Numbering

**5.5 Tweakable block-cipher with HMAC (XTS-AES-256-HMAC-SHA-512)**

A cryptographic unit that implements this mode shall use the AES-256-XTS algorithm as specified in IEEE P1619 to encrypt, and HMAC-SHA-512 as specified by FIPS PUB 198 and FIPS PUB 180-2 to generate the MAC, with the following specifications:

- a) Block cipher algorithm: AES with a 256-bit (32-byte) key (see NIST FIPS 197).
- b) 512 bits (64 bytes) of key material for XTS
- c) Integrity algorithm: HMAC-SHA-512 with 512-bit (64-byte) key.
- d) Generate IVs according to 6.5 to be used as the 128-bit tweaks as specified in IEEE P1619.

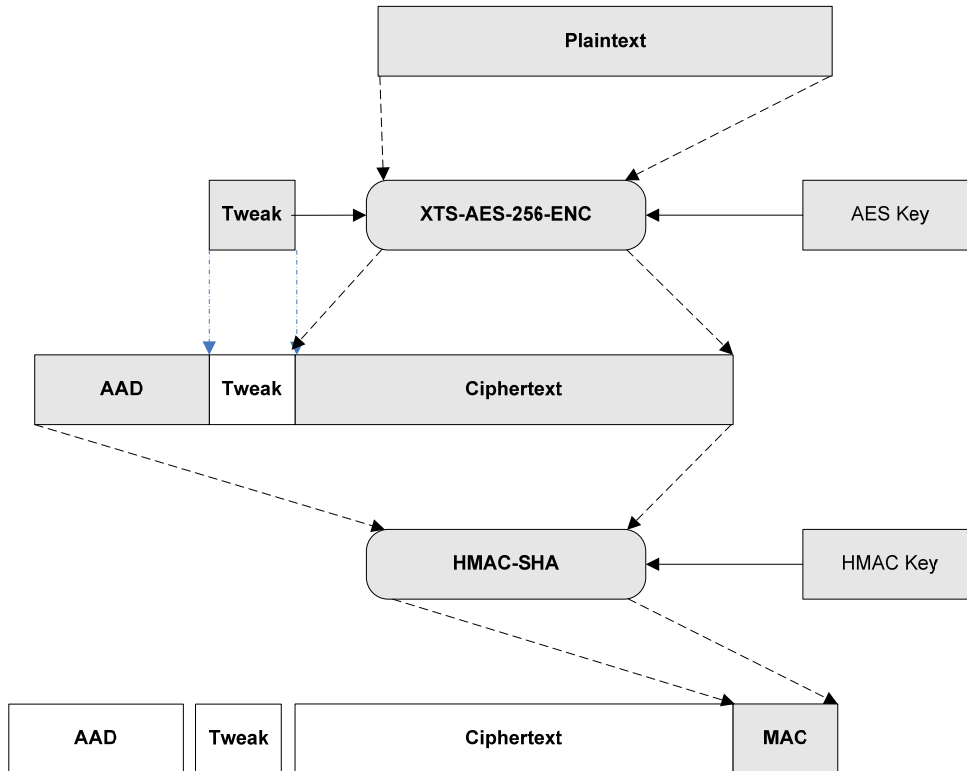
The cipher-key length shall be 1024 bits. The cryptographic unit shall use the first 512 bits of the cipher key as the XTS-AES-256 key in the encryption and decryption routines. The cryptographic unit shall use the remaining bits of the cipher key as the HMAC-SHA-512 key in the MAC generation and verification routines. In HMAC-SHA-512, the resulting MAC shall be 64 bytes long.

1 The cryptographic unit shall compute the HMAC over the concatenation of the AAD, tweak, and  
 2 ciphertext. If the AAD has variable-length, then there shall be sufficient information within the AAD to  
 3 allow the decryption routine to unambiguously determine where the AAD ends and the ciphertext starts.  
 4  
 5 During decryption, the cryptographic unit shall use this method to determine where the AAD ends and shall  
 6 send the special signal FAIL to the host if the AAD does not adhere to the documented format.

7 NOTE—It is possible to fulfill the previous requirement by including the length of the AAD within a fixed-length field  
 8 at the beginning of the AAD.

9 NOTE—Even though the plaintext record is required to be at least 16 bytes long, the host record may be smaller,  
 10 provided that the cryptographic unit provides padding.

11 Figure 3 shows the XTS-AES-256 encryption routine.  
 12  
 13



14  
 15 **Figure 3**—Depiction of an XTS-AES-256 encryption routine

Formatted: Bullets and Numbering

16 **6. Cryptographic key management and initialization vector requirements**

17 **6.1 Random bit generator**

18 If a cryptographic unit needs random data, then the cryptographic unit shall use a cryptographically-sound  
 19 random-bit generator (RBG) to generate this random data. In particular, the RBG shall be designed such  
 20 that it is not computationally feasible to predict subsequent outputs from the RBG based on knowledge of  
 21 previous outputs from the RBG or unencrypted information passed into the cryptographic unit.

1  
2 A cryptographic unit should implement an RBG that is compatible with NIST FIPS 140-2 or comparable  
3 standards (see [B19]). For implementation guidance describing suitable RBGs, see [B1], [B15], [B17], and  
4 [B23].  
5  
6 Documentation shall describe the RBG employed by the cryptographic unit, including algorithms and  
7 descriptions of sources of randomness.

### 8 6.2 Cryptographic key entry and export

9 A cryptographic unit should receive cryptographic keys from the key manager using a secure method (e.g.,  
10 physically secure interface, cryptographically protected communications).

11  
12 A cryptographic unit shall not make plaintext cryptographic keys available, except to authorized entities  
13 that use a physically secure port. A cryptographic unit may make encrypted (i.e., wrapped) cryptographic  
14 keys available externally.

15  
16 The key manager should refrain from entering the same cipher key into both an IEEE 1619.1-compliant  
17 cryptographic unit and a non-compliant cryptographic unit for purposes of encryption. In this situation, it  
18 is possible for the non-compliant cryptographic unit to compromise the security of the data (e.g., the non-  
19 compliant cryptographic unit may use the same sequence of IVs as the compliant cryptographic unit).

20  
21 If the cryptographic unit supports cryptographic key entry or export, then documentation shall specify the  
22 supported cryptographic key entry and export methods.

### 23 6.3 Handling the cipher key

24 The cryptographic unit shall use one or more of the following methods to create the cipher key:

- 25 a) Generate a new cipher key using only the output from an RBG and archive the cipher key as  
26 specified in 6.4.
- 27  
28 b) Use a cipher key from the key manager and include random information within the IV, as  
29 specified in 6.5.2 or 6.5.3.3.
- 30 c) Use a cipher key from the key manager and use unique IVs within a self-contained group (see  
31 6.6).

32  
33 The cryptographic unit may use key wrapping, as specified in 6.4, in conjunction with any of the items in  
34 the list above.

35 NOTE—The controller needs to be especially careful when configuring the cryptographic unit to use a cipher key from  
36 the key manager. In this case, it is important for the key manager to frequently generate new cipher keys because the  
37 risk of information leakage increases with the square of the amount of plaintext encrypted under the same cipher key  
38 (see B.7 and B.8).

### 42 6.4 Cryptographic key wrapping on the storage medium

43 Support of key wrapping is optional. The cryptographic unit may use key wrapping with any option within  
44 6.3

45  
46 When using key wrapping during encryption, the cipher key is wrapped with a KEK by either the  
47 cryptographic unit or the key manager, and the wrapped cipher key is then stored either on the storage  
48 medium or at the key manager.  
49

Deleted: user

Deleted: user

Deleted: user key and

Deleted: This subclause defines the interactions between the user key, which the key manager provides, and the cipher key used by the cryptographic unit.¶

Deleted: <#>GCM and CCM only: Transform the user key into a 256-bit cipher key, as specified in 6.4.¶

Formatted: Bullets and Numbering

Deleted: the user key directly as the

Deleted: the user key directly as the

Deleted: user

Deleted: use the user key directly as the

Deleted: user

Deleted: user

Deleted: ¶ Documentation shall specify the method for creating the cipher key from the user key in sufficient detail to allow a third party to implement the same algorithm.

Deleted: <#>Key-transform functions¶ <#>Overview¶ The following subclauses contain key-transform functions that the cryptographic unit may use. Each function transforms the user key, provided by the key manager, into a cipher key that the cryptographic unit uses for encryption or decryption. This process reduces the probability of cipher key and IV collisions among cryptographic units that are given the same user key. ¶ When using a key-transform function, the cryptographic unit shall perform the following steps before encrypting data with the cipher key:¶ ¶ <#>Generate a new nonce, using either an RBG if using KTF1 (see 6.4.2) or globally unique information if using KTF2 (see 6.4.3), KTF3 (see 6.4.4), or KTF4 (see 6.4.5).¶ <#>Pass the nonce and user key into the key-transform function and set the cipher key to its output.¶ <#>Either store on the storage medium sufficient information to allow reconstruction of the nonce, or use key-wrapping as described in 6.5 to archive the resulting cipher key. When using key wrapping in this case, the key manager provides both a KEK and user key to the cryptographic unit.¶

... [4]

Formatted: Bullets and Numbering

1 When using randomly generated cipher keys as per a) in 6.3, if the cipher key is wrapped by the  
2 cryptographic unit, then the cryptographic unit shall perform the following actions in order before  
3 encrypting data:

- 4
- 5 a) Create a wrapped cipher key using ~~a KEK from the key manager.~~
- 6 b) Save the wrapped cipher key using one or more of the following actions:
- 7 1) Store the wrapped cipher key on the storage medium
- 8 2) Export the wrapped cipher key to the key manager for archival.

Deleted: the user key as a

9  
10 When using key-wrapping during decryption, the cryptographic unit or key manager retrieves the wrapped  
11 cipher key (e.g., from the storage medium or key manager) and then unwraps it using the KEK.

12  
13 When unwrapping a wrapped key that was wrapped with an asymmetric public KEK, the key manager  
14 should not pass an asymmetric private KEK to the cryptographic unit. Instead, the key manager should  
15 retrieve the wrapped cipher key, use its asymmetric private KEK to unwrap it, and then pass the cipher key  
16 to the cryptographic unit using a secure method.

17  
18 Documentation shall describe any key-wrapping algorithm that the cryptographic unit supports.

19  
20 NOTE—The effective strength of the cipher key can be reduced if the key wrapping mechanism uses keys that are too  
21 short. See [B6], [B12], and [B21] for different estimates of equivalent key sizes between symmetric key and various  
22 asymmetric key encryption algorithms. If the wrapped key is exported to the key manager instead of stored on the  
23 storage medium and the storage medium is lost, then the effective strength of the cipher key is not reduced due to the  
24 key-wrapping algorithm. The effective strength of the cipher key is only reduced if the wrapped key is stolen.

