



IEEE Standard for Wide-Block Encryption for Shared Storage Media

STANDARDS

IEEE Computer Society

Developed by the
Cybersecurity and Privacy Standards Committee

IEEE Std 1619.2™-2021
(Revision of IEEE Std 1619.2-2010)



IEEE Standard for Wide-Block Encryption for Shared Storage Media

Developed by the

Cybersecurity and Privacy Standards Committee
of the
IEEE Computer Society

Approved 9 May 2021

IEEE SA Standards Board

THIS IS A REDLINE DOCUMENT PRODUCED BY TECHSTREET, AN ASME COMPANY.

This document is intended to provide users with an indication of changes from one edition to the next. It includes a full-text version of the new document, plus an indication of changes from the previous version.

Redlines are designed to save time and improve efficiencies by using the latest software technology to find and highlight document changes. More professionals are using valuable new technologies like redlines, to help improve outcomes in a fast paced global business world.

Because it may not be technically possible to capture all changes accurately, it is recommended that users consult previous editions as appropriate. In all cases, only the current base version of this publication is to be considered the official document.

This Redline is produced solely by Techstreet. IEEE has not prepared this Redline and is not responsible for its accuracy.

Redline Processing Notes:

1. ~~Red Text~~ - Red strikethrough text denotes deletions.
2. Blue Text - Blue underlined text denotes modifications and additions.

Abstract: EME2-AES and XCB-AES wide-block encryption with associated data (EAD) modes of the NIST AES block cipher, providing usage guidelines and test vectors, are described. A wide-block encryption algorithm behaves as a single block cipher with a large plaintext input and ciphertext output, but uses a narrow block cipher (in this case Advanced Encryption Standard [AES]) internally. These encryption modes are oriented toward random access storage devices that do not provide authentication, but need to reduce the granularity of a potential attack.

Keywords: data-at-rest security, encryption, encryption with associated data (EAD), encrypt-mix-encrypt-v2 mode of operation (EME2), extended codebook mode of operation (XCB), IEEE 1619.2™, security, storage

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2021 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 18 June 2021. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-1-5044-7513-6 STS24666
Print: ISBN 978-1-5044-7514-3 STDPD24666

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA, and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or completeness of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to results and workmanlike effort. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and Standards Coordinating Committees are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#). An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These

include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#). For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#). Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the [IEEE SA Patent Policy](#).

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Participants

At the time this IEEE standard was completed, the Security in Storage Working Group had the following membership:

Jim Hatfield, *Chair*
Eric Hibbard, *Vice Chair*

Richard Austin
Thomas Bowen
Tim Chevalier
Mike Danielson
Anthony Duran
Sherif Fares
Monty Forehand

John Geldman
Chris Hillier
Glen Jaquette
Michael Jaslowski
Jamie Pocas
Thomas Rivera

Yoni Shternhell
Curtis Stevens
Robert Strong
Paul Suhler
Gary Sutphin
Curtis Trimble
Burt Wagner

The following members of the individual Standards Association balloting group voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Johann Amsenga
Richard Austin
Juan Carreon
Roger Cummings
Andrew Fieldsend
Dan Friedman
John Geldman

Jim Hatfield
Eric Hibbard
Werner Hoelzl
Raj Jain
Glen Jaquette
Piotr Karocki

Johnny Marques
Rajesh Murthy
Pablo Rivas Perea
Paul Suhler
Dmitri Varsanofiev
Oren Yuen
Janusz Zalewski

When the IEEE SA Standards Board approved this standard on 9 May 2021, it had the following membership:

Gary Hoffman, *Chair*
Jon Walter Rosdahl, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Edward A. Addy
Doug Edwards
Ramy Ahmed Fathy
J. Travis Griffith
Thomas Koshy
Joseph L. Koepfinger*
David J. Law

Howard Li
Daozhuang Lin
Kevin Lu
Daleep C. Mohla
Chenhui Niu
Damir Novosel
Annette Reilly
Dorothy Stanley

Mehmet Ulema
Lei Wang
F. Keith Waters
Karl Weber
Sha Wei
Howard Wolfman
Daidi Zhong

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1619.2™-2021, IEEE Standard for Wide-Block Encryption for Shared Storage Media.

The purpose of this standard, similar to IEEE Std 1619™-2018 [B2], is to describe a method of encryption for data stored in logical block-based devices, where the threat model includes possible access to stored data by the adversary.¹ As in IEEE Std 1619-2018, this standard specifies length-preserving encryption algorithms to be applied to the plaintext logical block before storing it on the storage media.

This standard improves on IEEE Std 1619-2018 by defining *wide-block* encryption algorithms. This means that they act on the whole logical block at once, and each bit on the input plaintext influences every bit of the output ciphertext (and vice versa for decryption). In particular, this standard specifies the EME2-AES and the XCB-AES wide-block encryption algorithms.

