Internet Engineering Task Force (IETF)
Request for Comments: 9459
Category: Standards Track
ISSN: 2070-1721

R. Housley
Vigil Security
H. Tschofenig

September 2023

CBOR Object Signing and Encryption (COSE): AES-CTR and AES-CBC

Abstract

The Concise Binary Object Representation (CBOR) data format is designed for small code size and small message size. CBOR Object Signing and Encryption (COSE) is specified in RFC 9052 to provide basic security services using the CBOR data format. This document specifies the conventions for using AES-CTR and AES-CBC as content encryption algorithms with COSE.

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/rfc9459.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

This document specifies the conventions for using AES-CTR and AES-CBC as content encryption algorithms with the CBOR Object Signing and Encryption (COSE) [RFC9052] syntax. Today, encryption with COSE uses Authenticated Encryption with Associated Data (AEAD) algorithms [RFC5116], which provide both confidentiality and integrity protection. However, there are situations where another mechanism, such as a digital signature, is used to provide integrity. In these cases, an AEAD algorithm is not needed. The software manifest being defined by the IETF SUIT WG [SUIT-MANIFEST] is one example where a digital signature is always present.

2. Conventions and 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.

3. AES Modes of Operation

NIST has defined several modes of operation for the Advanced Encryption Standard [AES] [MODES]. AES supports three key sizes: 128 bits, 192 bits, and 256 bits. AES has a block size of 128 bits (16 octets). Each of these modes has different characteristics. The modes include: CBC (Cipher Block Chaining), CFB (Cipher FeedBack), OFB (Output FeedBack), and CTR (Counter).

Only AES Counter (AES-CTR) mode and AES Cipher Block Chaining (AES-CBC) are discussed in this document.

4. AES Counter Mode

When AES-CTR is used as a COSE content encryption algorithm, the encryptor generates a unique value that is communicated to the decryptor. This value is called an "Initialization Vector" (or "IV") in this document. The same IV and AES key combination MUST NOT be used more than once. The encryptor can generate the IV in any manner that ensures the same IV value is not used more than once with the same AES key.

When using AES-CTR, each AES encrypt operation generates 128 bits of key stream. AES-CTR encryption is the XOR of the key stream with the plaintext. AES-CTR decryption is the XOR of the key stream with the ciphertext. If the generated key stream is longer than the plaintext or ciphertext, the extra key stream bits are simply discarded. For this reason, AES-CTR does not require the plaintext to be padded to a multiple of the block size.

AES-CTR has many properties that make it an attractive COSE content encryption algorithm. AES-CTR uses the AES block cipher to create a stream cipher. Data is encrypted and decrypted by XORing with the key stream produced by AES encrypting sequential IV block values, called "counter blocks", where:

- The first block of the key stream is the AES encryption of the IV.

- The second block of the key stream is the AES encryption of (IV + 1) mod 2^128.

- The third block of the key stream is the AES encryption of (IV + 2) mod 2^128, and so on.

AES-CTR is easy to implement, can be pipelined and parallelized, and supports key stream precomputation. Sending of the IV is the only source of expansion because the plaintext and ciphertext are the same size.

When used correctly, AES-CTR provides a high level of confidentiality. Unfortunately, AES-CTR is easy to use incorrectly. Being a stream cipher, reuse of the IV with the same key is catastrophic. An IV collision immediately leaks information about the plaintext. For this reason, it is inappropriate to use AES-CTR with static keys. Extraordinary measures would be needed to prevent reuse of an IV value with the static key across power cycles. To be safe, implementations MUST use fresh keys with AES-CTR.

AES-CTR keys may be obtained either from a key structure or from a recipient structure. Implementations encrypting and decrypting MUST validate that the key type, key length, and algorithm are correct and appropriate for the entities involved.

With AES-CTR, it is trivial to use a valid ciphertext to forge other (valid to the decryptor) ciphertexts. Thus, it is equally catastrophic to use AES-CTR without a companion authentication and integrity mechanism. Implementations MUST use AES-CTR in conjunction with an authentication and integrity mechanism, such as a digital signature.

The instructions in Section 5.4 of [RFC9052] are followed for AES-CTR. Since AES-CTR cannot provide integrity protection for external additional authenticated data, the decryptor MUST ensure that no external additional authenticated data was supplied. See Section 6.

The 'protected' header MUST be a zero-length byte string.

4.1. AES-CTR COSE Key

When using a COSE key for the AES-CTR algorithm, the following checks are made:

- The 'kty' field MUST be present, and it MUST be 'Symmetric'.

- If the 'alg' field is present, it MUST match the AES-CTR algorithm being used.

- If the 'key_ops' field is present, it MUST include 'encrypt' when encrypting.

- If the 'key_ops' field is present, it MUST include 'decrypt' when decrypting.

4.2. AES-CTR COSE Algorithm Identifiers

The following table defines the COSE AES-CTR algorithm values. Note that these algorithms are being registered as "Deprecated" to avoid accidental use without a companion integrity protection mechanism.