25 *Examples of key wrapping algorithms:*

- 26
- 27 — NIST AES key wrap (see [B18]) with ~~a symmetric KEK.~~
- 28 — RSAES-OAEP (see [B24]) with ~~an RSA public key as the asymmetric public KEK for encryption and~~
- 29 ~~an RSA private key as the asymmetric private KEK for decryption.~~
- 30 — ECIES (see [B10]) with ~~an elliptic curve public key as the asymmetric public KEK for encryption and~~
- 31 ~~an elliptic curve private key as the asymmetric private KEK for decryption.~~

Deleted: the user key as the key-  
encryption key (

Deleted: )

Deleted: the user key as

Deleted: the user key as

Formatted: Bullets and Numbering

## 32 **6.5 Initialization Vector (IV) requirements**

### 33 **6.5.1 Overview**

34 Encrypting each plaintext record requires a cipher key and an IV, and using the same combination of cipher  
35 key and IV to encrypt more than one plaintext record introduces security vulnerabilities (see B.6). To  
36 minimize the chances of using the same combination of cipher key and IV to encrypt more than one  
37 plaintext record, the cryptographic unit shall generate the IVs according to one of the following methods:

- 38
- 39 a) **Random IV:** For each encrypted record, the cryptographic unit generates a new IV that consists
- 40 entirely of the output from an RBG (see 6.5.2)
- 41 b) **Nonce IV:** Use encryption sessions, according to 6.5.3.

42 Documentation shall specify the method for generating the IV.

Formatted: Bullets and Numbering

### 43 **6.5.2 Using random IVs**

44 A cryptographic unit may generate a random IV as input into each encrypted record. Such a random IV  
45 shall entirely consist of the output from an RBG.

1 **6.5.3 Encryption sessions**

Formatted: Bullets and Numbering

2 **6.5.3.1 Overview**

3 An encryption session is an interval in which one or more cryptographic units maintain a consistent  
4 sequence of IVs for encrypting plaintext records.

6 A cryptographic unit may maintain multiple independent encryption sessions simultaneously in which each  
7 independent encryption session uses a different cipher key and independent IVs.

Formatted: Bullets and Numbering

8 **6.5.3.2 Beginning of an encryption session**

9 Before starting an encryption session, the controller shall configure the cryptographic unit to use a  
10 particular method for creating or retrieving the cipher key, according to 6.3.

Deleted: user  
Deleted: association between the user key and

12 An encryption session shall begin only after one of the following events:

- 14 a) The cryptographic unit receives a cipher key from the key manager.
- 15 b) The cryptographic unit generates a new cipher key.

Deleted: user

17 **6.5.3.3 Encryption session IV requirements**

Formatted: Bullets and Numbering

18 The cryptographic unit shall encrypt each plaintext record with an IV that is unique within the encryption  
19 session. This requirement prevents plaintext leakage within an encryption session (see B.6).

Deleted: Error! Reference source not found.

21 When starting an encryption session, the cryptographic unit shall set the IV to an initial value. If the  
22 cryptographic unit uses a cipher key from the key manager (see 6.3), then the initial value for the IV shall  
23 contain at least 64 bits that are derived from an RBG. This initial value may continue from the last IV of  
24 the previous encryption session if the last IV is part of a consistent sequence that originally started from an  
25 initial value containing at least 64 random bits.

Deleted: directly  
Deleted: the user key as the

27 If the IV contains two or more fields with variable length, then the IV shall contain sufficient information  
28 to unambiguously determine the length of each variable-length field.

Deleted: If multiple cryptographic units use a key-transform function with a common key-transform nonce, then each IV shall be unique among all these cryptographic units (e.g., by including a cryptographic unit identifier within the IV).¶  
¶

30 If the IV is not written to the storage medium, then documentation shall describe the format of the IV and  
31 the cryptographic unit's mechanism for generating each IV.

32 **6.5.3.4 End of an encryption session**

Formatted: Bullets and Numbering

33 An encryption session shall end after one of the following events:

- 35 a) The cryptographic unit loses the encryption session state, including the cipher key and IV. In this  
36 case, the cryptographic unit should notify the host.
- 38 b) The cryptographic unit is unable to create an IV that is unique within the encryption session. In  
39 this case, the cryptographic unit should send the special signal FAIL to the host.

41 If a cryptographic unit is encrypting data in encryption sessions and the encryption session ends, then the  
42 cryptographic unit shall not encrypt any more data until another encryption session starts.

44 The cryptographic unit may clear its encryption session state based on a command received from the host.  
45 Documentation shall describe such a command, if supported.

Formatted: Bullets and Numbering

46 **6.6 Creating unique IVs within a self-contained group**

47 This subclause defines requirements for creating unique IVs within a self-contained group of cryptographic  
48 units. Support of these requirements is mandatory for cryptographic units that support 6.3 c), and optional  
49 otherwise.

51 When creating unique IVs within a self-contained group, the following statements apply:

- 1 | a) Cryptographic units may share a common ~~cipher~~ key. Deleted: user  
2 | b) Compliance to this standard shall only apply to the entire self-contained group, not an individual  
3 | cryptographic unit within the group.  
4 | c) All cryptographic units within the self-contained group shall be configured to coordinate the  
5 | creation of unique IVs.  
6 | d) Cryptographic units shall not share ~~cipher~~ keys with any cryptographic unit that is not a member Deleted: user  
7 | of the self-contained group unless such a non-member of the self-contained group includes  
8 | random information within the IV as defined in 6.3 b).  
9 | e) Cryptographic units shall only be used with external services that ensure and document  
10 | compliance with statement d).  
11 | f) Documentation shall describe how the system prevents reuse of the same IV between any two  
12 | cryptographic units within the self-contained group and how the cryptographic units are  
13 | uniquely identified. Such identification should use cryptographic methods.

1 **Annex A**

2 (informative)

3 **Bibliography**

- 4 [B1] ANSI X9.31:1998, Digital Signatures Using Reversible Public Key Cryptography for the Financial  
5 Services Industry (rDSA), 1998.
- 6 [B2] ANSI X9.63:2001, Public Key Cryptography For the Financial Services Industry: Key Agreement  
7 and Key Transport Using Elliptic Curve Cryptography, 2001.
- 8 [B3] Biham, E., "New Types of Cryptanalytic Attacks Using Related Keys," *Advances in Cryptology -*  
9 *EUROCRYPT'93*, Springer-Verlag, 1994, pp. 398-409. Deleted:
- 10 [B4] Canetti, Ran, Shai Halevi, and Michael Steiner. "Mitigating Dictionary Attacks on Password-  
11 Protected Local Storage." *Advances in Cryptology - CRYPTO '06*. LNCS vol. 4117, pages 160-179.  
12 Springer-Verlag, *available from the World Wide Web site* <<http://eprint.iacr.org/2006/276>>, 2006.
- 13 [B5] ECMA-321, Streaming Lossless Data Compression Algorithm (SLDC), June 2001.
- 14 [B6] ECRYPT IST-2002-507932 D.SPA.16, ECRYPT Yearly Report on Algorithms and Keysizes  
15 (2005), Jan 2006.
- 16 [B7] Ferguson, Niels, Russ Housley, and Doug Whiting, Submission to NIST: Counter with CBC-MAC  
17 (CCM) AES Mode of Operation, *available from the World Wide Web site*  
18 <<http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ccm/ccm.pdf>>.
- 19 [B8] Ferguson, Niels, Authentication weaknesses in GCM, *available from the World Wide Web site*  
20 <<http://csrc.nist.gov/CryptoToolkit/modes/comments/CWC-GCM/Ferguson2.pdf>>.
- 21 [B9] IEEE std 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)  
22 specifications.
- 23 [B10] IEEE std 1363a-2004, IEEE Standard Specifications for Public-Key Cryptography Amendment 1:  
24 Additional Techniques, 2004.
- 25 [B11] IETF RFC 2898, PKCS #5: Password-Based Cryptography Specification Version 2.0., *available*  
26 *from the World Wide Web site* <<http://www.ietf.org/rfc/rfc2898.txt>>, September 2000.
- 27 [B12] IETF RFC 3766, Determining Strengths For Public Keys Used For Exchanging Symmetric Keys,  
28 April 2004.
- 29 [B13] INCITS/T10, Information Technology - SCSI Stream Commands - 3 (SSC-3).
- 30 [B14] ISO/IEC 8824-1:2002, Information Technology – Abstract Syntax Notation One (ASN.1):  
31 Specification of Basic Notation. Equivalent to ITU-T Rec. X.680.
- 32 [B15] ISO/IEC 18031, Information Technology – Security techniques – Random bit generation, November  
33 2005.
- 34 [B16] ISO/IEC 18033-2:2006, Information technology – Security techniques – Encryption algorithms –  
35 Part 2: Asymmetric ciphers, 2006.

- 1 [B17] Keller, Sharon S., NIST-Recommended Random Number Generator Based on ANSI X9.31  
2 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, January 2005.
- 3 [B18] NIST AES Key Wrap Specification, November 2001.
- 4 [B19] NIST FIPS 140-2, Federal Information Processing Standard 140-2, Announcing the Standard for  
5 Security Requirements for Cryptographic Modules.
- 6 [B20] NIST Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes  
7 Using Discrete Logarithm Cryptography, March 2006.
- 8 [B21] NIST Special Publication 800-57, Recommendation for Key Management – Part 1: General  
9 (Revised), May 2006.
- 10 [B22] NIST Special Publication 800-63, Electronic Authentication Guideline: Recommendations of the  
11 National Institute of Standards and Technology, April 2006.
- 12 [B23] NIST Special Publication 800-90, Recommendation for Random Number Generation Using  
13 Deterministic Random Bit Generators.
- 14 [B24] RSA PKCS #1 v2.1: RSA Cryptography Standard, June 2002.

## 1 **Annex B**

2 (informative)

### 3 **Security concerns**

#### 4 **B.1 Threat model**

5 This standard is meant to protect stored data in settings where an attacker might have full access to the  
6 storage medium: It is assumed that the attacker might be able to read the content of the storage medium and  
7 can also write to it, including replacing some of the stored data with arbitrary data of the attacker's  
8 choosing. It is also assumed that the attacker might have access to very large amounts of encrypted data.  
9 Such a threat model is suitable for situations where the storage medium is not tightly bound to the  
10 cryptographic unit. A prime example is tape encryption, where it is expected that encrypted cartridges are  
11 routinely accessed separately from the tape drive where they were first written.

12  
13 Furthermore, this standard is meant to offer some protection even in highly adversarial situations where an  
14 attacker can have repeated access to the storage media of a live system, can monitor or modify the storage  
15 as it is repeatedly being written and over-written. However, only limited protection is provided against  
16 replay attacks (see C.2).

17  
18 Beyond watching and modifying ciphertext, the attacker may have some known- or chosen-plaintext  
19 capabilities. This means that the attacker may have some a-priori knowledge of the plaintext corresponding  
20 to ciphertext that is written on the storage medium, and it may even be able to influence the host into  
21 writing plaintext records containing text of the attacker's choosing. This is a realistic assumption when the  
22 host represents a multi-user system and some of these users are not highly trusted, and can be realistic even  
23 in a single user cases (e.g., when the content of a web cache is written to encrypted disk).

24  
25 On the other hand, this threat model does not cover security of the information in transit (i.e., how the  
26 cryptographic unit receives and sends data to be securely stored). It also does not cover most aspects of key  
27 management, such as the generation, transfer, and secure storage of the keys (and does not address mis-  
28 management of keys such as using a cryptographic transformation to encrypt its own key). Many of these  
29 aspects are addressed by other standards. Finally, a variety of physical (side-channel) attacks against the  
30 cryptographic unit, such as timing, power, radiation, fault injection, and the design of a secure random-bit  
31 generator (RBG) are out of scope.