Wide-block encryption better hides plaintext statistics and provides better protection than the narrow-block encryption, defined in IEEE Std 1619-2018, against attacks that involve traffic analysis and/or manipulations of ciphertext on the raw storage media.

¹ The numbers in brackets correspond to those of the bibliography in Annex A.

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
1.3 Word usage	1
2. Normative references	1
3. Definitions, acronyms, and abbreviations	2
3.1 Definitions	2
3.2 Acronyms and abbreviations	2
4. Mathematical conventions	2
5. Wide-block encryption algorithms	3
5.1 Encryption with associated data	3
5.2 EME2-AES algorithm	5
5.3 XCB-AES algorithm	10
6. Compliance	16
Annex A (informative) Bibliography	17
Annex B (informative) Implementation guidance	18
B.1 Security considerations	18
B.2 Performance characteristics of EME2-AES and XCB-AES	18
B.3 Application to logical block-level disk encryption	19
Annex C (informative) Test vectors	20
C.1 Encoding	20
C.2 EME2-AES test vectors	20
C.3 XCB-AES test vectors	67

IEEE Standard for Wide-Block Encryption for Shared Storage Media

~~IMPORTANT NOTICE: This standard is not intended to ensure safety, security, health, or environmental protection. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.~~

~~This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Documents." They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.~~

1. Overview

1.1 Scope

This standard specifies an architecture for encryption of data in random access storage devices, oriented toward applications that benefit from wide encryption-block sizes of 512 bytes and above.

1.2 Purpose

This standard specifies an architecture for media security and enabling components. Wide encryption blocks are well suited to environments where the attacker has repeated access to cryptographic communication or ciphertext, or is able to perform traffic analysis of data access patterns. The standard is oriented toward fixed-size encryption blocks without data expansion, but anticipates an optional data expansion mode to resist attacks involving data tampering.

1.3 Word usage

The word *shall* indicates ~~shall: A keyword indicating a~~ mandatory ~~requirements~~ ~~requirement~~ strictly to be followed in order to conform to ~~the~~ ~~this~~ standard and from which no deviation is permitted (*shall* equals *is required to*).^{1, 2}

The word *should* indicates ~~should: A keyword indicating~~ that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; ~~or that (in the negative form) a certain course of action is deprecated but not prohibited~~ (*should* equals *is recommended* ~~that to~~).

The word *may* is used to indicate ~~may: A keyword indicating~~ a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

NIST Federal Information Processing Standard 197 (FIPS 197), Advanced Encryption Standard (AES). November 2001.^{3 4}

¹ The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.

² The use of *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

^{3 4} NIST publications are available from the National Institute of Standards and Technology, ~~NIST Public Inquiries, NIST, 100 Bureau Drive, Stop 3460, Gaithersburg, MD, 20899 3460, USA~~ (<https://www.nist.gov/>).

NIST Special Publication 800-38A (NIST SP 800-38A), Recommendation for Block Cipher Modes of Operation—Methods and Techniques.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary* ~~Online: Glossary of Terms & Definitions~~ should be consulted ~~referenced~~ for terms not defined in this clause.^{4 2}

associated data: Data that is associated with the plaintext, but that does not need to be encrypted. The associated data input should characterize the plaintext. In other algorithms for encryption of data at rest, the associated data is often referred to as *tweak value*.

encryption with associated data (EAD): A cryptographic algorithm that consists of an encryption procedure used to encrypt a plaintext and the associated data and a secret key, and a decryption procedure used to decrypt a ciphertext with the same associated data and secret key.

3.2 Keywords

~~can:~~ A keyword indicating a capability (~~can equals is able to~~).

~~required:~~ ~~See: shall.~~

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

~~shall not:~~ A phrase indicating an absolute prohibition of this standard.

^{4 2} ~~The~~ *IEEE Standards Dictionary* Online is available at: <http://dictionary.ieee.org>. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page ~~Glossary of Terms & Definitions is available at <http://shop.ieee.org/>.~~

3.2 Acronyms and abbreviations

AES	Advanced Encryption Standard (see NIST FIPS 197 ^{5 3})
CTR	counter (as in AES counter mode of operation)
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode of authenticated encryption
GF	Galois Field (see Menezes, Van Oorschot, and Vanstone ~~et al.~~ [B4] ^{6 4})
LBA	Logical Block Address

4. Mathematical conventions

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

Binary numbers are represented by a string of one or more binary digits, followed by the subscript 2. For example, the decimal number 26 is represented as 00011010₂ in binary.

Hexadecimal numbers are represented by a string of one or more hexadecimal digits, prefixed by the string “0x.” For example, the decimal number 135 is represented as 0x87 in hexadecimal. The number of bits in a bit string X is denoted as #X.

