Stream: Internet Engineering Task Force (IETF)
RFC: 9709
Category: Standards Track
Published: January 2025
ISSN: 2070-1721
Author: R. Housley
Vigil Security

RFC 9709
Encryption Key Derivation in the Cryptographic Message Syntax (CMS) Using HKDF with SHA-256

Abstract

This document specifies the derivation of the content-encryption key or the content-authenticated-encryption key in the Cryptographic Message Syntax (CMS) using the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) with SHA-256. The use of this mechanism provides protection against an attacker that manipulates the content-encryption algorithm identifier or the content-authenticated-encryption algorithm identifier.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9709.

Copyright Notice

Table of Contents

1. Introduction

This document specifies the derivation of the content-encryption key for the Cryptographic Message Syntax (CMS) enveloped-data content type [RFC5652], the content-encryption key for the CMS encrypted-data content type [RFC5652], or the content-authenticated-encryption key for the authenticated-enveloped-data content type [RFC5083].

The use of this mechanism provides protection against an attacker that manipulates the content-encryption algorithm identifier or the content-authenticated-encryption algorithm identifier. Johannes Roth and Falko Strenzke presented such an attack at IETF 118 [RS2023], where:

1. The attacker intercepts a CMS authenticated-enveloped-data content [RFC5083] that uses either AES-CCM or AES-GCM [RFC5084].
2. The attacker turns the intercepted content into a "garbage" CMS enveloped-data content (Section 6 of [RFC5652]) that is composed of AES-CBC guess blocks.
3. The attacker sends the "garbage" message to the victim, and the victim reveals the result of the decryption to the attacker.
4. If any of the transformed plaintext blocks match the guess for that block, then the attacker learns the plaintext for that block.

With highly structured messages, one block can reveal the only sensitive part of the original message.

This attack is thwarted if the encryption key depends upon the delivery of the unmodified algorithm identifier.

The mitigation for this attack has three parts:

1. Potential recipients include the id-alg-cek-hkdf-sha256 algorithm identifier (with no parameters) in S/MIME Capabilities to indicate support for this mitigation.
2. As a flag to the recipient that this mitigation is being used, carry the id-alg-cek-hkdf-sha256 algorithm identifier as the contentEncryptionAlgorithm in the EncryptedContentInfo structure. This structure is used in the enveloped-data content type, the encrypted-data content type, and the authenticated-enveloped-data content type. The parameters field of the id-alg-cek-hkdf-sha256 algorithm identifier identifies the content-encryption algorithm or the content-authenticated-encryption algorithm and any associated parameters.
3. Perform encryption with a derived content-encryption key or content-authenticated-encryption key:

```
CEK' = HKDF(CEK, AlgorithmIdentifier)
```

1.1. ASN.1

CMS values are generated using ASN.1 [X680], using the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [X690].

1.2. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Cryptographic Algorithm Agility Considerations

There is no provision for key derivation functions other than HKDF, and there is no provision for hash functions other than SHA-256. If there is ever a need to support another key derivation function or another hash function, it will be very straightforward to assign a new object identifier. At this point, keeping the design very simple seems most important.

2. Use of HKDF with SHA-256 to Derive Encryption Keys

The mitigation uses the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) [RFC5869] to derive output keying material (OKM) from input keying material (IKM). HKDF is used with the SHA-256 hash function [FIPS180]. The derivation includes the DER-encoded AlgorithmIdentifier as the optional info input value. The encoded value includes the ASN.1 tag for SEQUENCE (0x30), the length, and the value. This AlgorithmIdentifier is carried as the parameter to the id-alg-cek-hkdf-sha256 algorithm identifier. If an attacker were to change the originator-provided AlgorithmIdentifier, then the recipient will derive a different content-encryption key or content-authenticated-encryption key.

The CMS_CEK_HKDF_SHA256 function uses the HKDF-Extract and HKDF-Expand functions to derive the OKM from the IKM:

Inputs:
IKM input keying material
info DER-encoded AlgorithmIdentifier

Output:
OKM output keying material (same size as IKM)

The output OKM is calculated as follows:

```
OKM_SIZE = len(IKM)  /* length in octets */
IF OKM_SIZE > 8160 THEN raise error

salt = "The Cryptographic Message Syntax"
PRK = HKDF-Extract(salt, IKM)

OKM = HKDF-Expand(PRK, info, OKM_SIZE)
```

3. The id-alg-cek-hkdf-sha256 Algorithm Identifier

The id-alg-cek-hkdf-sha256 algorithm identifier indicates that the CMS_CEK_HKDF_SHA256 function defined in Section 2 is used to derive the content-encryption key or the content-authenticated-encryption key.

