

Use of the Advanced Encryption Standard (AES) Encryption  
Algorithm in Cryptographic Message Syntax (CMS)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This document specifies the conventions for using the Advanced Encryption Standard (AES) algorithm for encryption with the Cryptographic Message Syntax (CMS).

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [MUSTSHOULD].

1. Overview

This document specifies the conventions for using Advanced Encryption Standard (AES) content encryption algorithm with the Cryptographic Message Syntax [CMS] enveloped-data and encrypted-data content types.

CMS values are generated using ASN.1 [X.208-88], using the Basic Encoding Rules (BER) [X.209-88] and the Distinguished Encoding Rules (DER) [X.509-88].

### 1.1. AES

The Advanced Encryption Standard (AES) [AES] was developed to replace DES [DES]. The AES Federal Information Processing Standard (FIPS) Publication specifies a cryptographic algorithm for use by U.S. Government organizations. However, the AES will also be widely used by organizations, institutions, and individuals outside of the U.S. Government.

Two researchers who developed and submitted the Rijndael algorithm for consideration are both cryptographers from Belgium: Dr. Joan Daemen of Proton World International and Dr. Vincent Rijmen, a postdoctoral researcher in the Electrical Engineering Department of Katholieke Universiteit Leuven.

The National Institute of Standards and technology (NIST) selected the Rijndael algorithm for AES because it offers a combination of security, performance, efficiency, ease of implementation, and flexibility. Specifically, Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. The very low memory requirements of the Rijndael algorithm make it very well suited for restricted-space environments, in which it also demonstrates excellent performance. The Rijndael algorithm operations are among the easiest to defend against power and timing attacks. Additionally, it appears that some defense can be provided against such attacks without significantly impacting the algorithm's performance. Finally, the algorithm's internal round structure appears to have good potential to benefit from instruction-level parallelism.

The AES specifies three key sizes: 128, 192 and 256 bits.

## 2. Enveloped-data Conventions

The CMS enveloped-data content type consists of encrypted content and wrapped content-encryption keys for one or more recipients. The AES algorithm is used to encrypt the content.

Compliant software **MUST** meet the requirements for constructing an enveloped-data content type stated in [CMS] Section 6, "Enveloped-data Content Type".

The AES content-encryption key **MUST** be randomly generated for each instance of an enveloped-data content type. The content-encryption key (CEK) is used to encrypt the content.

AES can be used with the enveloped-data content type using any of the following key management techniques defined in [CMS] Section 6.

1) Key Transport: The AES CEK is uniquely wrapped for each recipient using the recipient's public RSA key and other values. Section 2.2 provides additional details.

2) Key Agreement: The AES CEK is uniquely wrapped for each recipient using a pairwise symmetric key-encryption key (KEK) generated using an originator's randomly generated private key (ES-DH [DH]) or previously generated private key (SS-DH [DH]), the recipient's public DH key, and other values. Section 2.3 provides additional details.

3) Previously Distributed Symmetric KEK: The AES CEK is wrapped using a previously distributed symmetric KEK (such as a Mail List Key). The methods by which the symmetric KEK is generated and distributed are beyond the scope of this document. Section 2.4 provides additional details.

4) Password Encryption: The AES CEK is wrapped using a KEK derived from a password or other shared secret. Section 2.5 provides additional details.

Documents defining the use of the Other Recipient Info structure will need to define the proper use for the AES algorithm if desired.

## 2.1. EnvelopedData Fields

The enveloped-data content type is ASN.1 encoded using the EnvelopedData syntax. The fields of the EnvelopedData syntax MUST be populated as follows:

The EnvelopedData version is determined based on a number of factors.

See [CMS] section 6.1 for the algorithm to determine this value.

The EnvelopedData recipientInfos CHOICE is dependent on the key management technique used. Section 2.2, 2.3, 2.4 and 2.5 provide additional information.

The EnvelopedData encryptedContentInfo contentEncryptionAlgorithm field MUST specify a symmetric encryption algorithm. Implementations MUST support content encryption with AES, but implementations MAY support other algorithms as well.

The EnvelopedData unprotectedAttrs MAY be present.

## 2.2. KeyTransRecipientInfo Fields

The enveloped-data content type is ASN.1 encoded using the EnvelopedData syntax. The fields of the EnvelopedData syntax MUST be populated as follows:

The KeyTransRecipientInfo version MUST be either 0 or 2. If the RecipientIdentifier is the CHOICE issuerAndSerialNumber, then the version MUST be 0. If the RecipientIdentifier is subjectKeyIdentifier, then the version MUST be 2.