#### 32 **B.2 Lack of security when using passwords as cryptographic keys**

33 The security of a cryptographic unit depends on high-quality cryptographic keys. Ideally, the  
34 cryptographic keys should come from a cryptographically-sound random-bit generator. The user should  
35 not use sources that lack randomness, such as passwords, for the user key. It is relatively easy for an  
36 attacker to launch an off-line dictionary attack against passwords. NIST provides guidelines that estimate  
37 the amount of randomness within common passwords (see [B22]). See also [B4] and [B11].

#### 38 **B.3 Replay attacks**

39 An implementer should consider adding requirements of keeping the record numbers in proper sequence on  
40 a particular storage medium. Otherwise, the cryptographic unit is vulnerable to a *replay attack*, where the  
41 attacker replaces a record on the media with some other properly-authenticated record (such as a prior  
42 version of the current record).

43

1 The attack is applicable in environments where the attacker can read and write directly to the storage  
 2 medium, and can reorder or repeat encrypted records, assuming the cryptographic unit does not validate the  
 3 ordering. This could be a powerful attack if the adversary has extensive knowledge of the plaintext and  
 4 wishes to change the contents of a backup set by replacing specific records with other records, all encrypted  
 5 with the same cipher key.

6  
 7 Many network encryption standards, such as IPsec, use sequence numbers to handle replay attacks.  
 8 Similarly, a cryptographic unit should keep the records in sequential order. This could be simply  
 9 accomplished by including the sequential record number within the AAD or IV fields.

10  
 11 Maintaining a sequential record number helps, but does not handle the case of append operations that  
 12 overwrite previous data. The addition of a write-pass counter helps ensure that the data has both the correct  
 13 record number and was written in the correct sequence. Otherwise, an attacker could replay records from  
 14 either a different tape or a previous write pass of the current tape. A validated write-pass count would  
 15 prevent this attack.

16  
 17 Unfortunately, including the record number and write-pass count within the AAD or IV field would make it  
 18 hard, if not impossible, to perform a direct copy of the raw encrypted data from one tape to another.  
 19 Transferring encrypted data would only be possible by performing an entire tape copy.

20  
 21 Ordering verification does not seem to be applicable in environments where the host has random-access to  
 22 the storage, such as in a hard drive. Protecting against replay attacks in these environments must therefore  
 23 be done by other means, such as controlling the access to the media when possible, or relying on a higher-  
 24 level application. This is beyond the scope of this standard.

25  
 26 Some level of operational protection against replay attacks might be provided by not allowing direct (raw)  
 27 read and write of encrypted records on the storage medium. This increases the difficulty for attackers by  
 28 depriving them of ready-made tools to use in these attacks, instead forcing them to implement their own  
 29 tools. It is clear, however, that this operational protection is only effective against casual attackers.  
 30 Determined, well-financed attackers can always build their own tools.

#### 31 **B.4 Passing plaintext to the host before checking the MAC**

32 This standard makes an allowance for cryptographic units to pass partially decrypted plaintext to the host  
 33 before checking the message authentication code (MAC). The purpose is to allow implementations to be  
 34 compliant even if they are unable to store an entire decrypted record before passing plaintext back to the  
 35 host. Note, however, that such implementations might have difficulty in gaining certifications such as FIPS  
 36 140-2 (see [B19]). Additionally, both SP 800-38C and 800-38D require that the device validate the MAC  
 37 before returning any plaintext.

38  
 39 Because of this allowance, it is important that the host does not act on any decrypted plaintext before the  
 40 MAC validation finishes. Data encrypted using CTR (counter) mode is especially malleable to an attacker  
 41 because flipped bits in the ciphertext directly flip corresponding bits in the plaintext. It is also easy to  
 42 modify any CRC embedded within the plaintext because CRC residuals are linear and depend only on the  
 43 other modified bits within the ciphertext. If the attacker has knowledge of the plaintext, it is easy to make  
 44 arbitrary changes to compromise the data. Note that CBC is significantly less vulnerable and XTS is not  
 45 vulnerable to this attack.

46  
 47 The cryptographic unit can provide operational protection against casual attackers by not implementing  
 48 commands that allow direct (raw) read and write of encrypted records on the storage medium (see B.3).

#### 49 **B.5 Checking for integrity of a cryptographic key**

50 Using a corrupted cipher key by the cryptographic unit might lead to wasted resources, and might even  
 51 open an exposure to related-key attacks [B3]. For example, when using one of the HMAC modes in this

1 standard (see 5.4 and 5.5), a cipher key in which only some of the first 256 bits are corrupted would cause  
 2 an encrypted record to pass the MAC-check but still be decrypted to something other than the original  
 3 plaintext record.

4  
 5 To avoid using corrupted keys, some measures should be taken to verify the integrity of the cipher key in  
 6 use. One method for ensuring key integrity is using a “key signature”. For example, this could be an  
 7 HMAC computed over the cipher key using some higher-level key. For further guidance, see [B21].

8 **B.6 Avoiding collisions of initialization vectors**

9 All the modes that are specified in this standard rely on an initialization vector (IV) that is assumed to be  
 10 non-repeating within the scope of the cipher key in use. In all of them, using the same combination of  
 11 cipher key and IV to encrypt two different records (referred to as an IV-collision) result in some exposure  
 12 to attacks and leaks information about the plaintext.

13  
 14 This exposure is particularly acute in CCM and GCM that use counter-mode encryption (i.e. a block cipher  
 15 turned into a stream cipher). In these modes, re-use of the same IV under the same cipher key poses the  
 16 same risk as re-use of the key stream in a stream cipher – namely, the exclusive-or of the two ciphertext  
 17 records equals to the XOR of the two plaintext records, allowing an attacker to learn information about the  
 18 plaintext records by observing the ciphertext. Moreover, an IV collision in GCM might in some  
 19 circumstances reveal information about the authentication key (which is generated internal to the GCM  
 20 algorithm) to an attacker, thus allowing the attacker to forge authentication tags (see [B8]). This exposure  
 21 is less acute in the other two modes (CBC and XTS), but it exists even there. For example, an IV collision  
 22 lets the attacker see if two encrypted records are the same, or even if some specific blocks in these records  
 23 are the same. To maintain the security of the encryption modes in this standard, it is therefore important to  
 24 take proactive steps to avoid IV collisions.

25  
 26 When considering the probability of IV collisions, it is important to take into account the possibility that  
 27 the same user key is loaded into cryptographic units from different manufacturers, and that these  
 28 cryptographic units may use very different strategies for IV collision avoidance. Therefore, it is important  
 29 that each cryptographic unit guarantees some level of IV collision security, regardless of the behavior of  
 30 any other cryptographic units that may be given the same user key, or there is a guarantee that the key  
 31 manager and cryptographic unit maintains a consistent state between the user key and IV (see 6.5).

32  
 33 Some examples of IV collision avoidance strategies are described in B.7, along with analysis of their  
 34 effectiveness in various settings.

35 **B.7 Examples of IV collision avoidance strategies**

36 **B.7.1 Example 1: Using random IVs**

37 | In this example, the cryptographic unit receives a ~~cipher key~~ from the key manager, ~~to be used with some~~  
 38 cryptographic mode that employs  $n$ -bit IVs. With each encrypted record, the cryptographic unit extracts  $n$   
 39 bits from its RBG for use as the record’s IV.

**Deleted:** user  
**Deleted:** and uses that user key directly as the cipher key

40  
 41 To analyze the effectiveness of this strategy, one can observe that any two IVs assume the same value with  
 42 probability exactly  $2^{-n}$  (since all the IV’s are random). When encrypting  $c$  records with the same cipher key,  
 43 we therefore have  $c(c-1)/2$  (i.e.,  $c$  choose 2) possible collision events, each occurring with probability  $2^{-n}$ .  
 44 | Using Boole’s inequality (the union bound), one can derive the upper-bound on the probability of IV  
 45 collisions given in Equation (1)

**Deleted:** [need reference to define Boole’s inequality and union bound; these are not in the IEEE dictionary]  
**Formatted:** Bullets and Numbering

46 
$$p \leq \frac{c(c-1)}{2^{n+1}} < \frac{c^2}{2^{n+1}} \tag{1}$$



1 inequality to upper-bound the probability that any two of them give rise to collisions, as shown in Equation  
 2 (3).

Formatted: Bullets and Numbering

3 
$$p < \left( \frac{S(S-1)}{2} \right) \left( \frac{R}{2^{n-1}} \right) < \frac{S^2 R}{2^n}$$
 (3)

4 where

- 5  $p$  is the probability of any IV collision occurring
- 6  $S$  is the number of encryption sessions under a particular cipher key
- 7  $R$  is the number of records that are encrypted in every encryption session
- 8  $n$  is the IV length, measured in bits

9  
 10 For example, with  $S=2^{25}$  encryption sessions, each with  $R=2^{15}$  encrypted records (so the total is still  $2^{40}$   
 11 encrypted records), we get an upper bound of  $p < 2^{-63}$  when using 128-bit IVs and  $p < 2^{-31}$  when using 96-bit  
 12 IVs (compared to the bounds of  $2^{-49}$  and  $2^{-17}$ , respectively, that we got from Equation (1)). The same  
 13 argument as in C.6.1 can be used here too to show that this bound is quite tight.

14 **B.7.3 Example 3: Randomizing only the key**

15 In this example, the cryptographic unit chooses a fresh cipher key for every encryption, and uses the integer  
 16  $I$  as the IV for the  $I$ th record in an encryption session. Clearly, no collisions are possible between IVs in the  
 17 same encryption session, and therefore the only risk is key-collision, which happens with negligible  
 18 probability (since the encryption keys are at least 256-bit long).  
 19

Deleted: Transforming the key with unique identifiers

20 **B.7.4 Example 4: Using the key within a self-contained group**

21 Consider a cryptographic unit without an RBG, but that can provide unique identifiers with each write pass.  
 22 The cryptographic unit then uses in each encryption session a nonce from a self-contained group (see 6.6)  
 23 in lieu of a random nonce, and uses the integer  $I$  as the IV for the  $I$ th record in a write-pass.  
 24

Deleted: the

Deleted: key-transform procedure from 1.1 with globally unique nonce

25 In this example we can no longer use Equation (1) to upper-bound the collision probability, since there no  
 26 randomness involved in the key- or IV-derivation. However, in a properly configured system the key-  
 27 transform function will always be used with unique nonces, hence the resulting cipher key is unlikely to  
 28 ever repeat. In fact, as long as the cipher key remains only within the self-contained group, one can use the  
 29 same bound as in the example from C.6.2. It should be noted, however, that this configuration is more  
 30 sensitive to mis-configuration error than the configurations above that use randomness as their means for  
 31 collision avoidance.