Name	Value	Key Size	Description	Recommended
A128CTR	-65534	128	AES-CTR w/ 128-bit key	Deprecated
A192CTR	-65533	192	AES-CTR w/ 192-bit key	Deprecated
A256CTR	-65532	256	AES-CTR w/ 256-bit key	Deprecated

Table 1

5. AES Cipher Block Chaining Mode

AES-CBC mode requires a 16-octet IV. Use of a randomly or pseudorandomly generated IV ensures that the encryption of the same plaintext will yield different ciphertext.

AES-CBC performs an XOR of the IV with the first plaintext block before it is encrypted. For successive blocks, AES-CBC performs an XOR of the previous ciphertext block with the current plaintext before it is encrypted.

AES-CBC requires padding of the plaintext; the padding algorithm specified in Section 6.3 of [RFC5652] MUST be used prior to encrypting the plaintext. This padding algorithm allows the decryptor to unambiguously remove the padding.

The simplicity of AES-CBC makes it an attractive COSE content encryption algorithm. The need to carry an IV and the need for padding lead to an increase in the overhead (when compared to AES-CTR). AES-CBC is much safer for use with static keys than AES-CTR. That said, as described in [RFC4107], the use of automated key management to generate fresh keys is greatly preferred.

AES-CBC does not provide integrity protection. Thus, an attacker can introduce undetectable errors if AES-CBC is used without a companion authentication and integrity mechanism. Implementations MUST use AES-CBC in conjunction with an authentication and integrity mechanism, such as a digital signature.

The instructions in Section 5.4 of [RFC9052] are followed for AES-CBC. Since AES-CBC cannot provide integrity protection for external additional authenticated data, the decryptor MUST ensure that no external additional authenticated data was supplied. See Section 6.

The 'protected' header MUST be a zero-length byte string.

5.1. AES-CBC COSE Key

When using a COSE key for the AES-CBC algorithm, the following checks are made:

* The 'kty' field MUST be present, and it MUST be 'Symmetric'.

* If the 'alg' field is present, it MUST match the AES-CBC algorithm being used.

* If the 'key_ops' field is present, it MUST include 'encrypt' when encrypting.

* If the 'key_ops' field is present, it MUST include 'decrypt' when decrypting.

5.2. AES-CBC COSE Algorithm Identifiers

The following table defines the COSE AES-CBC algorithm values. Note that these algorithms are being registered as "Deprecated" to avoid accidental use without a companion integrity protection mechanism.

Name	Value	Key Size	Description	Recommended
A128CBC	-65531	128	AES-CBC w/ 128-bit key	Deprecated
A192CBC	-65530	192	AES-CBC w/ 192-bit key	Deprecated
A256CBC	-65529	256	AES-CBC w/ 256-bit key	Deprecated

Table 2

6. Implementation Considerations

COSE libraries that support either AES-CTR or AES-CBC and accept Additional Authenticated Data (AAD) as input MUST return an error if one of these non-AEAD content encryption algorithms is selected. This ensures that a caller does not expect the AAD to be protected when the cryptographic algorithm is unable to do so.

7. IANA Considerations

IANA has registered six COSE algorithm identifiers for AES-CTR and AES-CBC in the "COSE Algorithms" registry [IANA-COSE].

The information for the six COSE algorithm identifiers is provided in Sections 4.2 and 5.2. Also, for all six entries, the "Capabilities" column contains "[kty]", the "Change Controller" column contains "IETF", and the "Reference" column contains a reference to this document.

8. Security Considerations

This document specifies AES-CTR and AES-CBC for COSE, which are not AEAD ciphers. The use of the ciphers is limited to special use cases, such as firmware encryption, where integrity and authentication is provided by another mechanism.

Since AES has a 128-bit block size, regardless of the mode employed, the ciphertext generated by AES encryption becomes distinguishable from random values after 2^64 blocks are encrypted with a single key. Implementations should change the key before reaching this limit.

To avoid cross-protocol concerns, implementations MUST NOT use the same keying material with more than one mode. For example, the same keying material must not be used with AES-CTR and AES-CBC.

There are fairly generic precomputation attacks against all block cipher modes that allow a meet-in-the-middle attack against the key. These attacks require the creation and searching of huge tables of ciphertext associated with known plaintext and known keys. Assuming that the memory and processor resources are available for a precomputation attack, then the theoretical strength of AES-CTR and AES-CBC is limited to 2^(n/2) bits, where n is the number of bits in the key. The use of long keys is the best countermeasure to precomputation attacks.

When used properly, AES-CTR mode provides strong confidentiality. Unfortunately, it is very easy to misuse this counter mode. If counter block values are ever used for more than one plaintext with the same key, then the same key stream will be used to encrypt both plaintexts, and the confidentiality guarantees are voided.

What happens if the encryptor XORs the same key stream with two different plaintexts? Suppose two plaintext octet sequences P1, P2, P3 and Q1, Q2, Q3 are both encrypted with key stream K1, K2, K3. The two corresponding ciphertexts are:

```
(P1 XOR K1), (P2 XOR K2), (P3 XOR K3)

(Q1 XOR K1), (Q2 XOR K2), (Q3 XOR K3)
```

If both of these two ciphertext streams are exposed to an attacker, then a catastrophic failure of confidentiality results, since:

```
(P1 XOR K1) XOR (Q1 XOR K1) = P1 XOR Q1
(P2 XOR K2) XOR (Q2 XOR K2) = P2 XOR Q2
(P3 XOR K3) XOR (Q3 XOR K3) = P3 XOR Q3
```

Once the attacker obtains the two plaintexts XORed together, it is relatively straightforward to separate them. Thus, using any stream cipher, including AES-CTR, to encrypt two plaintexts under the same key stream leaks the plaintext.