The KeyTransRecipientInfo RecipientIdentifier provides two alternatives for specifying the recipient's certificate, and thereby the recipient's public key. The recipient's certificate MUST contain a RSA public key. The CEK is encrypted with the recipient's RSA public key. The issuerAndSerialNumber alternative identifies the recipient's certificate by the issuer's distinguished name and the certificate serial number; the subjectKeyIdentifier identifies the recipient's certificate by the X.509 subjectKeyIdentifier extension value.

The KeyTransRecipientInfo keyEncryptionAlgorithm field specifies the key transport algorithm (i.e., RSAES-OAEP [RSA-OAEP]), and the associated parameters used to encrypt the CEK for the recipient.

The KeyTransRecipientInfo encryptedKey is the result of encrypting the CEK with the recipient's RSA public key.

## 2.3. KeyAgreeRecipientInfo Fields

This section describes the conventions for using ES-DH or SS-DH and AES with the CMS enveloped-data content type to support key agreement. When key agreement is used, then the RecipientInfo keyAgreeRecipientInfo CHOICE MUST be used.

The KeyAgreeRecipient version MUST be 3.

The EnvelopedData originatorInfo field MUST be the originatorKey alternative. The originatorKey algorithm fields MUST contain the dh-public-number object identifier with absent parameters. The originatorKey publicKey MUST contain the originator's ephemeral public key.

The EnvelopedData ukm MAY be present.

The EnvelopedData keyEncryptionAlgorithm MUST be the id-alg-ESDH algorithm identifier [CMSALG].

## 2.3.1. ES-DH/AES Key Derivation

Generation of the AES KEK to be used with the AES-key wrap algorithm is done using the method described in [DH].

## 2.3.1.1. Example 1

ZZ is the 20 bytes 00 01 02 03 04 05 06 07 08 09  
0a 0b 0c 0d 0e 0f 10 11 12 13

The key wrap algorithm is AES-128 wrap, so we need 128 bits (16 bytes) of keying material.

No partyAInfo is used.

Consequently, the input to SHA-1 is:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 ; ZZ
30 1b
  30 11
    06 09 60 86 48 01 65 03 04 01 05          ; AES-128 wrap OID
    04 04
      00 00 00 01          ; Counter
a2 06
  04 04
    00 00 00 80          ; key length
```

And the output is the 32 bytes:

d6 d6 b0 94 c1 02 7a 7d e6 e3 11 72 94 a3 53 64 49 08 50 f9

Consequently,

K= d6 d6 b0 94 c1 02 7a 7d e6 e3 11 72 94 a3 53 64

## 2.3.1.2. Example 2

ZZ is the 20 bytes 00 01 02 03 04 05 06 07 08 09  
0a 0b 0c 0d 0e 0f 10 11 12 13

The key wrap algorithm is AES-256 key wrap, so we need 256 bits (32 bytes) of keying material.

The partyAInfo used is the 64 bytes

01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01  
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01  
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01  
01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01

Consequently, the input to first invocation of SHA-1 is:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 ; ZZ
30 5f
  30 11
    06 09 60 86 48 01 65 03 04 01 2d          ; AES-256 wrap OID
    04 04
      00 00 00 01          ; Counter
a0 42
  04 40
    01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01 ; partyAInfo
      01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
      01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
      01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
a2 06
  04 04
    00 00 01 00          ; key length
```

And the output is the 20 bytes:

88 90 58 5C 4E 28 1A 5C 11 67 CA A5 30 BE D5 9B 32 30 D8 93

The input to second invocation of SHA-1 is:

```

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 ; ZZ
30 5f
  30 11
    06 09 60 86 48 01 65 03 04 01 2d          ; AES-256 wrap OID
    04 04
      00 00 00 02          ; Counter
a0 42
  04 40
    01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01 ; partyAInfo

    01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
    01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
    01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 01
a2 06
  04 04
    00 00 01 00          ; key length

```

And the output is the 20 bytes:

```
CB A8 F9 22 BD 1B 56 A0 71 C9 6F 90 36 C6 04 2C AA 20 94 37
```

Consequently,

```

K = 88 90 58 5C 4E 28 1A 5C 11 67 CA A5 30 BE D5 9B
    32 30 D8 93 CB A8 F9 22 BD 1B 56 A0

```

### 2.3.2. AES CEK Wrap Process

The AES key wrap algorithm encrypts one AES key in another AES key. The algorithm produces an output 64-bits longer than the input AES CEK, the additional bits are a checksum. The algorithm uses 6\*n AES encryption/decryption operations where n is number of 64-bit blocks in the AES CEK. Full details of the AES key wrap algorithm are available at [AES-WRAP].