The procedures used in this standard are as follows:

^{5 3} Information on references can be found in Clause 2.

^{6 4} The numbers in brackets correspond to those of the bibliography in Annex A.

5. Wide-block encryption algorithms

5.1 Encryption with associated data

5.1.1 Data units and associated data

This standard specifies length-preserving encryption with associated data (EAD) algorithms that are suitable for the encryption of data at rest. An EAD algorithm consists of an encryption procedure and a decryption procedure. The encryption procedure accepts three inputs as follows:

- Secret key
- Plaintext
- Associated data

It returns a single ciphertext value. Each of these inputs is a byte string.

The security of these procedures depends on the secret key consisting of random or pseudo-random data that is unpredictable to the adversary. Each EAD algorithm accepts a key of a fixed length, but different algorithms may have keys of different lengths.

The plaintext input contains the data to be encrypted. Plaintext is divided into data units that, within a particular key scope, may have different lengths. An EAD algorithm defines the range of admissible plaintext lengths. A compliant implementation should keep the size of the data units the same within a key scope. If an implementation allows variable-length data units within a key scope, then the ~~implementation's~~ [implementation's](#) design should be such that it is able to unambiguously determine the length of each data unit without using data that is potentially under the control of an adversary. A compliant implementation shall use distinct associated data for distinct data units.

The associated data input contains data that is associated with the plaintext, but that does not need to be encrypted. The choice of data for this input is described in more detail as follows. Within a particular key scope, associated data units may have different lengths.

The ciphertext returned by the encryption procedure is the same length as the plaintext.

The decryption procedure accepts three inputs as follows:

- Secret key
- Ciphertext
- Associated data value

It returns a single plaintext value. The decryption procedure is the reverse of the encryption procedure; more specifically, if the encryption of the plaintext P with the key K and the associated data A results in the ciphertext C , then the decryption of C with the key K and the associated data A results in the plaintext P .

A conforming implementation shall include in the associated data only information that is available, in plaintext form, at the time of encryption and the time of decryption.

The associated data input shall uniquely identify the plaintext. This is because whenever the same plaintext is encrypted two different times using the same key but with distinct associated data values, the result is two distinct ciphertext values. Thus the use of distinct associated data values hides the equality of the plaintexts from an attacker.

5.1.2 Using EAD to protect a string of data blocks

An EAD may be used to protect a string of data blocks, such as those in a data-storage disk. In this application, the associated data input to the encryption and decryption procedure should contain the logical index of the block on which the procedure is acting. When this information is included in the associated data, cases in which two distinct data blocks contain identical plaintext values are hidden from an adversary. Figure 1 shows an example of how an EAD performs encryption and decryption.

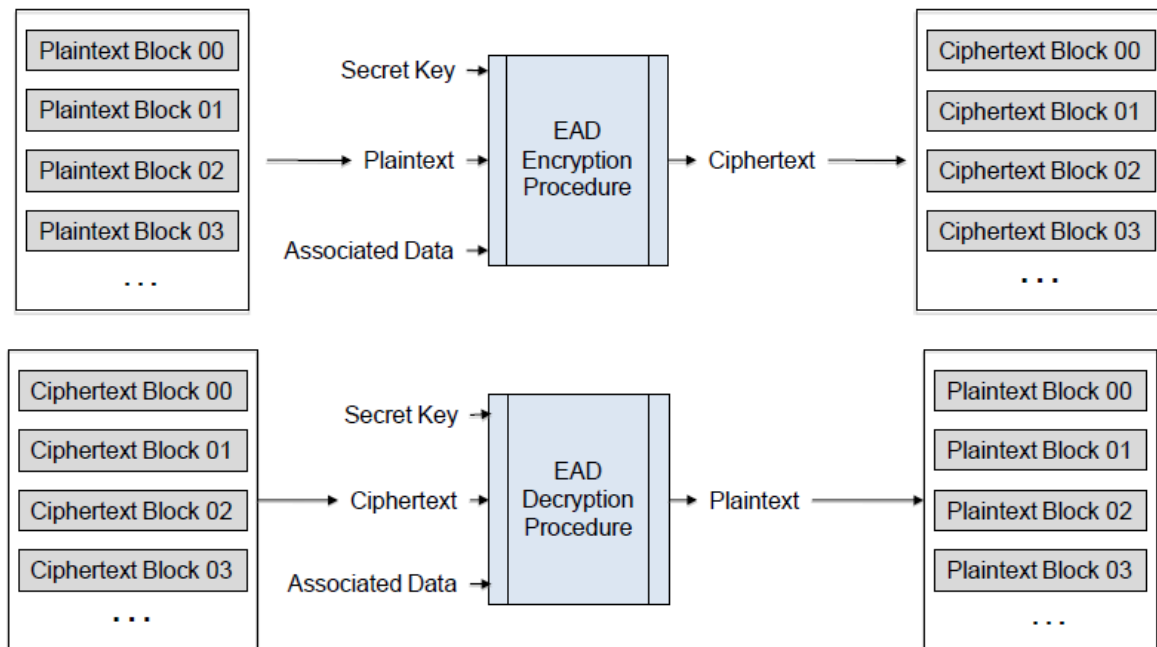


Figure 1—EAD encryption and decryption of a block device

If multiple disks are being protected with a single secret key, then the associated data input should contain both the logical index of the block and an additional distinguishing parameter that is unique to each of the disks. When this information is included in the associated data, cases in which two distinct data blocks on different disks contain identical plaintext values are hidden from an adversary.