Deleted: globally unique nonce is indeed unique

32 **B.8 How many records to encrypt with one key?**

33 The bounds from the examples in B.7 can be used as guidance for the maximum amount of data to encrypt  
 34 with a single cipher key. Specifically, given the maximum acceptable probability of an IV-collision, and  
 35 knowledge of the cryptographic mode and the collision-avoidance strategy used by the cryptographic unit,  
 36 one may set an upper bound for data to encrypt with a single cipher key.  
 37

Deleted: user

38 Table B.1 below contains the maximum number of records for a single key for the collision-avoidance  
 39 strategies in the examples from B.7.1 and B.7.2. For random IVs we use the expression from Equation (2)  
 40 (which is heuristically a bit more accurate than the expression in Equation (1) for large values of  $\epsilon$ ), and for  
 41 incrementing random IVs we use the expression from Equation (3) with  $R=2^{15}$  (i.e., assuming that each  
 42 encryption session is used to encrypt  $2^{15}$  records).  
 43

44 It should be stressed, however, that the bounds in Table B.1 only consider the probability of IV-collisions,  
 45 and in most settings there are many other considerations that must be taken into account. For example, 5.1

1 includes some other limits on the amount of data that can be encrypted per cipher key, and XTS and CBC  
 2 encryption modes entail their own limitations, as discussed in their respective standards (IEEE std 1619 and  
 3 FIPS PUB 800-38A, respectively).  
 4

5 **Table B.1—Maximum number of encrypted records per key**

Maximum acceptable probability of IV-collisions	96-bit IVs		128-bit IVs	
	Equation (2)	Equation (3), $R=2^{15}$	Equation (2)	Equation (3), $R=2^{15}$
$p = 2^{-40}$ (~1 in $10^{12}$ )	$\sim 3.80 \times 10^8$	$\sim 4.86 \times 10^{10}$	$\sim 2.49 \times 10^{13}$	$\sim 3.18 \times 10^{15}$
$p = 2^{-30}$ (~1 in $10^9$ )	$\sim 1.21 \times 10^{10}$	$\sim 1.55 \times 10^{12}$	$\sim 7.96 \times 10^{14}$	$\sim 1.02 \times 10^{17}$
$p = 2^{-20}$ (~1 in $10^6$ )	$\sim 3.89 \times 10^{11}$	$\sim 4.97 \times 10^{13}$	$\sim 2.55 \times 10^{16}$	$\sim 3.26 \times 10^{18}$
$p = 2^{-10}$ (1 in 1024)	$\sim 1.24 \times 10^{13}$	$\sim 1.59 \times 10^{15}$	$\sim 8.15 \times 10^{17}$	$\sim 1.04 \times 10^{20}$
$p = 0.5$	$\sim 3.98 \times 10^{14}$	$\sim 3.60 \times 10^{16}$	$\sim 2.61 \times 10^{19}$	$\sim 2.36 \times 10^{21}$

6  
7

1 **Annex C**  
 2 (informative)

3 **Documentation Requirements summary**

4 Table C.1 summarizes the documentation required for compliance to this standard.  
 5

6 **Table C.1—Documentation requirements**

Documentation description	See
<del>An implementer of this standard shall provide documentation to the end-user with the cryptographic unit.</del>	<del>4.1</del>
<del>Documentation shall describe how the plaintext record formatter generates plaintext records from host records.</del>	<del>4.2.4.2</del>
<del>Documentation shall describe how the plaintext record consolidator generates host records from plaintext records.</del>	<del>4.2.4.3</del>
Documentation shall disclose whether the cryptographic unit validates the MAC before returning any plaintext	4.4.2
Documentation shall define the special signal <i>FAIL</i> and describe how the host receives such a signal. The special signal <i>FAIL</i> should identify the records that failed the MAC validation.	4.4.2
Documentation shall define the special signal <i>PASS</i> and describe how the host receives such a signal.	4.4.2
If a cryptographic unit supports ordering-verification, then documentation shall specify the methods for enabling or disabling this functionality, and shall specify how the cryptographic unit notifies the host of inconsistent IV or AAD ordering.	4.4.3
Documentation shall specify the parameter limits for the cryptographic unit, if different from those in Table 2	5.1
If the cryptographic unit supports CBC-HMAC-SHA, then documentation shall describe the format of the AAD and the method used to determine where the AAD ends and the CBC-IV starts	5.4
If the cryptographic unit supports XTS-HMAC-SHA, then documentation shall describe the format of the AAD and the method used to determine where the AAD ends and the Tweak starts.	5.5
Documentation shall describe the RBG employed by the cryptographic unit, including algorithms and descriptions of sources of randomness.	6.1
If the cryptographic unit supports cryptographic key entry or export, then documentation shall specify the supported cryptographic key entry and export methods.	6.2
Documentation shall specify the method for creating the cipher key from the user key in sufficient detail to allow a third party to implement the same algorithm.	6.3
Documentation shall describe any key-wrapping algorithm that the cryptographic unit supports.	6.4
Documentation shall specify the method for generating the IV.	6.5.1
If the IV is not written to the storage medium, then documentation shall describe the format of the IV and the cryptographic unit's mechanism for generating each IV.	6.5.3.3
The cryptographic unit may clear its encryption session state based on a command received from the host. Documentation shall describe such a command, if supported.	6.5.3.4
When creating unique IVs within a self-contained group, documentation shall describe how the system prevents reuse of the same IV between any two cryptographic units within the self-contained group and how the cryptographic units are uniquely identified	6.6

**Deleted:** 4.1

**Deleted:** Overview

**Deleted:** 1.1.1

**Deleted:** Documentation shall describe the mapping between host records and plaintext records.

**Deleted:** 1.1.1

**Deleted:** If a cryptographic unit supports padding of the plaintext, then documentation shall describe the padding algorithm and the method for determining the length of the host record, using only information protected by the MAC.

**Deleted:** If a cryptographic unit supports key-transform, then documentation shall specify the format of the nonce used in the key-transform function. ... [5]

7



**Deleted: <#>**  
 (informative)  
**P1619.1 Standard encryption interchange format¶**  
**<#>Overview¶**  
 This annex describes a format that could facilitate interchange of encrypted records. Supporting this format is optional.¶  
 ¶  
 All fields within this format are aligned to 4-byte boundaries. If a particular sub-field is not aligned to 32-bits, then the formatter shall pad to 4 bytes using zeros. All integers are in big-endian format; the most significant bits reside in the first byte (see 3.4).¶  
 ¶  
 The key manager provides a user key as input to the cryptographic unit. The user key may be in any format.¶  
**<#>Format description¶**  
**<#>Format table¶**  
 Table D.1 shows the elements that constitute the encryption interchange format. Each element in this table is in order, as it would appear within a data file or packet.¶  
 ¶  
 The length fields in this format are not necessarily authenticated, so care should be taken when using them. The documentation of the cryptographic unit describes how to authenticate the length fields.¶  
 ¶  
**Table D.1—Elements of the encryption interchange format¶**  
 Name ... [6]

**Formatted:** Bullets and Numbering

**Annex D**

(informative)

**Test vectors****D.1 General**

A cryptographic unit should test its cryptographic functions using test vectors included within this standard and/or within the reference documents. Table D.1 shows the recommended test vectors for each chosen cryptographic mode:

Deleted: Error! Reference source not found.

**Table D.1—Recommended test vectors**

Algorithm	Recommend test vectors
CCM-128-AES-256 (see 5.2)	See D.2
GCM-128-AES-256 (see 5.3)	See D.3
CBC-AES-256-HMAC-SHA-1 (see 5.4)	See D.4
CBC-AES-256-HMAC-SHA-256 (see 5.4)	See D.5
CBC-AES-256-HMAC-SHA-512 (see 5.4)	See D.5
XTS-AES-256-HMAC-SHA-512 (see 5.5)	See D.6

For all of the test vectors, the following abbreviations apply:

KEY	256-bit AES key
HMK	160, 256 or 512-bit HMAC key
NON	Nonce
CIV	Initialization vector for CBC-HMAC-SHA modes
IV	Initialization vector
HDR	AAD (Additional Authentication Data)
RPT	Repeat the previous HDR (AAD) a given number of times
PTX	Plaintext
CTX	Ciphertext
TAG	MAC (Message Authentication Code)
KEY1	(XTS only) first 256 bits of the cipher key
KEY2	(XTS only) second 256 bits of the cipher key
KEY3	(XTS only) last 512 bits of the cipher key
DUS	Data unit sequence number

For readability, the examples explicitly parse the XTS-AES key into Key1 and Key2. Key 3 is the XTS-AES HMAC key.

All numbers within the XTS test vectors are in little-endian bit order (same as IEEE P1619). The base for these numbers is hexadecimal.

All numbers within all other test vectors are in big-endian bit order, in which the most significant byte is on the left. The base for these numbers is hexadecimal, except for the 'RPT' field, which is in decimal. Within a particular test vector, if multiple lines start with the same prefix, these lines are concatenated.

For all test vectors, each pair of hexadecimal digits is grouped into a byte such that the left digit is the most significant and the right digit is the least significant.



1 RPT 0256  
2 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
3 CTX 04f883aeb3bd0730eaf50bb6de4fa2212034e4e41b0e75e577f6bf2422c0f6d2  
4 TAG 3376d2cf256ef613c56454cbb5265834

Formatted: Bullets and Numbering

5 **D.2.7 CCM test vector 7**

6 KEY 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
7 IV 101112131415161718191a1b  
8 HDR 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
9 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f  
10 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
11 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
12 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f  
13 PTX 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f  
14 PTX a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefb  
15 PTX c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeedf  
16 PTX e0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2f3f4f5f6f7f8f9fafbfcfdfefff  
17 CTX 24d8a38e939d2710cad52b96fe6f82010014c4c43b2e55c557d69f0402e0d6f2  
18 CTX 06c53d6cbd3f1c3c6de5dcdcad9fb74f25741dea741149fe4278a0cc24741e86  
19 CTX 58cc0523b8d7838c60fblde4b7c3941f5b26dea9322aa29656ec37ac18a9b108  
20 CTX a6f38b7917f5a9c398838b22afbd17252e96694a9e6237964a0eae21c0a6e152  
21 CTX 15a0e82022926be97268249599e456e05029c3ebc07d78fc5b4a0862e04e68c2  
22 CTX 9514c7bdafc4b52e04833bf30622e4eb42504a44a9dcbc774752de7bb82891ad  
23 CTX 1eba9dc3281422a8aba8654268d3d9c81705f4c5a531ef856df5609a159af738  
24 CTX eb753423ed2001b8f20c23725f2bef18c409f7e52132341f27cb8f0e79894dd9  
25 TAG ebb1fa9d28ccfe21bdfea7e6d91e0bab

Formatted: Bullets and Numbering

26 **D.2.8 CCM test vector 8**

27 KEY fb7615b23d80891dd470980bc79584c8b2fb64ce6097878d17fce45a49e830b7  
28 IV dbd1a3636024b7b402da7d6f  
29 HDR 36  
30 PTX a9  
31 CTX 9d  
32 TAG 3261b1cf931431e99a32806738ecbd2a

Formatted: Bullets and Numbering

33 **D.2.9 CCM test vector 9**

34 KEY f8d476cfd646ea6c2384cb1c27d6195dfef1a9f37b9c8d21a79c21f8cb90d289  
35 IV dbd1a3636024b7b402da7d6f  
36 HDR 7bd859a247961a21823b380e9fe8b65082ba61d3  
37 PTX 90ae61cf7baebd4cade494c54a29ae70269aec71  
38 CTX 6c05313e45dc8ec10bea6c670bd94f31569386a6  
39 TAG 8f3829e8e76ee23c04f566189e63c686