The following object identifier identifies the id-alg-cek-hkdf-sha256 algorithm:

```
id-alg-cek-hkdf-sha256 OBJECT IDENTIFIER ::= { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
   id-smime(16) alg(3) 31 }
```

The id-alg-cek-hkdf-sha256 parameters field has an ASN.1 type of AlgorithmIdentifier.

Using the conventions from [RFC5911], the id-alg-cek-hkdf-sha256 algorithm identifier is defined as:

```
ContentEncryptionAlgorithmIdentifier ::=
  AlgorithmIdentifier{CONTENT-ENCRYPTION, { ... } }

cea-CEKHKDFSHA256 CONTENT-ENCRYPTION ::= {
  IDENTIFIER id-alg-cek-hkdf-sha256
  PARAMS TYPE ContentEncryptionAlgorithmIdentifier ARE required
  SMIME-CAPS { IDENTIFIED BY id-alg-cek-hkdf-sha256 } }
```

4. SMIMECapabilities Attribute Conventions

The SMIMECapabilities attribute is defined in Section 2.5.2 of [RFC8551]. An S/MIME client announces the set of cryptographic functions it supports using the SMIMECapabilities attribute.

If an S/MIME client supports the mechanism in this document, the id-alg-cek-hkdf-sha256 object identifier **SHOULD** be included in the set of cryptographic functions. The parameter with this encoding **MUST** be absent.

The encoding for id-alg-cek-hkdf-sha256, in hexadecimal, is:

```
30 0d 06 0b 2a 86 48 86 f7 0d 01 09 10 03 1f
```

5. Use of HKDF with SHA-256 with CMS

This section describes the originator and recipient processing to implement this mitigation for each of the CMS encrypting content types.

5.1. Enveloped-Data Content Type

The fourth step of constructing an enveloped-data content type is repeated below from Section 6 of [RFC5652]:

> 4. The content is encrypted with the content-encryption key. Content encryption may require that the content be padded to a multiple of some block size; see Section 6.3.

To implement this mitigation, the originator expands this step as follows:

- Include the id-alg-cek-hkdf-sha256 algorithm identifier in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure, and set the contentEncryptionAlgorithm.parameters field to the AlgorithmIdentifier for the content-encryption algorithm that will be used to encrypt the content, including both the algorithm and optional parameters.
- Derive the new content-encryption key (CEK') from the original content-encryption key (CEK) and the ContentEncryptionAlgorithmIdentifier, which is carried in the contentEncryptionAlgorithm.parameters field:

```
CEK' = CMS_CEK_HKDF_SHA256(CEK, ContentEncryptionAlgorithmIdentifier)
```

- The content is encrypted with the new content-encryption key (CEK'). Content encryption may require that the content be padded to a multiple of some block size; see Section 6.3 of [RFC5652].

The presence of the id-alg-cek-hkdf-sha256 algorithm identifier in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure tells the recipient to derive the new content-encryption key (CEK') as shown above, and then use it for decryption of the EncryptedContent. If the id-alg-cek-hkdf-sha256 algorithm identifier is not present in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure, then the recipient uses the original content-encryption key (CEK) for decryption of the EncryptedContent.

5.2. Encrypted-Data Content Type

As specified in Section 8 of [RFC5652], the content-encryption key is managed by other means.

To implement this mitigation, the originator performs the following:

- Include the id-alg-cek-hkdf-sha256 algorithm identifier in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure, and set the contentEncryptionAlgorithm.parameters field to the AlgorithmIdentifier for the content-encryption algorithm that will be used to encrypt the content, including both the algorithm and optional parameters.
- Derive the new content-encryption key (CEK') from the original content-encryption key (CEK) and the ContentEncryptionAlgorithmIdentifier, which is carried in the contentEncryptionAlgorithm.parameters field:

```
CEK' = CMS_CEK_HKDF_SHA256(CEK, ContentEncryptionAlgorithmIdentifier)
```

- The content is encrypted with the new content-encryption key (CEK'). Content encryption may require that the content be padded to a multiple of some block size; see Section 6.3 of [RFC5652].

The presence of the id-alg-cek-hkdf-sha256 algorithm identifier in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure tells the recipient to derive the new content-encryption key (CEK') as shown above, and then use it for decryption of the EncryptedContent. If the id-alg-cek-hkdf-sha256 algorithm identifier is not present in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure, then the recipient uses the original content-encryption key (CEK) for decryption of the EncryptedContent.