NIST has assigned the following OIDs to define the AES key wrap algorithm.

```

id-aes128-wrap OBJECT IDENTIFIER ::= { aes 5 }
id-aes192-wrap OBJECT IDENTIFIER ::= { aes 25 }
id-aes256-wrap OBJECT IDENTIFIER ::= { aes 45 }

```

In all cases the parameters field MUST be absent. The OID gives the KEK key size, but does not make any statements as to the size of the wrapped AES CEK. Implementations MAY use different KEK and CEK

sizes. Implements MUST support the CEK and the KEK having the same length. If different lengths are supported, the KEK MUST be of equal or greater length than the CEK.

#### 2.4. KEKRecipientInfo Fields

This section describes the conventions for using AES with the CMS enveloped-data content type to support previously distributed symmetric KEKs. When a previously distributed symmetric KEK is used to wrap the AES CEK, then the RecipientInfo KEKRecipientInfo CHOICE MUST be used. The methods used to generate and distribute the symmetric KEK are beyond the scope of this document. One possible method of distributing keys is documented in [SYMKEYDIST].

The KEKRecipientInfo fields MUST be populated as specified in [CMS] Section 6.2.3, KEKRecipientInfo Type.

The KEKRecipientInfo keyEncryptionAlgorithm algorithm field MUST be one of the OIDs defined in section 2.3.2 indicating that the AES wrap function is used to wrap the AES CEK. The KEKRecipientInfo keyEncryptionAlgorithm parameters field MUST be absent.

The KEKRecipientInfo encryptedKey field MUST include the AES CEK wrapped using the previously distributed symmetric KEK as input to the AES wrap function.

#### 2.5. PasswordRecipientInfo Fields

This section describes the conventions for using AES with the CMS enveloped-data content type to support password-based key management.

When a password derived KEK is used to wrap the AES CEK, then the RecipientInfo PasswordRecipientInfo CHOICE MUST be used.

The keyEncryptionAlgorithm algorithm field MUST be one of the OIDs defined in section 2.3.2 indicating the AES wrap function is used to wrap the AES CEK. The keyEncryptionAlgorithm parameters field MUST be absent.

The encryptedKey field MUST be the result of the AES key wrap algorithm applied to the AES CEK value.

### 3. Encrypted-data Conventions

The CMS encrypted-data content type consists of encrypted content with implicit key management. The AES algorithm is used to encrypt the content.

Compliant software MUST meet the requirements for constructing an enveloped-data content type stated in [CMS] Section 8, "Encrypted-data Content Type".

The encrypted-data content type is ASN.1 encoded using the EncryptedData syntax. The fields of the EncryptedData syntax MUST be populated as follows:

The EncryptedData version is determined based on a number of factors.

See [CMS] section 9.1 for the algorithm to determine this value.

The EncryptedData encryptedContentInfo contentEncryptionAlgorithm field MUST specify a symmetric encryption algorithm. Implementations MUST support encryption using AES, but implementations MAY support other algorithms as well.

The EncryptedData unprotectedAttrs MAY be present.

#### 4. Algorithm Identifiers and Parameters

This section specifies algorithm identifiers for the AES encryption algorithm.

##### 4.1. AES Algorithm Identifiers and Parameters

The AES algorithm is defined in [AES]. RSAES-OAEP [RSA-OAEP] MAY be used to transport AES keys.

AES is added to the set of symmetric content encryption algorithms defined in [CMSALG]. The AES content-encryption algorithm, in Cipher Block Chaining (CBC) mode, for the three different key sizes are identified by the following object identifiers:

```
id-aes128-CBC OBJECT IDENTIFIER ::= { aes 2 }
id-aes192-CBC OBJECT IDENTIFIER ::= { aes 22 }
id-aes256-CBC OBJECT IDENTIFIER ::= { aes 42 }
```

The AlgorithmIdentifier parameters field MUST be present, and the parameters field MUST contain a AES-IV:

```
AES-IV ::= OCTET STRING (SIZE(16))
```

Content encryption algorithm identifiers are located in the EnvelopedData EncryptedContentInfo contentEncryptionAlgorithm and the EncryptedData EncryptedContentInfo contentEncryptionAlgorithm fields.

Content encryption algorithms are used to encrypt the content located in the EnvelopedData EncryptedContentInfo encryptedContent and the EncryptedData EncryptedContentInfo encryptedContent fields.

## 5. SMIMECapabilities Attribute Conventions

An S/MIME client SHOULD announce the set of cryptographic functions it supports by using the S/MIME capabilities attribute. This attribute provides a partial list of object identifiers of cryptographic functions and MUST be signed by the client. The algorithm OIDs SHOULD be logically separated in functional categories and MUST be ordered with respect to their preference.

RFC 2633 [MSG], Section 2.5.2 defines the SMIMECapabilities signed attribute (defined as a SEQUENCE of SMIMECapability SEQUENCES) to be used to specify a partial list of algorithms that the software announcing the SMIMECapabilities can support.