Formatted: Bullets and Numbering

40 **D.3 GCM test vectors**

41 **D.3.1 GCM test vector 1**

42 KEY 00  
43 IV 0000000000000000000000000000  
44 PTX 00000000000000000000000000000000  
45 CTX cea7403d4d606b6e074ec5d3baf39d18  
46 TAG d0d1c8a799996bf0265b98b5d48ab919





```

1 HDR 688e1aa984de926dc7b4c47f44
2 PTX a2aab3ad8b17acdda288426cd7c429b7ca86b7aca05809c70ce82db25711cb53
3 PTX 02eb2743b036f3d750d6cf0dc0acb92950d546db308f93b4fff244afa9dc72bcd
4 PTX 758d2c
5 CTX ee62552aebc0c3c7daae12bb6c32ca5a005f4a1aaab004ed0f0b30abbf15acf4
6 CTX c50c59662d4b4468419544e7f981973563ce556ae50859ee09b14d31a053986f
7 CTX 9ac89b
8 TAG 9cd0db936e26d44be974ba868285a2e1

```

Formatted: Bullets and Numbering

9 **D.4 CBC-HMAC-SHA-1 Test Vectors**

10 **D.4.1 AES CBC-HMAC-SHA-1 test vector 1**

```

11 KEY 0000000000000000000000000000000000000000000000000000000000000000
12 HMK 0000000000000000000000000000000000000000000000000000000000000000
13 CIV 0000000000000000000000000000000000000000000000000000000000000000
14 PTX 0000000000000000000000000000000000000000000000000000000000000000
15 CTX dc95c078a2408989ad48a21492842087
16 TAG 59bb230e817ad3f377d623d2ca97eefdf0fd467c

```

Formatted: Bullets and Numbering

17 **D.4.2 AES CBC-HMAC-SHA-1 test vector 2**

```

18 KEY 0000000000000000000000000000000000000000000000000000000000000000
19 HMK 0000000000000000000000000000000000000000000000000000000000000000
20 CIV 0000000000000000000000000000000000000000000000000000000000000000
21 HDR 0000000000000000000000000000000000000000000000000000000000000000
22 TAG 66040990c7992a2a00d037d0b8631c0db1785897

```

Formatted: Bullets and Numbering

23 **D.4.3 AES CBC-HMAC-SHA-1 test vector 3**

```

24 KEY 0000000000000000000000000000000000000000000000000000000000000000
25 HMK 0000000000000000000000000000000000000000000000000000000000000000
26 CIV 0000000000000000000000000000000000000000000000000000000000000000
27 HDR 0000000000000000000000000000000000000000000000000000000000000000
28 PTX 0000000000000000000000000000000000000000000000000000000000000000
29 CTX dc95c078a2408989ad48a21492842087
30 TAG d5adb529213cd69a9a3d69cf2d10b0b469d936fe

```

Formatted: Bullets and Numbering

31 **D.4.4 AES CBC-HMAC-SHA-1 test vector 4**

```

32 KEY fb7615b23d80891dd470980bc79584c8b2fb64ce60978f4d17fce45a49e830b7
33 HMK 1b07a0e93c1f4c3aaddf671dd2611ac2fe22d34c
34 CIV dbd1a3636024b7b402da7d6fe3fb056e
35 PTX a845348ec8c5b5f126f50e76fefdlb1e
36 CTX fd057a7f6d17bd747aced7b6fc948567
37 TAG 3bd64954b1b5b0a98ac3a6f95d2e5fe65b5377c0

```

Formatted: Bullets and Numbering

38 **D.4.5 AES CBC-HMAC-SHA-1 test vector 5**

```

39 KEY 404124234445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
40 HMK 202122232425262728292a2b2c2d2e2f30313233
41 CIV 101112131415161718191a1b1c1d1e1f
42 HDR 000102030405060708090a0b0c0d0e0f10111213
43 PTX 202122232425262728292a2b2c2d2e2f

```

```

1 CTX 7b626546c8d79cdeb66edef23b9b7d72
2 TAG c7932ddb8fc2212b56b1207e81019b556f4bb7d9
3 D.4.6 AES CBC-HMAC-SHA-1 test vector 6
4 KEY 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
5 HMK 202122232425262728292a2b2c2d2e2f30313233
6 CIV 101112131415161718191a1b1c1d1e1f
7 HDR 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
8 HDR 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
9 HDR 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
10 HDR 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
11 HDR 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f
12 HDR a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbef
13 HDR c0c1c2c3c4c5c6c7c8c9cacbccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeeff
14 HDR e0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2f3f4f5f6f7f8f9fafbfcfdfefff
15 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
16 CTX 7b626546c8d79cdeb66edef23b9b7d723d5f9d5bc2a411f1eb448442250eeca2
17 TAG 49dc3eacadcc028df2bf9a4598e3fec6624c8b38

```

Formatted: Bullets and Numbering

```

18 D.4.7 AES CBC-HMAC-SHA-1 test vector 7
19 KEY 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
20 HMK 202122232425262728292a2b2c2d2e2f30313233
21 CIV 101112131415161718191a1b1c1d1e1f
22 HDR 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
23 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
24 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
25 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
26 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
27 PTX 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f
28 PTX a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbef
29 PTX c0c1c2c3c4c5c6c7c8c9cacbccdcecf0d1d2d3d4d5d6d7d8d9dadbdcddeeff
30 PTX e0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2f3f4f5f6f7f8f9fafbfcfdfefff
31 CTX c9cb7b3859e1a550bcbf11b624022c56c3ad1479e5ce7034d7a03c13d8fb9502
32 CTX 6f7254c50ce4ebd743486d00e09ddd8e873a7e98984ad43f57088c510e911700
33 CTX 6acfe2fef69b4010f0f05a93af7d3a93a02085780fd5acb3a4eb870933077752
34 CTX 2f2c18e310ac0c0c3766bea3e97f71996336e4831f3b411fb2700ddbab565673
35 CTX 315bf4ab73c7e11abac4d0cfc228f1ac60dd10f85f9c2ade46a9af5each6a24a
36 CTX 43839b942e71ca4ce2080a809a04a849105da07efbbb2f60b9c376e0354e2a27
37 CTX da1eaa5c7adea77890cc25b6bd48229e17ce518040ceb46a04fc7b62444e77b5
38 CTX aaf3dbf60a660a2b68ec640622716b07758d99a0f598a73ed8bdae74fa3aae2f
39 TAG 2e08d65f81ff646ad05ab7aaf42903aa760e577a

```

Formatted: Bullets and Numbering

```

40 D.4.8 AES CBC-HMAC-SHA-1 test vector 8
41 KEY fb7615b23d80891dd470980bc79584c8b2fb64ce6097878d17fce45a49e830b7
42 HMK cc84a6cca8f97b8a5624071aec7d09e7cf5bdaff
43 CIV dbd1a3636024b7b402da7d6f54a67dc8
44 HDR 7bd859a2
45 PTX 90ae61cf7baebd4cade494c54a29ae70
46 CTX 6cd763ff6144ede649c486f9404a5307
47 TAG efc87d364ccab9d4bdc241185f1d847e2e16d8c4

```

Formatted: Bullets and Numbering



**D.5.4 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vector 4**

Formatted: Bullets and Numbering

1 KEY fb7615b23d80891dd470980bc79584c8b2fb64ce60978f4d17fce45a49e830b7  
 2 HMK 1b07a0e93c1f4c3aadff671dd2611ac2fe22d34c6b6d8630c30dd44f41d49fe5  
 3 HMK ad0a3dbdd0f13ca27e6523c5e4e2ab12884741a1af9b95f3cf6c0aec3b68ba40  
 4 CIV dbd1a3636024b7b402da7d6fe3fb056e  
 5 PTX a845348ec8c5b5f126f50e76fefdl1b1e  
 6 CTX fd057a7f6d17bd747aced7b6fc948567  
 7 HMAC-SHA-256  
 8 TAG 3e5530fb364c80696b1b2f69e8d0de064a3e07ad1a0b795f00fcdec1649cabcb  
 9 HMAC-SHA-512  
 10 TAG 444aec157e48e683626bf14d26c9bfd9515d5def34582c034f0c3311dd7d9753  
 11 TAG 591f3effe264b8cdfaf755177b8a020a47edb7331fef628523d708aefe09b0da  
 12

Formatted: Bullets and Numbering

**D.5.5 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vector 5**

13 KEY 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
 14 HMK 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
 15 HMK 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
 16 AAD 000102030405060708090a0b0c0d0e0f10111213  
 17 CIV 101112131415161718191a1b1c1d1e1f  
 18 PTX 202122232425262728292a2b2c2d2e2f  
 19 CTX 7b626546c8d79cdeb66edef23b9b7d72  
 20 HMAC-SHA-256  
 21 TAG 6b0fe0b40a41e32d2c61726a3d7834014a8ee07873ccfe0c23f3a9073b90b099  
 22 HMAC-SHA-512  
 23 TAG efac7480579348343d1e9af4fc6896968080439717c3b2c3e63013aa718261f0  
 24 TAG e3ee43c6fdb4372f020d64c9fee4bc7743cfd9262d3adf03aec4f8d99fd178e4  
 25

Formatted: Bullets and Numbering

**D.5.6 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vector 6**

26 KEY 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
 27 HMK 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
 28 HMK 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
 29 AAD 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f  
 30 AAD 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
 31 AAD 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
 32 AAD 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f  
 33 AAD 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f  
 34 AAD a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbebf  
 35 AAD c0c1c2c3c4c5c6c7c8c9cacbcccdcecfdd0d1d2d3d4d5d6d7d8d9daddbdcddeedf  
 36 AAD e0e1e2e3e4e5e6e7e8e9eaebeceedeefeff0f1f2f3f4f5f6f7f8f9fafbfcfdfefff  
 37 CIV 101112131415161718191a1b1c1d1e1f  
 38 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
 39 CTX 7b626546c8d79cdeb66edef23b9b7d723d5f9d5bc2a411f1eb448442250eeca2  
 40 HMAC-SHA-256  
 41 TAG c2ea45b50293d8f62d348ef23aec702268eb66bb3e2248eb9f71a5817709da2f  
 42 HMAC-SHA-512  
 43 TAG 8b2e672aacc78b6ff58c770fd0d6ed252201ebbae95dceec912c0cf3bf27171b  
 44 TAG 3627a6fefc3cc8b9f9e64b542b64c06ebb786f986cdc8296bac15111dbffa82f  
 45