5.2 EME2-AES algorithm

5.2.1 The mult-by-alpha procedure

The encryption and decryption procedures described in the following subclauses use a mult-by-alpha(X) procedure that multiplies a 16-byte value X by a primitive element α in the field $\text{GF}(2^{128})$. The input value is first converted into a byte string $X[i]$, $i = 0, 1, \dots, 15$, where $X[0]$ is the first byte of the byte string.

The multiplication by alpha is defined in Table 1 or by a mathematically equivalent procedure.

Table 1—The mult-by-alpha(X) procedure

```
// compute mult-by-alpha(X), where X is a 16-byte value
//
// X[i] denotes the ith byte of X, and indexing starts at 0
//
// i is an integer used in loops
//
// Y[i] denotes the ith byte of the output Y, and indexing starts at 0

1. for i = 0 to 15 do
2.   Y[i] = 2*X[i] mod 256
3.   if (i>0 and X[i-1]>127) then Y[i]=Y[i]+1
4. end-for
5. if (X[15] > 127) then Y[0] = Y[0] ⊕ 0x87
6. return Y
```

Conceptually, the procedure is a left shift of each byte by one bit with carry propagating from one byte to the next. Also, if the 15th (last) byte shift results in a carry, a special value (hexadecimal 0x87) is xor'ed into the first byte. This value is derived from the modulus of the Galois Field (polynomial $x^{128} + x^7 + x^2 + x + 1$).

5.2.2 EME2-AES encryption

Equation (1) describes the EME2-AES encryption procedure:

$$C = \text{EME2-AES-Enc}(Key, T, P) \quad (1)$$

where

Key is the 48- or 64-byte EME2-AES key

T is the associated data, of arbitrary byte length (zero or more bytes)

P is the plaintext, of length 16 bytes or more

C is the ciphertext resulting from the operation, of the same byte-length as *P*

The input to the EME2-AES encryption procedure is parsed as follows:

- The key is partitioned into three fields, $Key = KAD \mid KECB \mid KAES$, with KAD (the associated data key) consisting of the first 16 bytes, $KECB$ (the ECB pass key) consisting of the following 16 bytes, and $KAES$ (the AES encryption/decryption key) consisting of the remaining 16 or 32 bytes.
- If not empty, the associated data T is partitioned into a sequence of blocks $T = T_1 \mid T_2 \mid \dots \mid T_r$, where each of the blocks $T_1, T_2, \dots, T_{r-1}$ is of length exactly 16 bytes, and T_r is of length between 1 and 16 bytes.
- The plaintext P is partitioned into a sequence of blocks $P = P_1 \mid P_2 \mid \dots \mid P_m$, where each of the blocks $P_1, P_2, \dots, P_{m-1}$ is of length exactly 16 bytes, and P_m is of length between 1 and 16 bytes.

As part of the ciphertext computation, the associated data T are processed by the sequence of steps in Table 2 or by a mathematically equivalent procedure.

Table 2—The function H for processing the associated data T

```

// Function H( $K_{AES}$ ,  $K_{AD}$ ,  $T = (T_1 \mid \dots \mid T_{r-1} \mid T_r)$ ) :
1. if bytelength( $T$ ) == 0 then
2.    $T\_star = AES-ENC(K_{AES}, K_{AD})$ 
3. else
4.    $K_{AD} = \text{mult-by-alpha}(K_{AD})$ 
5.   for  $j = 1$  to  $r-1$ 
6.      $TT_j = AES-ENC(K_{AES}, K_{AD} \oplus T_j) \oplus K_{AD}$ 
7.      $K_{AD} = \text{mult-by-alpha}(K_{AD})$ 
8.   next
9.   if bytelength( $T_r$ ) < 16 then
10.     $T_r = T_r \mid 0x80 \mid 0x00 \mid \dots$  // pad  $T_r$  to 16 bytes, 0x80 followed by 0's
11.     $K_{AD} = \text{mult-by-alpha}(K_{AD})$ 
12.   endif
13.   $TT_r = AES-ENC(K_{AES}, K_{AD} \oplus T_r) \oplus K_{AD}$ 
14.   $T\_star = TT_1 \oplus TT_2 \oplus \dots \oplus TT_r$ 
15. endif
16. return  $T\_star$  // return the 16 byte value  $T\_star$ 

```

A conforming implementation shall compute the ciphertext by executing the sequence of steps in Table 3 or a mathematically equivalent procedure.