5.3. Authenticated-Enveloped-Data Content Type

The fifth step of constructing an authenticated-enveloped-data content type is repeated below from Section 2 of [RFC5083]:

> 5. The attributes collected in step 4 are authenticated and the CMS content is authenticated and encrypted with the content- authenticated-encryption key. If the authenticated encryption algorithm requires either the additional authenticated data (AAD) or the content to be padded to a multiple of some block size, then the padding is added as described in Section 6.3 of [CMS].

> Note that [CMS] refers to [RFC3852], which has been obsoleted by [RFC5652], but the text in Section 6.3 was unchanged in RFC 5652.

To implement this mitigation, the originator expands this step as follows:

- Include the id-alg-cek-hkdf-sha256 algorithm identifier in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure, and set the contentEncryptionAlgorithm.parameters field to the AlgorithmIdentifier for the content-

authenticated-encryption algorithm that will be used for authenticated encryption of the content, including both the algorithm and optional parameters.

- Derive the new content-authenticated-encryption key (CEK') from the original content-authenticated-encryption key (CEK) and the ContentEncryptionAlgorithmIdentifier:

```
CEK' = CMS_CEK_HKDF_SHA256(CEK, ContentEncryptionAlgorithmIdentifier)
```

- The attributes collected in step 4 are authenticated and the CMS content is authenticated and encrypted with the new content-authenticated-encryption key (CEK'). If the authenticated encryption algorithm requires either the additional authenticated data (AAD) or the content to be padded to a multiple of some block size, then the padding is added as described in Section 6.3 of [RFC5652].

The presence of the id-alg-cek-hkdf-sha256 algorithm identifier in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure tells the recipient to derive the new content-authenticated-encryption key (CEK') as shown above, and then use it for authenticated decryption of the EncryptedContent and the authentication of the AAD. If the id-alg-cek-hkdf-sha256 algorithm identifier is not present in the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure, then the recipient uses the original content-authenticated-encryption (CEK) for decryption and authentication of the EncryptedContent and the authentication of the AAD.

6. Security Considerations

This mitigation always uses HKDF with SHA-256. One KDF algorithm was selected to avoid the need for negotiation. In the future, if a weakness is found in the KDF algorithm, a new attribute will need to be assigned for use with an alternative KDF algorithm.

If the attacker removes the id-alg-cek-hkdf-sha256 object identifier from the contentEncryptionAlgorithm.algorithm field of the EncryptedContentInfo structure prior to delivery to the recipient, then the recipient will not attempt to derive CEK', which will deny the recipient access to the content but will not assist the attacker in recovering the plaintext content.

If the attacker changes contentEncryptionAlgorithm.parameters field of the EncryptedContentInfo structure prior to delivery to the recipient, then the recipient will derive a different CEK', which will not assist the attacker in recovering the plaintext content. Providing the object identifier as an input to the key derivation function is sufficient to mitigate the attack described in [RS2023], but this mitigation includes both the object identifier and the parameters to protect against some yet-to-be-discovered attack that only manipulates the parameters.

Implementations **MUST** protect the content-encryption keys and content-authenticated-encryption keys, including the CEK and CEK'. Compromise of a content-encryption key may result in disclosure of the associated encrypted content. Compromise of a content-authenticated-encryption key may result in disclosure of the associated encrypted content or allow modification of the authenticated content and the AAD.

Implementations **MUST** randomly generate content-encryption keys and content-authenticated-encryption keys. Using an inadequate pseudorandom number generator (PRNG) to generate cryptographic keys can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys and then search the resulting small set of possibilities, rather than brute-force searching the whole key space. The generation of quality random numbers is difficult. [RFC4086] offers important guidance on this topic.

7. Privacy Considerations

If the message-digest attribute is included in the AuthAttributes, then the attribute value will contain the unencrypted one-way hash value of the plaintext of the content. Disclosure of this hash value enables content tracking, and it can be used to determine if the content matches one or more candidates. For these reasons, the AuthAttributes **SHOULD NOT** contain the message-digest attribute.

8. Operational Considerations

CMS is often used to provide encryption in messaging environments, where various forms of unsolicited messages (such as spam and phishing) represent a significant volume of unwanted traffic. Mitigation strategies for unwanted message traffic involve analysis of plaintext message content. When recipients accept unsolicited encrypted messages, they become even more vulnerable to unwanted traffic since many mitigation strategies will be unable to access the plaintext message content. Therefore, software that receives messages that have been encrypted using CMS ought to provide alternate mechanisms to handle the unwanted message traffic. One approach that does not require disclosure of keying material to a server is to reject or discard encrypted messages unless they purport to come from a member of a previously approved originator list.