Formatted: Bullets and Numbering

**D.5.7 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vector 7**

46 KEY 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
 47 HMK 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f  
 48 HMK 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f  
 49

```

1 AAD 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
2 CIV 101112131415161718191a1b1c1d1e1f
3 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
4 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
5 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
6 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
7 PTX 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f
8 PTX a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbdbf
9 PTX c0c1c2c3c4c5c6c7c8c9cacbcccdcecfcd0d1d2d3d4d5d6d7d8d9daddbdcddeedf
10 PTX e0e1e2e3e4e5e6e7e8e9eaebeceedeefef0f1f2f3f4f5f6f7f8f9fafbfcfdfeff
11 CTX c9cb7b3859e1a550bcbf11b624022c56c3ad1479e5ce7034d7a03c13d8fb9502
12 CTX 6f7254c50ce4ebd743486d00e09ddd8e873a7e98984ad43f57088c510e911700
13 CTX 6acfe2fef69b4010f0f05a93af7d3a93a02085780fd5acb3a4eb870933077752
14 CTX 2f2c18e310ac0c0c3766bea3e97f71996336e4831f3b411fb2700ddbab565673
15 CTX 315bf4ab73c7e11abac4d0cfc228f1ac60dd10f85f9c2ade46a9af5eacb6a24a
16 CTX 43839b942e71ca4ce2080a809a04a849105da07efbbb2f60b9c376e0354e2a27
17 CTX da1eaa5c7adea77890cc25b6bd48229e17ce518040ceb46a04fc7b62444e77b5
18 CTX aaf3dbf60a660a2b68ec640622716b07758d99a0f598a73ed8bdae74fa3aae2f
19 HMAC-SHA-256
20 TAG ebfe6f31be473ab22b649602a77f7408508dfa50cad109cbc97f2fe5f8bb8583
21 HMAC-SHA-512
22 TAG 7b326204521161942844c0970391344cdac71ce0440325b02203b537dd930799
23 TAG 0e158541dfc52cfcf69d3e8085658de4c98bc030273bad369fdf28aaad40e63c

```

Formatted: Bullets and Numbering

24 **D.5.8 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vector 8**

```

25 KEY fb7615b23d80891dd470980bc79584c8b2fb64ce6097878d17fce45a49e830b7
26 HMK cc84a6cca8f97b8a5624071aec7d09e7cf5bdaff239d467270f9716ba234d109
27 HMK ac60cf491d5105fc60fc5804c6474bc35cf9ead9123da80f649ca15a98a243d6
28 AAD 7bd859a2
29 CIV dbd1a3636024b7b402da7d6f54a67dc8
30 PTX 90ae61cf7baebd4cade494c54a29ae70
31 CTX 6cd763ff6144ede649c486f9404a5307
32 HMAC-SHA-256
33 TAG f130415f56372bcd17250339d82118ca347be4cfff9f69181757cf5e98b0a775
34 HMAC-SHA-512
35 TAG 1604c3afb72546c2f6a9135df46ae799fdae4d9f5a87fdffd552016c5e4ed98a
36 TAG 393b62822df55b076e3dc6f9668234919bbdcc99f2b40379754cc6ac30c97250

```

Formatted: Bullets and Numbering

37 **D.5.9 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vector 9**

```

38 KEY 0000000000000000000000000000000000000000000000000000000000000000
39 HMK 0000000000000000000000000000000000000000000000000000000000000000
40 HMK 0000000000000000000000000000000000000000000000000000000000000000
41 NON 0000000000000000000000000000000000000000000000000000000000000000
42 CIV dc95c078a2408989ad48a21492842087
43 PTX 0000000000000000000000000000000000000000000000000000000000000000
44 CTX 08c374848c228233c2b34f332bd2e9d3
45 HMAC-SHA-256
46 TAG 1f4dd7b6d7436b5b7d325c0c2411ed4fc02c101949eb8269e8166e8c6325e858
47 HMAC-SHA-512
48 TAG d8677480b0466345b3c32baa2c2b502fb3bfba01e759c4d1da04ca7c20dd9e55
49 TAG 00b3675d0e78e080125b68fd0c584ff3144b1e155a1136785ad723f3c69e23b5

```

Formatted: Bullets and Numbering

1 **D.6 XTS-AES-256-HMAC-512 Test Vectors**

2 **D.6.1 XTS-AES-256-HMAC-512 Vector 1**

```

3 Key1 2718281828459045235360287471352662497757247093699959574966967627
4 Key2 3141592653589793238462643383279502884197169399375105820974944592
5 HMK 0000000000000000000000000000000000000000000000000000000000000000
6 HMK 0000000000000000000000000000000000000000000000000000000000000000
7 DUS f000000000000000000000000000000000000000000000000000000000000000
8 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
9 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
10 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
11 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
12 PTX 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f
13 PTX a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbebf
14 PTX c0c1c2c3c4c5c6c7c8c9cacbcccdcecfdd0d1d2d3d4d5d6d7d8d9daddbdcddeedf
15 PTX e0e1e2e3e4e5e6e7e8e9eaebeceedeefeff0f1f2f3f4f5f6f7f8f9fafbfcfdfefff
16 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
17 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
18 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
19 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
20 PTX 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f
21 PTX a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbbebf
22 PTX c0c1c2c3c4c5c6c7c8c9cacbcccdcecfdd0d1d2d3d4d5d6d7d8d9daddbdcddeedf
23 PTX e0e1e2e3e4e5e6e7e8e9eaebeceedeefeff0f1f2f3f4f5f6f7f8f9fafbfcfdfefff
24 CTX 1c3b3a102f770386e4836c99e370cf9bea00803f5e482357a4ae12d414a3e63b
25 CTX 5d31e276f8fe4a8d66b317f9ac683f44680a86ac35adfc3345befecb4bb188fd
26 CTX 5776926c49a3095eb108fd1098baec70aaa66999a72a82f27d848b21d4a741b0
27 CTX c5cd4d5ffff9dac89aeba122961d03a757123e9870f8acf1000020887891429ca
28 CTX 2a3e7a7d7df7b10355165c8b9a6d0a7de8b062c4500dc4cd120c0f7418dae3d0
29 CTX b5781c34803fa75421c790dfe1de1834f280d7667b327f6c8cd7557e12ac3a0f
30 CTX 93ec05c52e0493ef31a12d3d9260f79a289d6a379bc70c50841473d1a8cc81ec
31 CTX 583e9645e07b8d9670655ba5bbcfec6dc3966380ad8fecb17b6ba02469a020a
32 CTX 84e18e8f84252070c13e9f1f289be54fbc481457778f616015e1327a02b140f1
33 CTX 505eb309326d68378f8374595c849d84f4c333ec4423885143cb47bd71c5edae
34 CTX 9be69a2ffecelbec9de244fbel5992b11b77c040f12bd8f6a975a44a0f90c29
35 CTX a9abc3d4d893927284c58754cce294529f8614dcd2aba991925fedc4ae74ffac
36 CTX 6e333b93eb4aff0479da9a410e4450e0dd7ae4c6e2910900575da401fc07059f
37 CTX 645e8b7e9bdfef33943054ff84011493c27b3429eaedb4ed5376441a77ed4385
38 CTX 1ad77f16f541dfd269d50d6a5f14fb0aab1cbb4c1550be97f7ab4066193c4caa
39 CTX 773dad38014bd2092fa755c824bb5e54c4f36ffda9fcea70b9c6e693e148c151
40 TAG 1c7105d3c1e8e235ffb013d5e8023729a35cdeacc16af1d7f5f0fec6c036b167
41 TAG 871649687c5692aaa0ada9773671939bbce2a3d15dcae43671aa6ca5f3a96a6f

```

Formatted: Bullets and Numbering

42 **D.6.2 XTS-AES-256-HMAC-512 Vector 2**

```

43 Key1 2718281828459045235360287471352662497757247093699959574966967627
44 Key2 3141592653589793238462643383279502884197169399375105820974944592
45 HMK 1b07a0e93c1f4c3aadff671dd2611ac2fe22d34c6b6d8630c30dd44f41d49fe5
46 HMK ad0a3dbdd0f13ca27e6523c5e4e2ab12884741a1af9b95f3cf6c0aec3b68ba40
47 DUS ffff000000000000000000000000000000000000000000000000000000000000
48 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
49 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
50 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
51 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f

```

Formatted: Portuguese (Brazil)



```

1 CTX 9298ebc699c0c8634715a320bb4f075d622e74c8c932004f25b41e361025b5a8
2 CTX 7815391f6108fc4afa6a05d9303c6ba68a128a55705d415985832fdeaae6c8e1
3 CTX 9110e84d1b1ff199a2692119edc96132658f09da7c623efcecc712537a3d94c0bf
4 CTX 5d7e352ec94ae5797fdb377dc1551150721adf15bd26a8efc2fcaad56881fa9e
5 CTX 62462c28f30ae1ceaca93c345cf243b73f542e2074a705bd2643bb9f7cc79bb6
6 CTX e7091ea6e232df0f9ad0d6cf502327876d82207abf2115cdacf6d5a48f6c1879
7 CTX a65b115f0f8b3cb3c59d15dd8c769bc014795a1837f3901b5845eb491adfefe0
8 CTX 97b1fa30a12fc1f65ba22905031539971a10f2f36c321bb51331cdefb39e3964
9 CTX c7ef079994f5b69b2edd83a71ef549971ee93f44eac3938fcdd61d01fa71799d
10 CTX a3a8091c4c48aa9ed263ff0749df95d44fef6a0bb578ec69456aa5408ae32c7a
11 CTX f08ad7ba8921287e3bbe31b767be06a0e705c864a769137df28292283ea81a2
12 CTX 480241b44d9921cdbc1bc28dc1fda114bd8e5217ac9d8ebafa720e9da4f9ace
13 CTX 231cc949e5b96fe76fffc21063fddc83a6b8679c00d35e09576a875305bed5f36
14 CTX ed242c8900dd1fa965bc950dfce09b132263a1eef52dd6888c309f5a7d712826
15 TAG a9fe02bbb70c062c93d958bc32936609a25alffa2dcd9f33aee88be73d943d4f
16 TAG dcbd459c0ecb0111c9c74cfcf2d5104f5f8262ae52444d6e744d8046f73ec7f2

```

Formatted: Bullets and Numbering

17 **D.6.4 XTS-AES-256-HMAC-512 Vector 4**

```

18 Key1 2718281828459045235360287471352662497757247093699959574966967627
19 Key2 3141592653589793238462643383279502884197169399375105820974944592
20 HMK e19c148c56a3aa6737471aaba909f06a17705e98bb8ee347e253c26cbf00cc5
21 HMK 3147ec26beb88413da0268d39bb4a707678277a0c927c10f565496d0fe3349d5
22 AAD e3e220f1f7f8ef20f9e820ece520e1e9ed20e6ea20e0ea20ecf4faf220f4e2f9
23 AAD 20e7e1e5f8e420f0e7eee3e420f9f6f6e420e0ea20e0d0a
24 DUS ffffffff00000000000000000000000000
25 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
26 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
27 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
28 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
29 PTX 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f
30 PTX a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbdbf
31 PTX c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9daddbdcddeedf
32 PTX e0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2f3f4f5f6f7f8f9fafbfcfdfefff
33 PTX 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
34 PTX 202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f
35 PTX 404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f
36 PTX 606162636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
37 PTX 808182838485868788898a8b8c8d8e8f909192939495969798999a9b9c9d9e9f
38 PTX a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbdbf
39 PTX c0c1c2c3c4c5c6c7c8c9cacbcccdcecf0d1d2d3d4d5d6d7d8d9daddbdcddeedf
40 PTX e0e1e2e3e4e5e6e7e8e9eaebeceedeef0f1f2f3f4f5f6f7f8f9fafbfcfdfefff
41 CTX bf53d2dade78e822a4d949a9bc6766b01b06a8ef70d26748c6a7fc36d80ae4c5
42 CTX 520f7c4ab0ac8544424fa405162fef5a6b7f229498063618d39f0003cb5fb8d1
43 CTX c86b643497da1ff945c8d3bedeca4f479702a7a735f043ddb1d6aaade3c4a0ac
44 CTX 7ca7f3fa5279bef56f82cd7a2f38672e824814e10700300a055e1630b8f1cb0e
45 CTX 919f5e942010a416e2bf48cb46993d3cb6a51c19bacf864785a00bc2ecff15d3
46 CTX 50875b246ed53e68be6f55bd7e05cfc2b2ed6432198a6444b6d8c247fab941f5
47 CTX 69768b5c429366f1d3f00f0345b96123d56204c01c63b22ce78baf116e525ed9
48 CTX 0fdea39fa469494d3866c31e05f295ff21fea8d4e6e13d67e47ce722e9698a1c
49 CTX 1048d68ebcde76b86fcf976eab8aa9790268b7068e017a8b9b749409514f1053
50 CTX 027fd16c3786ealbac5f15cb79711ee2abe82f5cf8b13ae73030ef5b9e4457e7
51 CTX 5d1304f988d62dd6fcb94ed38ba831da4b7634971b6cd8ec325d9c61c00f1df
52 CTX 73627ed3745a5e8489f3a95c69639c32cd6e1d537a85f75cc844726e8a72fc00
53 CTX 77ad22000f1d5078f6b866318c668f1ad03d5a5fced5219f2eabbd0aa5c0f460
54 CTX d183f04404a0d6f469558e81fab24a167905ab4c7878502ad3e38fdba62a4155
55 CTX 6cec37325759533ce8f25f367c87bb5578d667ae93f9e2fd99bcbcb5f2fba88c