Data forgery is trivial with AES-CTR mode. The demonstration of this attack is similar to the key stream reuse discussion above. If a known plaintext octet sequence P1, P2, P3 is encrypted with key stream K1, K2, K3, then the attacker can replace the plaintext with one of its own choosing. The ciphertext is:

```
(P1 XOR K1), (P2 XOR K2), (P3 XOR K3)
```

The attacker simply XORs a selected sequence Q1, Q2, Q3 with the ciphertext to obtain:

```
(Q1 XOR (P1 XOR K1)), (Q2 XOR (P2 XOR K2)), (Q3 XOR (P3 XOR K3))
```

Which is the same as:

```
((Q1 XOR P1) XOR K1), ((Q2 XOR P2) XOR K2), ((Q3 XOR P3) XOR K3)
```

Decryption of the attacker-generated ciphertext will yield exactly what the attacker intended:

```
(Q1 XOR P1), (Q2 XOR P2), (Q3 XOR P3)
```

AES-CBC does not provide integrity protection. Thus, an attacker can introduce undetectable errors if AES-CBC is used without a companion authentication mechanism.

If an attacker is able to strip the authentication and integrity mechanism, then the attacker can replace it with one of their own creation, even without knowing the plaintext. The usual defense against such an attack is an Authenticated Encryption with Associated Data (AEAD) algorithm [RFC5116]. Of course, neither AES-CTR nor AES-CBC is an AEAD. Thus, an implementation should provide integrity protection for the 'kid' field to prevent undetected stripping of the authentication and integrity mechanism; this prevents an attacker from altering the 'kid' to trick the recipient into using a different key.

With AES-CBC mode, implementers should perform integrity checks prior to decryption to avoid padding oracle vulnerabilities [Vaudenay].

With the assignment of COSE algorithm identifiers for AES-CTR and AES-CBC in the COSE Algorithms Registry, an attacker can replace the COSE algorithm identifiers with one of these identifiers. Then, the attacker might be able to manipulate the ciphertext to learn some of the plaintext or extract the keying material used for authentication and integrity.

Since AES-CCM [RFC3610] and AES-GCM [GCMMODE] use AES-CTR for encryption, an attacker can switch the algorithm identifier to AES-CTR and then strip the authentication tag to bypass the authentication and integrity, allowing the attacker to manipulate the ciphertext.

An attacker can switch the algorithm identifier from AES-GCM to AES-CBC, guessing 16 bytes of plaintext at a time, and see if the recipient accepts the padding. Padding oracle vulnerabilities are discussed further in [Vaudenay].

9. References

9.1. Normative References

[AES]   National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", NIST FIPS 197, DOI 10.6028/NIST.FIPS.197-upd1, May 2023, <https://doi.org/10.6028/NIST.FIPS.197-upd1>.

[MODES]   Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", NIST Special Publication 800-38A, DOI 10.6028/NIST.SP.800-38A, December 2001, <https://doi.org/10.6028/NIST.SP.800-38A>.

[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>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <https://www.rfc-editor.org/info/rfc4107>.

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

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

9.2. Informative References

[GCMMODE] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, DOI 10.6028/NIST.SP.800-38D, November 2007, <https://doi.org/10.6028/NIST.SP.800-38D>.

[IANA-COSE]

IANA, "CBOR Object Signing and Encryption (COSE)", <https://www.iana.org/assignments/cose>.

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, DOI 10.17487/RFC3610, September 2003, <https://www.rfc-editor.org/info/rfc3610>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[SUIT-MANIFEST] Moran, B., Tschofenig, H., Birkholz, H., Zandberg, K., and Ø. Rønningstad, "A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest", Work in Progress, Internet-Draft, draft-ietf-suit-manifest-22, 27 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-suit-manifest-22>.

[Vaudenay] Vaudenay, S., "Security Flaws Induced by CBC Padding -- Applications to SSL, IPSEC, WTLS...", EUROCRYPT 2002, 2002, <https://www.iacr.org/cryptodb/archive/2002/EUROCRYPT/2850/2850.pdf>.

Acknowledgements

Many thanks to David Brown for raising the need for non-AEAD algorithms to support encryption within the SUIT manifest. Many thanks to Ilari Liusvaara, Scott Arciszewski, John Preuß Mattsson, Laurence Lundblade, Paul Wouters, Roman Danyliw, Sophie Schmieg, Stephen Farrell, Carsten Bormann, Scott Fluhrer, Brendan Moran, and John Scudder for the review and thoughtful comments.

Authors' Addresses

Russ Housley
Vigil Security, LLC
Email: housley@vigilsec.com

Hannes Tschofenig
Email: hannes.tschofenig@gmx.net