### 5.1. AES S/MIME Capability Attributes

If an S/MIME client is required to support symmetric encryption with AES, the capabilities attribute MUST contain the AES object identifier specified above in the category of symmetric algorithms. The parameter with this encoding MUST be absent.

The encodings for the mandatory key sizes are:

Key Size	Capability
128	30 0B 06 09 60 86 48 01 65 03 04 01 02
196	30 0B 06 09 60 86 48 01 65 03 04 01 16
256	30 0B 06 09 60 86 48 01 65 03 04 01 2A

When a sending agent creates an encrypted message, it has to decide which type of encryption algorithm to use. In general the decision process involves information obtained from the capabilities lists included in messages received from the recipient, as well as other information such as private agreements, user preferences, legal restrictions, and so on. If users require AES for symmetric encryption, the S/MIME clients on both the sending and receiving side MUST support it, and it MUST be set in the user preferences.

## 6. Security Considerations

If RSA-OAEP [PKCS#1v2.0] and RSA PKCS #1 v1.5 [PKCS#1v1.5] are both used to transport the same CEK, then an attacker can still use the Bleichenbacher attack against the RSA PKCS #1 v1.5 encrypted key. It is generally unadvisable to mix both RSA-OAEP and RSA PKCS#1 v1.5 in the same set of recipients.

Implementations must protect the RSA private key and the CEK. Compromise of the RSA private key may result in the disclosure of all messages protected with that key. Compromise of the CEK may result in disclosure of the associated encrypted content.

The generation of AES CEKs relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate these values can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. RFC 1750 [RANDOM] offers important guidance in this area.

When wrapping a CEK with a KEK, the KEK MUST always be at least the same length as the CEK. An attacker will generally work at the weakest point in an encryption system. This would be the smaller of the two key sizes for a brute force attack.

## Normative References

- [AES] National Institute of Standards. FIPS Pub 197: Advanced Encryption Standard (AES). 26 November 2001.
- [CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.
- [AES-WRAP] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.
- [CMSALG] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.
- [DES] National Institute of Standards and Technology. FIPS Pub 46: Data Encryption Standard. 15 January 1977.
- [DH] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

- [MUSTSHOULD] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RSA-OAEP] Housley, R. "Use of the RSAES-OAEP Key Transport Algorithm in the Cryptographic Message Syntax (CMS)", RFC 3560, July 2003.
- [X.208-88] CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.
- [X.209-88] CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.
- [X.509-88] CCITT. Recommendation X.509: The Directory - Authentication Framework. 1988.

#### Informational References

- [MSG] Ramsdell, B., Editor, "S/MIME Version 3 Message Specification", RFC 2633, June 1999.
- [PKCS#1v1.5] Kaliski, B., "PKCS #1: RSA Encryption, Version 1.5", RFC 2313, March 1998.
- [PKCS#1v2.0] Kaliski, B., "PKCS #1: RSA Encryption, Version 2.0", RFC 2437, October 1998.
- [RANDOM] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.
- [SYMKEYDIST] Turner, S., "CMS Symmetric Key Management and Distribution", Work in Progress, January 2003.

#### Acknowledgements

This document is the result of contributions from many professionals. We appreciate the hard work of all members of the IETF S/MIME Working Group.

## Appendix A ASN.1 Module

```
CMSAesRsaesOaep {iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-aes(19) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL --
IMPORTS
    -- PKIX
    AlgorithmIdentifier
        FROM PKIXExplicit88 {iso(1) identified-organization(3) dod(6)
            internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
            id-pkix1-explicit(18)};

-- AES information object identifiers --

aes OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16) us(840)
    organization(1) gov(101) csor(3)_nistAlgorithms(4) 1 }

-- AES using CBC-chaining mode for key sizes of 128, 192, 256

id-aes128-CBC OBJECT IDENTIFIER ::= { aes 2 }
id-aes192-CBC OBJECT IDENTIFIER ::= { aes 22 }
id-aes256-CBC OBJECT IDENTIFIER ::= { aes 42 }

-- AES-IV is a the parameter for all the above object identifiers.

AES-IV ::= OCTET STRING (SIZE(16))

-- AES Key Wrap Algorithm Identifiers - Parameter is absent

id-aes128-wrap OBJECT IDENTIFIER ::= { aes 5 }
id-aes192-wrap OBJECT IDENTIFIER ::= { aes 25 }
id-aes256-wrap OBJECT IDENTIFIER ::= { aes 45 }

END

Author's Address

    Jim Schaad
    Soaring Hawk Consulting

    EMail: jimsch@exmsft.com
```

## Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