Table 3—The EME2-AES encryption procedure

```

// EME2-AES-Enc( $K_{AES}$ ,  $K_{ECB}$ ,  $K_{AD}$ ,  $T$ ,  $P = (P_1 \mid \dots \mid P_{m-1} \mid P_m)$ ):
1.  $T\_star = H(K_{AES}, K_{AD}, T)$  // Process the associated data
2. if bytelength( $P_m$ ) == 16 then
3.   lastFull = m
4. else
5.   lastFull = m-1
6.    $PPP_m = P_m \mid 0x80 \mid 0x00 \mid \dots$  // Pad  $P_m$  to 16 bytes, 0x80 followed by 0's
7. endif

// First ECB pass
8.  $L = K_{ECB}$ 
9. for j = 1 to lastFull
10.   $PPP_j = \text{AES-Enc}(K_{AES}, L \oplus P_j)$ 
11.   $L = \text{mult-by-alpha}(L)$ 
12.next

// Intermediate mixing
13.  $MP = PPP_1 \oplus PPP_2 \oplus \dots \oplus PPP_m \oplus T\_star$ 
14. if bytelength( $P_m$ ) < 16 then
15.   $MM = \text{AES-Enc}(K_{AES}, MP)$ 
16.   $MC = MC_1 = \text{AES-Enc}(K_{AES}, MM)$ 
17. else
18.   $MC = MC_1 = \text{AES-Enc}(K_{AES}, MP)$ 
19. endif
20.  $M = M_1 = MP \oplus MC$ 
21. for j = 2 to lastFull
22.  if (j-1 mod 128 > 0) then // use the current mask M
23.     $M = \text{mult-by-alpha}(M)$ 
24.     $CCC_j = PPP_j \oplus M$ 
25.  else // calculate a new mask M
26.     $MP = PPP_j \oplus M_1$ 
27.     $MC = \text{AES-Enc}(K_{AES}, MP)$ 
28.     $M = MP \oplus MC$ 
29.     $CCC_j = MC \oplus M_1$ 
30.  endif
31.next
32. if lastFull < m then
33.   $C_m = P_m \oplus (MM \text{ truncated to bytelength}(P_m) \text{ bytes})$ 
34.   $CCC_m = C_m \mid 0x80 \mid 0x00 \mid \dots$  // Pad  $C_m$  to 16 bytes, 0x80 followed by 0's
35. endif
36.  $CCC_1 = MC_1 \oplus CCC_2 \oplus \dots \oplus CCC_m \oplus T\_star$ 

// Second ECB Pass
37.  $L = K_{ECB}$ 
38. for j = 1 to lastFull
39.   $C_j = \text{AES-Enc}(K_{AES}, CCC_j) \oplus L$ 
40.   $L = \text{mult-by-alpha}(L)$ 
41.next
42. return  $C = (C_1 \mid \dots \mid C_{m-1} \mid C_m)$ 

```

An illustration of the EME2-AES encryption procedure (for plaintext of 130 full blocks and one partial block) is provided in Figure 2.

The input to the EME2-AES decryption procedure is parsed as follows:

- The key is partitioned into three fields, $Key = KAD \mid KECB \mid KAES$, with KAD (the associated data key) consisting of the first 16 bytes, $KECB$ (the ECB pass key) consisting of the following 16 bytes, and $KAES$ (the AES encryption/decryption key) consisting of the remaining 16 or 32 bytes.
- If not empty, the associated data is partitioned into a sequence of blocks $T = T_1 \mid T_2 \mid \dots \mid T_r$, where each of the blocks $T_1, T_2, \dots, T_{r-1}$ is of length exactly 16 bytes, and T_r is of length between 1 and 16 bytes.
- The ciphertext C is partitioned into a sequence of blocks $C = C_1 \mid C_2 \mid \dots \mid C_m$, where each of the blocks $C_1, C_2, \dots, C_{m-1}$ is of length exactly 16 bytes, and C_m is of length between 1 and 16 bytes.

A conforming implementation shall compute the plaintext by executing the sequence of steps in Table 4 or a mathematically equivalent procedure.