```

Formatted: Portuguese (Brazil)



1 CTX e8a8eff1ae4020cfa39936b66827b23f371b92200be90251e6d73c5f86de5fd4  
2 CTX a950781933d79a28272b782a2ec313efdfcc0628f43d744c2dc2ff3dcb66999b  
3 CTX 50c7ca895b0c64791eeaa5f29499fb1c026f84ce5b5c72ba1083cddb5ce45434  
4 CTX 631665c333b60b11593fb253c5179a2c8db813782a004856a1653011e93fb6d8  
5 CTX 76c18366dd8683f53412c0c180f9c848592d593f8609ca736317d356e13e2bff  
6 CTX 3a9f59cd9aeb19cd482593d8c46128bb32423b37a9adfb482b99453fbe25a41b  
7 CTX f6feb4aa0bef5ed24bf73c762978025482c13115e4015aac992e5613a3b5c2f6  
8 CTX 85b84795cb6e9b2656d8c88157e52c42f978d8634c43d06fea928f2822e465aa  
9 CTX 6576e9bf419384506cc3ce3c54ac1a6f67dc66f3b30191e698380bc999b05abc  
10 CTX e19dc0c6dcc2dd001ec535ba18deb2df1a101023108318c75dc98611a09dc48a  
11 CTX 0acdec676fabdf222f07e026f059b672b56e5cbc8e1d21bbd867dd9272120546  
12 CTX 81d70ea737134cdfce93b6f82ae22423274e58a0821cc5502e2d0ab4585e94de  
13 CTX 6975be5e0b4efce51cd3e70c25a1fbbbd609d273ad5b0d59631c531f6a0a57b9  
14 TAG 0664e417f6740411cc10c55d6a8c6f43b7ad21f95f0f6b4751b6049990d13136  
15 TAG 8fef3f1b42e172fbbec6b8133fdbcbb8dccb3fed9c345818dc0ae11ace07e0c43  
16  
17

1. Overview .....	1
1.1 Scope .....	1
1.2 Purpose .....	1
1.3 Description of clauses and annexes .....	1
2. Normative references.....	1
3. Definitions, acronyms, and abbreviations .....	2
3.1 Keywords.....	2
3.2 Definitions .....	3
3.3 Acronyms and abbreviations .....	5
3.4 Mathematical conventions .....	6
4. General concepts .....	6
4.1 Introduction .....	6
4.2 Overview of roles .....	7
4.3 Host .....	9
4.4 Key manager.....	12
4.5 Cryptographic unit.....	13
4.6 Storage medium.....	17
4.7 Documentation.....	<b>Error! Bookmark not defined.</b>
5. Cryptographic modes.....	17
5.1 Cryptographic mode selection .....	17
5.2 Encryption .....	18
5.3 Decryption .....	19
5.4 Parameter limits.....	20
5.5 Counter mode with CBC-MAC (CCM).....	21
5.6 Galois/counter mode (GCM) .....	21
5.7 Cipher Block Chaining with HMAC-SHA (CBC-HMAC-SHA).....	22
5.8 Tweakable block-cipher with HMAC (XTS-AES-256-HMAC-SHA-512).....	23
6. Cryptographic key management and initialization vector requirements.....	24
6.1 Random bit generator .....	24
6.2 Cryptographic key entry and export .....	25
6.3 Handling the user key and cipher key.....	25
6.4 Cryptographic key wrapping on the storage medium .....	26
6.5 Initialization Vector (IV) requirements .....	27
6.6 Creating unique IVs within a self-contained group .....	29
Annex A (informative) Bibliography .....	30
Annex B (informative) Security concerns .....	32
B.1 Threat model .....	32
B.2 Lack of security when using passwords as cryptographic keys .....	32

B.3 Replay attacks .....	32
B.4 Passing plaintext to the host before checking the MAC .....	33
B.5 Checking for integrity of a cryptographic key .....	33
B.6 Avoiding collisions of initialization vectors .....	34
B.7 Examples of IV collision avoidance strategies .....	34
B.8 How many records to encrypt with one key? .....	36
Annex C (informative) Documentation Requirements summary .....	38
Annex D (informative) Test vectors .....	32
D.1 General.....	32
D.2 CCM test vectors .....	33
D.3 GCM test vectors .....	34
D.4 CBC-HMAC-SHA-1 Test Vectors .....	37
D.5 CBC-HMAC-SHA-256 and CBC-HMAC-SHA-512 Test Vectors.....	39
D.6 XTS-AES-256-HMAC-512 Test Vectors .....	42

*Examples of storage media:*

- Magnetic tape cartridge
- Hard disk
- Flash memory
- Holographic or optical storage

## Cryptographic modes

### Cryptographic mode selection

The cryptographic unit shall support at least one of the cryptographic modes shown in Table 1:

—Cryptographic modes

	Name	Description	Ref.
	CCM-128-AES-256	Counter with 128-bit cipher block chaining MAC	5.1
	GCM-128-AES-256	Galois/Counter Mode with 128-bit MAC	5.2
	CBC-AES-256-HMAC-SHA-1	Cipher block chaining with 160-bit HMAC	5.3
	CBC-AES-256-HMAC-SHA-256	Cipher block chaining with 256-bit HMAC	5.3
	CBC-AES-256-HMAC-SHA-512	Cipher block chaining with 512-bit HMAC	5.3
	XTS-AES-256-HMAC-SHA-512	Tweakable block cipher with 512-bit HMAC	5.4

The cryptographic unit receives host records from the host as a basic unit of data for encryption. When performing encryption, the cryptographic unit shall format the host records into plaintext records that are input into the encryption routine. Documentation shall describe the mapping between host records and plaintext records.

To minimize buffering requirements and latency, the cryptographic unit may define a maximum size for the plaintext records that is smaller than the maximum host record size allowed by the cryptographic unit. If a particular host record is larger than the maximum plaintext record length, then the cryptographic unit may split the host record into multiple plaintext records with optional padding or reformatting. When splitting a host record into multiple plaintext records, the cryptographic unit shall include sufficient information within the AAD, IV, or plaintext record to allow the decryption routine to unambiguously reconstruct the original host record. To help fulfill this requirement, the cryptographic unit should use ordering verification to detect tampering or reordering of the encrypted records (see 4.6.3).

The cryptographic unit may apply padding to the host records to form the plaintext records. Regardless of whether the host record is padded, there shall be sufficient information within either the plaintext record or AAD to unambiguously determine the length of the host record. If a cryptographic unit supports padding of the plaintext, then documentation shall describe the padding algorithm and the method for determining the length of the host record, using only information protected by the MAC.

The host may configure the cryptographic unit to write a particular host record to the storage medium either with encryption or without encryption. In this case, the cryptographic unit may mix both encrypted records and plaintext records on the storage medium. The cryptographic unit may write additional information without encryption to the storage medium, assuming that such information does not reveal cryptographic keys or plaintext that was intended to be encrypted.

**NOTE**—See 6.3 for a description of the relationship between the user key, which is provided by the key manager, and the cipher key, which is used by the cryptographic unit for encryption and decryption.

## Key-transform functions

### Overview

The following subclauses contain key-transform functions that the cryptographic unit may use. Each function transforms the user key, provided by the key manager, into a cipher key that the cryptographic unit uses for encryption or decryption. This process reduces the probability of cipher key and IV collisions among cryptographic units that are given the same user key.

When using a key-transform function, the cryptographic unit shall perform the following steps before encrypting data with the cipher key:

Generate a new nonce, using either an RBG if using KTF1 (see 6.4.2) or globally unique information if using KTF2 (see 6.4.3), KTF3 (see 6.4.4), or KTF4 (see 6.4.5).

Pass the nonce and user key into the key-transform function and set the cipher key to its output.

Either store on the storage medium sufficient information to allow reconstruction of the nonce, or use key-wrapping as described in 6.5 to archive the resulting cipher key. When using key wrapping in this case, the key manager provides both a KEK and user key to the cryptographic unit.

When using a key-transform function in decryption mode, the cryptographic unit shall reconstruct the nonce using information from the storage medium and use the nonce and user key as inputs into the key-transform function, and then set the cipher key to the output of the key-transform function.

If during the previous encryption the cryptographic unit used both a key-transform function and key-wrapping (see c) above), then using a key-transform function on the subsequent decryption is not necessary. In this case, the cryptographic unit needs only to unwrap the cipher key before decryption.

The cryptographic unit should use a random nonce, but may use a globally unique nonce, as an input into the key-transform function. A newly generated random nonce shall contain at least 64-bits of random data and should contain 256-bits of random data. When using a key-transform function, a new nonce shall be generated for each new encryption session.

If a cryptographic unit supports key-transform, then documentation shall specify the format of the nonce used in the key-transform function.

### Key-transform function #1 (KTF1)

This subclause defines a key-transform function that uses the SHA-384 (i.e. SHA-2 with 384-bit digest) hashing function (see NIST FIPS 180-2) and follows the format provided by NIST SP 800-56A in the subclause entitled "Concatenation Key Derivation Function". This key-transform function produces a 256-bit cipher key, and therefore shall not be used in conjunction with any mode other than the CCM or GCM mode.

Support for KTF1 is optional.

NOTE—Compliance with this key transform function does not assure compliance with NIST SP 800-56A. The context of this KTF does not precisely fit that of SP 800-56A.

Table 4 shows the format of the data provided as input to the SHA-384 hashing function. All integers within this field are stored in big endian order.