9. IANA Considerations

For the ASN.1 module in Appendix A of this document, IANA has assigned the following object identifier (OID) in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry:

Decimal	Description	References
80	id-mod-CMS-CEK-HKDF-SHA256-2023	RFC 9709

Table 1

IANA has allocated the id-alg-cek-hkdf-sha256 algorithm identifier as specified in Section 3 in the "SMI Security for S/MIME Algorithms (1.2.840.113549.1.9.16.3)" registry as follows:

Decimal	Description	References
31	id-alg-cek-hkdf-sha256	RFC 9709

Table 2

10. References

10.1. Normative References

[FIPS180] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, DOI 10.17487/RFC5083, November 2007, <https://www.rfc-editor.org/info/rfc5083>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[X680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680>.

[X690] ITU-T, "Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1-2021, February 2021, <https://www.itu.int/rec/T-REC-X.690>.

10.2. Informative References

[RFC3852]

Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, DOI 10.17487/RFC3852, July 2004, <https://www.rfc-editor.org/info/rfc3852>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5084] Housley, R., "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)", RFC 5084, DOI 10.17487/RFC5084, November 2007, <https://www.rfc-editor.org/info/rfc5084>.

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/info/rfc5911>.

[RS2023] Roth, J. and F. Strenzke, "AEAD-to-CBC Downgrade Attacks on CMS", IETF 118 Proceedings, 8 November 2023, <https://datatracker.ietf.org/meeting/118/materials/slides-118-lamps-attack-against-aead-in-cms>.

Appendix A. ASN.1 Module

This ASN.1 module builds upon the conventions established in [RFC5911].

```
<CODE BEGINS>
CMS-CEK-HKDF-SHA256-Module-2023
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    id-smime(16) id-mod(0) id-mod-CMS-CEK-HKDF-SHA256-2023(80) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS
  AlgorithmIdentifier{}, CONTENT-ENCRYPTION, SMIME-CAPS
  FROM AlgorithmInformation-2009 -- in RFC 5911
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-algorithmInformation-02(58) } ;


--
-- CEK-HKDF-SHA256 Algorithm
--

id-alg-cek-hkdf-sha256 OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) id-smime(16) alg(3) 31 }

ContentEncryptionAlgorithmIdentifier ::=
    AlgorithmIdentifier{CONTENT-ENCRYPTION, { ... } }

cea-CEKHKDFSHA256 CONTENT-ENCRYPTION ::= {
    IDENTIFIER id-alg-cek-hkdf-sha256
    PARAMS TYPE ContentEncryptionAlgorithmIdentifier ARE required
    SMIME-CAPS { IDENTIFIED BY id-alg-cek-hkdf-sha256 } }

--
-- S/MIME Capability for CEK-HKDF-SHA256 Algorithm
--

SMimeCaps SMIME-CAPS ::= { cap-CMSCEKHKDFSHA256, ... }

cap-CMSCEKHKDFSHA256 SMIME-CAPS ::=
    { -- No value -- IDENTIFIED BY id-alg-cek-hkdf-sha256 }

END

<CODE ENDS>
```

Appendix B. CMS_CEK_HKDF_SHA256 Function Examples

This appendix provides two test vectors for the CMS_CEK_HKDF_SHA256 function.

B.1. CMS_CEK_HKDF_SHA256 with AES-128-GCM

This test vector includes an AlgorithmIdentifier for AES-128-GCM.

```
IKM = c702e7d0a9e064b09ba55245fb733cf3

The AES-128-GCM AlgorithmIdentifier:
 algorithm=2.16.840.1.101.3.4.1.6
 parameters=GCMParameters:
  aes-nonce=0x5c79058ba2f43447639d29e2
  aes-ICVlen is omitted; it indicates the DEFAULT of 12

DER-encoded AlgorithmIdentifier:
  301b06096086480165030401063 00e040c5c79058ba2f43447639d29e2

OKM = 2124ffb29fac4e0fbbc7d5d87492bff3
```

B.2. CMS_CEK_HKDF_SHA256 with AES-128-CBC

This test vector uses includes an AlgorithmIdentifier for AES-128-CBC.

```
IKM = c702e7d0a9e064b09ba55245fb733cf3

The AES-128-CBC AlgorithmIdentifier:
 algorithm=2.16.840.1.101.3.4.1.2
 parameters=AES-IV=0x651f722ffd512c52fe072e507d72b377

DER-encoded AlgorithmIdentifier:
  301d06096086480165030401020410651f722ffd512c52fe072e507d72b377

OKM = 9cd102c52f1e19ece8729b35bfeceb50
```

Acknowledgements

Thanks to Mike Ounsworth, Carl Wallace, and Joe Mandel their careful review and constructive comments.

Author's Address

Russ Housley
Vigil Security, LLC
Herndon, VA
United States of America
Email: housley@vigilsec.com