NOTE—The only difference between the encryption and decryption procedures is that all the AES-Enc calls in Table 3 are replaced in Table 4 by calls to AES-Dec. The function H is defined in Table

2.7.5

2.7.5 Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

Table 4—The EME2-AES decryption procedure

```

// EME2-AES-Dec( $K_{AES}$ ,  $K_{ECB}$ ,  $K_{AD}$ ,  $T$ ,  $C = (C_1 \mid \dots \mid C_{m-1} \mid C_m)$ ):
1.  $T\_star = H(K_{AES}, K_{AD}, T)$  // Process the associated data
2. if  $bytelength(C_m) == 16$  then
3.    $lastFull = m$ 
4. else
5.    $lastFull = m-1$ 
6.    $CCC_m = C_m \mid 0x80 \mid 0x00 \mid \dots$  // Pad  $C_m$  to 16 bytes, 0x80 followed by 0's
7. endif

// First ECB pass
8.  $L = K_{ECB}$ 
9. for  $j = 1$  to  $lastFull$ 
10.   $CCC_j = AES-Enc(K_{AES}, L \oplus C_j)$ 
11.   $L = mult-by-alpha(L)$ 
12. next

// Intermediate mixing
13.  $MC = CCC_1 \oplus CCC_2 \oplus \dots \oplus CCC_m \oplus T\_star$ 
14. if  $bytelength(C_m) < 16$  then
15.   $MM = AES-Dec(K_{AES}, MC)$ 
16.   $MP = MP_1 = AES-Dec(K_{AES}, MM)$ 
17. else
18.   $MP = MP_1 = AES-Dec(K_{AES}, MC)$ 
19. endif
20.  $M = M_1 = MP \oplus MC$ 
21. for  $j = 2$  to  $lastFull$ 
22.  if  $(j-1 \bmod 128 > 0)$  then // use the current mask M
23.     $M = mult-by-alpha(M)$ 
24.     $PPP_j = CCC_j \oplus M$ 
25.  else // calculate a new mask M
26.     $MC = CCC_j \oplus M_1$ 
27.     $MP = AES-Dec(K_{AES}, MC)$ 
28.     $M = MP \oplus MC$ 
29.     $PPP_j = MP \oplus M_1$ 
30.  endif
31. next
32. if  $lastFull < m$  then
33.   $P_m = C_m \oplus (MM \text{ truncated to } bytelength(C_m) \text{ bytes})$ 
34.   $PPP_m = P_m \mid 0x80 \mid 0x00 \mid \dots$  // Pad  $P_m$  to 16 bytes, 0x80 followed by 0's
35. endif
36.  $PPP_1 = MP_1 \oplus PPP_2 \oplus \dots \oplus PPP_m \oplus T\_star$ 

// Second ECB Pass
37.  $L = K_{ECB}$ 
38. for  $j = 1$  to  $lastFull$ 
39.   $P_j = AES-Dec(K_{AES}, PPP_j) \oplus L$ 
40.   $L = mult-by-alpha(L)$ 
41. next
42. return  $P = (P_1 \dots P_{m-1} P_m)$ 

```

5.3 XCB-AES algorithm

5.3.1 Overview of XCB-AES

The XCB-AES algorithm use the AES block cipher encryption procedures AES-Enc and AES-Dec, as well as the hash function h and the pseudorandom function c . The variables H , K_e , K_d , and K_c are derived from K , essentially by running the AES-Enc encryption procedure in CTR mode (see NIST SP 800-38A). A conforming implementation may store these values between computation of encryption and decryption procedures, in order to trade off storage and computational load.

5.3.2 The function c

Let k be the size of the key fed to the AES procedure (either 16 or 32 bytes).

The function c : $\{0, 255\}^k \times \{0, 255\}^{16} \rightarrow \{0, 255\}^l$, where the output length l (in bytes) is bounded by $0 \leq l \leq 2^{36}$, generates an arbitrary-length output by running the AES-Enc procedure in CTR mode, using its 16-byte input as the initial counter value. Its definition is as follows:

$$c(K, W, l) = \text{AES-Enc}(K, W) \parallel \text{AES-Enc}(K, \text{incr}(W)) \parallel \dots \parallel \text{msbt}(\text{AES-Enc}(K, \text{incr}^{n-1}(W)))$$

where the output length l is indicated as an explicit parameter for clarity; $n = \lceil l / 16 \rceil$ is the number of 16-byte blocks in the output, and $t = l \bmod 16$ is the number of bytes in the trailing block.

Here the function incr : $\{0, 255\}^{16} \rightarrow \{0, 255\}^{16}$ is the increment operation that is used to generate successive counter values. This function treats the rightmost 4 bytes of its argument as a non-negative integer with the least significant byte on the right, and increments this value modulo 2^{32} . More formally,

$$\text{incr}(X) = X[0:95] \parallel (X[96:127] + 1 \bmod 2^{32})$$

where bit strings are implicitly converted into integers.

5.3.3 Multiplication in $GF(2^{128})$

The multiplication operation of two 16-byte values $X, Y \in GF(2^{128})$ is mathematically equivalent to an operation on bit vectors. The result $Z = X \cdot Y$ is also an element of $GF(2^{128})$. The input values are first converted into a byte string $X[i], i = 0, 1, \dots, 15$, where the most significant byte is $X[0]$, and the least significant byte is $X[15]$.