—Format of data provided as input to the SHA-384 hashing function

Byte row	Byte offset															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Counter				User key (0-11)											
16	User key (12-27)															
32	User key (28-31)				Algorithm ID (0-11)											
48	Algorithm ID (12-15)				Host ID				Device ID				Reserved		Nonce size	
64	Nonce															
80																
96																

The Counter field corresponds to the *Counter* input in the Concatenation Key Derivation Function (see SP 800-56A). The Counter field shall contain the value 00000001<sub>16</sub>.

The User key field contains the user key provided by the key manager, and is given as *X* in SP 800-56A.

The Algorithm ID field contains a byte-string that uniquely identifies this transform. For this transform, the Algorithm ID field shall equal the ASCII string "IEEE1619.1 KTF1", with a single trailing NULL (00<sub>16</sub>) byte.

The Host ID field contains the identifier number for the host, as required by SP 800-56A. For this standard, the Host ID field shall contain the value 00000001<sub>16</sub>.

The Device ID field contains the identifier number for the cryptographic unit, as required by SP 800-56A. For this standard, the Device ID field shall contain the value 00000002<sub>16</sub>.

The Reserved field shall equal 0000<sub>16</sub>.

The Nonce size field contains the number of bytes within the Nonce field. The size of the Nonce field shall be between 8 and 47 bytes, inclusive, and should be 32 bytes.

When using the KTF1 function, the Nonce field shall contain a random nonce.

NOTE—Since the cipher key in CCM and GCM modes is 32-bytes long, a random nonce longer than 32-bytes provides diminishing amounts of extra assurance.

Equation (1) shows how to create the cipher key from the user key and nonce.

$$KeyConf || CipherKey = \text{SHA-384}(\text{Information from Table 4}) \quad (\oplus)$$

where:

- KeyConf* is a reserved 16-byte value
- CipherKey* is a 32-byte value used by the cryptographic unit for encryption

Table 5 shows an example of how each byte is populated with the constant data.

**—Example of constant data fields for key-transform function #1**

Byte row	Byte offset																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	00	00	00	01	User key												
16	User key (cont.)																
32	User key (cont.)				'I'	'E'	'E'	'E'	'I'	'6'	'1'	'9'	'.'	'1'	' '	'K'	
48	'T'	'F'	'1'	'\0'	00	00	00	01	00	00	00	02	00	00	Nonce size		
64	Nonce																
80																	
96																	

**Key-transform function #2 (KTF2)**

This subclause describes key-transform function #2 (KTF2). Support of KTF2 is optional.

KTF2 is identical to KTF1 (see 6.4.2) with the following modifications:

The Algorithm ID field shall equal the ASCII string "IEEE1619.1 KTF2", with a single trailing NULL (00<sub>16</sub>) byte.

The first bytes of the Nonce field shall contain an NAA (see ISO/IEC 14165-252). The remaining bytes within the Nonce field in conjunction with the IV (see 6.6.3.3) shall be globally unique within the scope of the NAA for each encrypted record.

**Key-transform function #3 (KTF3)**

This subclause describes key-transform function #3 (KTF3). Support of KTF3 is optional.

KTF3 is identical to KTF1 (see 6.4.2) with the following modifications:

The Algorithm ID field shall equal the ASCII string "IEEE1619.1 KTF3", with a single trailing NULL (00<sub>16</sub>) byte.

The first 8 bytes of the Nonce field shall contain an EUI-64 [need reference]. The remaining bytes within the Nonce field in conjunction with the IV (see 6.6.3.3) shall be globally unique within the scope of the EUI-64 for each encrypted record.

**Key-transform function #4 (KTF4)**

This subclause describes key-transform function #4 (KTF4). Support of KTF4 is optional.

KTF4 is identical to KTF1 (see 6.4.2) with the following modifications:

The Algorithm ID field shall equal the ASCII string "IEEE1619.1 KTF4", with a single trailing NULL (00<sub>16</sub>) byte.

The first 32 bytes of the Nonce field shall contain the output of a SHA-256 hashing function applied to an IQN [need reference]. The remaining bytes within the Nonce field in conjunction with the IV (see 6.6.3.3) shall be globally unique within the scope of the IQN for each encrypted record.

Page 30: [5] Deleted	Matthew Ball	6/1/2007 10:21:00 AM
If a cryptographic unit supports key-transform, then documentation shall specify the format of the nonce used in the key-transform function.		6.4.1
Page 31: [6] Deleted	Matthew Ball	5/17/2007 6:41:00 PM

(informative)

## P1619.1 Standard encryption interchange format

### Overview

This annex describes a format that could facilitate interchange of encrypted records. Supporting this format is optional.

All fields within this format are aligned to 4-byte boundaries. If a particular sub-field is not aligned to 32-bits, then the formatter shall pad to 4 bytes using zeros. All integers are in big-endian format; the most significant bits reside in the first byte (see 3.4).

The key manager provides a user key as input to the cryptographic unit. The user key may be in any format.

### Format description

#### Format table

Table D.1 shows the elements that constitute the encryption interchange format. Each element in this table is in order, as it would appear within a data file or packet.

The length fields in this format are not necessarily authenticated, so care should be taken when using them. The documentation of the cryptographic unit describes how to authenticate the length fields.

**Table D.1—Elements of the encryption interchange format**

Name	Size (bytes)	Description	Ref.
ID	8	Identifies this format	D.2.2
Mode	4	Identifies the encryption mode	D.2.3
Plaintext format	4	Describes the format of the plaintext	D.2.4
UCKT type	4	Identifies the method of deriving the cipher key from the user key	D.2.5
UCKT data size	4	The number of bytes within the UCKT data field	D.2.6
UCKT data	0+	(optional) The supplemental data sent to the key-transform function	D.2.7
APD type	4	Description of the APD field	D.2.8
APD size	4	(optional) The number of bytes within the APD field	D.2.9
APD	0+	(optional) Additional plaintext data	D.2.10
IV type	4	Describes the IV	D.2.11
IV size	4	Number of bytes within the IV (initialization vector)	D.2.12
IV	12+	The initialization vector	D.2.13
AAD size	8	Number of bytes within the AAD (additionally authenticated data)	D.2.14
AAD	0+	The additionally authenticated data with size given by AAD Size	D.2.15
Ciphertext size	8	Number of bytes within the ciphertext	D.2.16

Name	Size (bytes)	Description	Ref.
Ciphertext	0+	The encrypted plaintext data	D.2.17
MAC	16	The message authentication code	D.2.18

The following subclauses describe each element within Table D.1. All strings described in these subclauses contain ASCII characters.

## ID

The ID (identifier) field describes the general contents of this file. For this version, the ID field contains the ASCII string "P1619.1 ", with a trailing space.

## Mode

The Mode field describes the encryption mode used to encrypt the data. The possible values are as follows:

- "CCM " - The CCM algorithm as described in 5.6.
- "GCM " - The GCM algorithm as described in 5.5
- "CHS1" – The CBC-HMAC-SHA-1 algorithm as described in 5.7.
- "CHS2" – The CBC-HMAC-SHA-256 algorithm as described in 5.7.
- "CHS5" – The CBC-HMAC-SHA-512 algorithm as described in 5.7.
- "XTS5" – The XTS-AES-SHA-512 algorithm as described in 5.8.

## Plaintext format

The Plaintext format field contains a general description of the format of the plaintext after a successful decryption operation. The possible values are as follows:

- "PTNS" - The formatting of the plaintext is unknown or unspecified (plaintext not specified).
- ".zip" - The plaintext is compressed according to the PKZIP file format.
- ".bz2" - The plaintext is compressed according to the BZIP2 file format.
- "SLDC" - The plaintext is compressed according to ECMA 321 (see [B5]).
- All other codes starting with a '.' (period) - The plaintext format given by the three letter file suffix, according to the three characters following the '.'.
- All codes including only numerals and all-capital characters - Reserved.
- All other codes are available for vendor-specific purposes.

## UCKT type

The UCKT (User to Cipher Key Translation) type field contains a description of the method used to derive the cipher key from the user key. The possible values are as follows:

- "NOKT" - No key-transform function was used. The cipher key is the user key. In this case, the UCKT data field is empty (NO key translation).
- "KTF1" - The key-transform function as described in 6.4.2. In this case, the UCKT data field contains a random nonce input into KTF1.

"KTF2"- The key-transform function as described in 6.4.3. In this case, the UCKT data field contains a globally unique nonce input into KTF2.

"NAKW" – The UCKT data field contains the result of using NIST AES key-wrap (see [B18]) to wrap the cipher key using the user key. In this case, the UCKT data size is 40 bytes, and the user key and cipher key are both 32-bytes long.

"OAEP" – The UCKT data field contains the cipher key encrypted using RSA PKCS #1 v2.1 RSAES-OAEP (see [B24]) using distinguished encoding rules (DER). In this case, the user key is an RSA public key that encrypts the cipher key.

"ECES" – The UCKT data field contains the cipher key encrypted using the ECIES encryption algorithm (see [B10]). The user key is an elliptic curve public key that encrypts the cipher key.

All codes that contain only numerals and all-capital characters - Reserved.

All other codes are available for vendor-specific purposes.

### **UCKT data size**

The UCKT data size field contains a 4-byte integer that indicates the number of bytes within the UCKT data field.

### **UCKT data**

The UCKT data field contains a byte-stream containing data according to the UCKT type.

### **APD type**

The APD (additional plaintext data) type field describes the contents of any additional plaintext. The possible values for this field are as follows:

"NOAP" - There is no APD provided. In this case, there is no APD Size field.

"UKAD" - The APD field contains UKAD (unauthenticated key associated data) according to INCITS T10/SSC-3.

All codes that contain only numerals and all-capital characters - Reserved.

All other codes are available for vendor-specific purposes.

### **APD size**

The APD size field contains a 4-byte integer that indicates the number of bytes within the APD field. If the APD type field contains the string "NOAP", then the APD size field is not present and there is no APD field.

### **APD**

The APD (additional plaintext data) field contains optional plaintext data according to the format indicated by the APD type field. The APD size field indicates the number of bytes within the APD field. If the APD type field contains the string "NOAP", then the APD field is not present.

### **IV type**

The IV type field contains a 4-byte ASCII string that describes the method used to generate the IV. The possible values for this field are as follows:

"NONC" - The IV is a unique nonce, generated according to 6.6.3.3.

"RAND" - The IV is derived entirely from an RBG, according to 6.6.2.

All codes that consist of capital letters and digits are reserved.

All other codes are available for vendor-specific purposes.

### **IV size**

The IV size field contains a 4-byte integer that indicates the number of bytes within the IV field.

### **IV**

The IV (initialization vector) field contains the initialization vector used to encrypt the plaintext. The IV size field indicates the number of bytes within the IV field.

### **AAD size**

The AAD size field contains an 8-byte integer that indicates the number of bytes within the AAD field.

### **AAD**

The AAD (additionally authenticated data) field contains the AAD input into the encryption algorithm (see 5.2.2). The AAD size field gives the size in bytes of the AAD field.

### **Ciphertext size**

The ciphertext size field contains an 8-byte integer that indicates the number of bytes within the ciphertext field.

### **Ciphertext**

The ciphertext field contains the ciphertext that results from an encryption operation (see 5.2.3). The size of the ciphertext field is given by the ciphertext size field.

### **MAC**

The MAC (message authentication code) field contains the resulting MAC from an encryption operation (see 5.2.3).