The multiplication in $GF(2^{128})$ operation is defined in Table 5 or by a mathematically equivalent procedure.

Table 5—The multiplication in GF(2¹²⁸) procedure

```
// mult-GF(X, Y)

// compute Z = X * Y, where X, Y, Z are elements of GF(2^128) using a
// byte based algorithm
//
// here A[i] denotes the ith byte of A, and indexing starts at 0
//
// i and j are integers used in loops
// mask, b, and msb are unsigned integers

/* initialize z to the all-zero element */
7. for i = 0 to 15
8.   z[i] = 0
9. next

10. for i=0 to 15 /* loop over bytes of y */
11.   mask = 128
12.   while mask > 0 /* loop over bits in byte */
13.     /* if masked bit is set, add in terms from x */
14.     b = y[i] & mask
15.     if b != 0
16.       for j=0 to 15
17.         set z[j] to z[j] ^ x[j]
18.       next
19.     endif
20.     mask = mask / 2;
21.     /* now execute LFSR shift on x */
22.     set msb to x[15] & 0x01
23.     for j=15 down to 1
24.       set b to x[j-1] & 0x01
25.       set x[j] to (x[j] / 2) + b * 128
26.     next
27.     set x[0] to x[0] / 2
28.     if msb == 1
29.       set x[0] to x[0] ^ 0xe1
30.     endif
31.   endwhile
32. next
33. return z
```

NOTE—In Table 5, the multiplication operation uses the special element $R = 111000012 \mid 0^{120}$ (0xe1 | 0x00 | ... | 0x00).

5.3.4 The procedures h1 and h2

The procedures h1 and h2 are defined in terms of the underlying hash function h as follows:

$$h_1(H, Z, B) = h(H, 0^{128} \mid Z, B \mid 0^{\text{padlen1}(\#B)})$$

$$h_2(H, Z, B) = h(H, Z \parallel 0^{128}, B \parallel 0^{\text{padlen2}(\#B)} \parallel \text{bitlength}(Z \parallel 0^{128}) \parallel \text{bitlength}(B))$$

The procedure $\text{padlen2}(x)$ returns the integer between 0 and 127, inclusive, that when added to x is a multiple of 128. The procedure $\text{padlen1}(x)$ returns $\text{padlen2}(x) + 128$. Equation (3) and Equation (4) express these statements mathematically:

$$\text{padlen2}(x) = 128 * \text{ceil}(x/128) - x \quad (3)$$

$$\text{padlen1}(x) = 128 * (1 + \text{ceil}(x/128)) - x = 128 + \text{padlen2}(x) \quad (4)$$

where

$\text{ceil}(x)$ is the smallest integer that is larger than or equal to x .

The procedure $h : \{0, 1\}^{128} \times \{0, 1\}^a \times \{0, 1\}^c \rightarrow \{0, 1\}^{128}$ is defined by $h(H, A, C) = X_{m+n+1}$, where a and c are within the interval $[128, 2^{39}]$, m is the number of 16-byte blocks in A , n is the number of 16-byte blocks in C , and the variables $X_i \in \{0, 1\}^{128}$ for $i = 0, \dots, m + n + 1$ are defined as follows:

$$\begin{aligned} X_i &= 0 & \text{for } i = 0 \\ X_i &= (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \dots, m-1 \\ X_i &= (X_{m-1} \oplus A_m) \cdot H & \text{for } i = m \\ X_i &= (X_{i-1} \oplus C_{i-m}) \cdot H & \text{for } i = m+1, \dots, m+n-1 \\ X_i &= (X_{m+n-1} \oplus C_n) \cdot H & \text{for } i = m+n \\ X_i &= (X_{m+n} \oplus (\text{bitlength}(A) \parallel \text{bitlength}(C))) \cdot H & \text{for } i = m+n+1 \end{aligned}$$

Let A_i denote the 16-byte substring $A[128*(i-1): 128*i - 1]$, and let C_i denote $C[128*(i-1): 128*i - 1]$. In other words, A_i and C_i are the i th blocks of A and C , respectively, if those bit strings are decomposed into 16-byte blocks (the last A_i and C_i blocks are padded with 0s, if shorter than 16 bytes).

NOTE—This procedure is identical to the universal hash function that is used as a component of the Galois/Counter Mode (GCM) of Operation (see NIST SP 800-38D [B5]). It is equivalent to the procedure used in Step 5 of Algorithm 4 of that specification, but please note that it is different than GHASH as defined in that document.

5.3.5 XCB-AES encryption

The XCB-AES encryption procedure for an m -bit block P is modeled with Equation (5):

$$CT \leftarrow \text{XCB-AES-Enc}(K, P, Z) \quad (5)$$

where

K is the 16- or 32-byte XCB-AES key

P is the plaintext of m bits where $m \in [128, 2^{32}]$, and m is a multiple of 8

Z is the value of the associated data, of arbitrary byte length (zero or more bytes)

CT is the ciphertext resulting from the operation, of the same length as P

A conforming implementation shall compute the ciphertext by executing the following steps or a mathematically equivalent procedure (see Figure 3):

$$H \leftarrow \text{AES-Enc}(K, 0^{128})$$

$$K_e \leftarrow \text{msbk}(\text{AES-Enc}(K, 0^{125}|0012) \parallel \text{AES-Enc}(K, 0^{125}|0102))$$

$$K_d \leftarrow \text{msbk}(\text{AES-Enc}(K, 0^{125}|0112) \parallel \text{AES-Enc}(K, 0^{125}|1002))$$

$$K_c \leftarrow \text{msbk}(\text{AES-Enc}(K, 0^{125}|1012) \parallel \text{AES-Enc}(K, 0^{125}|1102))$$

$$A \leftarrow P[m - 128 : m - 1]$$

$$B \leftarrow P[0 : m - 127]$$

$$C \leftarrow \text{AES-Enc}(K_e, A)$$

$$D \leftarrow C \oplus h_1(H, Z, B)$$

$$E \leftarrow B \oplus c(K_c, D, \#B)$$

$$F \leftarrow D \oplus h_2(H, Z, E)$$

$$G \leftarrow \text{AES-Dec}(K_d, F)$$

$$CT \leftarrow E \parallel G$$

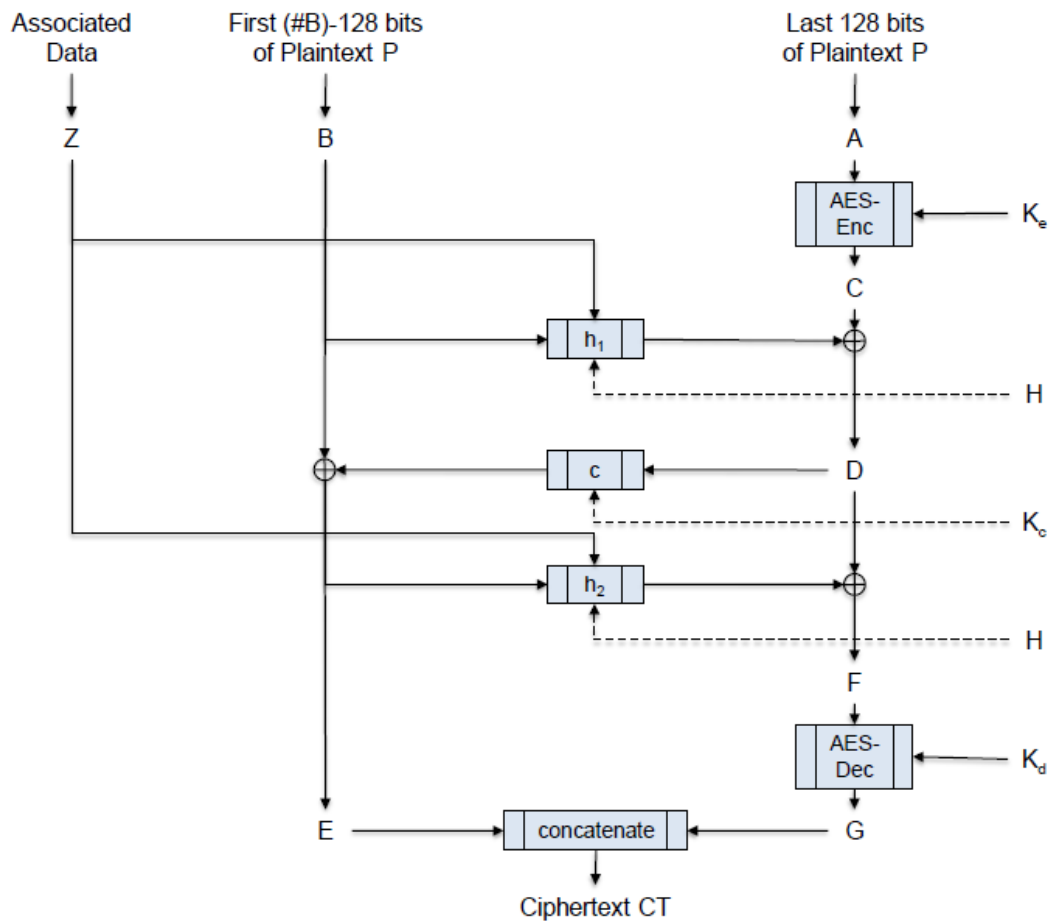


Figure 3 — An illustration of XCB-AES encryption