Internet Engineering Task Force (IETF)
Request for Comments: 6655
Category: Standards Track
ISSN: 2070-1721

D. McGrew
Cisco Systems
D. Bailey
RSA, Security Division of EMC

July 2012

AES-CCM Cipher Suites for Transport Layer Security (TLS)

Abstract

This memo describes the use of the Advanced Encryption Standard (AES) in the Counter with Cipher Block Chaining - Message Authentication Code (CBC-MAC) Mode (CCM) of operation within Transport Layer Security (TLS) and Datagram TLS (DTLS) to provide confidentiality and data origin authentication. The AES-CCM algorithm is amenable to compact implementations, making it suitable for constrained environments.

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 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6655.

Table of Contents

1. Introduction

This document describes the use of Advanced Encryption Standard (AES) [AES] in Counter with CBC-MAC Mode (CCM) [CCM] in several TLS ciphersuites. AES-CCM provides both authentication and confidentiality and uses as its only primitive the AES encrypt operation (the AES decrypt operation is not needed). This makes it amenable to compact implementations, which is advantageous in constrained environments. Of course, adoption outside of constrained environments is necessary to enable interoperability, such as that between web clients and embedded servers or between embedded clients and web servers. The use of AES-CCM has been specified for IPsec Encapsulating Security Payload (ESP) [RFC4309] and 802.15.4 wireless networks [IEEE802154].

Authenticated encryption, in addition to providing confidentiality for the plaintext that is encrypted, provides a way to check its integrity and authenticity. Authenticated Encryption with Associated Data, or AEAD [RFC5116], adds the ability to check the integrity and authenticity of some associated data that is not encrypted. This document utilizes the AEAD facility within TLS 1.2 [RFC5246] and the AES-CCM-based AEAD algorithms defined in [RFC5116]. Additional AEAD algorithms are defined that use AES-CCM but have shorter authentication tags and are therefore more suitable for use across networks in which bandwidth is constrained and message sizes may be small.

The ciphersuites defined in this document use RSA or Pre-Shared Key (PSK) as their key establishment mechanism; these ciphersuites can be used with DTLS [RFC6347]. Since the ability to use AEAD ciphers was introduced in DTLS version 1.2, the ciphersuites defined in this document cannot be used with earlier versions of that protocol.

2. 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 [RFC2119].

3. RSA-Based AES-CCM Cipher Suites

The ciphersuites defined in this document are based on the AES-CCM Authenticated Encryption with Associated Data (AEAD) algorithms AEAD_AES_128_CCM and AEAD_AES_256_CCM described in [RFC5116]. The following RSA-based ciphersuites are defined:

```
CipherSuite TLS_RSA_WITH_AES_128_CCM       = {0xC0,0x9C}
CipherSuite TLS_RSA_WITH_AES_256_CCM       = {0xC0,0x9D)
CipherSuite TLS_DHE_RSA_WITH_AES_128_CCM   = {0xC0,0x9E}
CipherSuite TLS_DHE_RSA_WITH_AES_256_CCM   = {0xC0,0x9F}
CipherSuite TLS_RSA_WITH_AES_128_CCM_8     = {0xC0,0xA0}
CipherSuite TLS_RSA_WITH_AES_256_CCM_8     = {0xC0,0xA1)
CipherSuite TLS_DHE_RSA_WITH_AES_128_CCM_8 = {0xC0,0xA2}
CipherSuite TLS_DHE_RSA_WITH_AES_256_CCM_8 = {0xC0,0xA3}
```

These ciphersuites make use of the AEAD capability in TLS 1.2 [RFC5246]. Each uses AES-CCM; those that end in "_8" have an 8-octet authentication tag, while the other ciphersuites have 16-octet authentication tags.

The Hashed Message Authentication Code (HMAC) truncation option described in Section 7 of [RFC6066] (which negotiates the "truncated_hmac" TLS extension) does not have an effect on cipher suites that do not use HMAC.

The "nonce" input to the AEAD algorithm is exactly that of [RFC5288]: the "nonce" SHALL be 12 bytes long and is constructed as follows: (this is an example of a "partially explicit" nonce; see Section 3.2.1 in [RFC5116]).

```
          struct {
opaque salt[4];
opaque nonce_explicit[8];
          } CCMNonce;
```

The salt is the "implicit" part of the nonce and is not sent in the packet. Instead, the salt is generated as part of the handshake process: it is either the client_write_IV (when the client is sending) or the server_write_IV (when the server is sending). The salt length (SecurityParameters.fixed_iv_length) is 4 octets. The nonce_explicit is the "explicit" part of the nonce. It is chosen by the sender and is carried in each TLS record in the GenericAEADCipher.nonce_explicit field. The nonce_explicit length (SecurityParameters.record_iv_length) is 8 octets. Each value of the nonce_explicit MUST be distinct for each distinct invocation of the GCM encrypt function for any fixed key. Failure to meet this uniqueness requirement can significantly degrade security. The nonce_explicit MAY be the 64-bit sequence number (as long as those values are assured to meet the distinctness requirement).

In DTLS, the 64-bit seq_num is the 16-bit epoch concatenated with the 48-bit seq_num.

When the nonce_explicit is equal to the sequence number, the CCMNonce will have the structure of the CCMNonceExample given below.

```
struct {
 uint32 client_write_IV; // low order 32-bits
 uint64 seq_num;         // TLS sequence number
} CCMClientNonce.


struct {
 uint32 server_write_IV; // low order 32-bits
 uint64 seq_num; // TLS sequence number
} CCMServerNonce.


struct {
 case client:
   CCMClientNonce;
 case server:
   CCMServerNonce:
} CCMNonceExample;
```

These ciphersuites make use of the default TLS 1.2 Pseudorandom Function (PRF), which uses HMAC with the SHA-256 hash function. The RSA and DHE_RSA, key exchange is performed as defined in [RFC5246].

4. PSK-Based AES-CCM Cipher Suites

As in Section 3, these ciphersuites follow [RFC5116]. The PSK and DHE_PSK key exchange is performed as in [RFC4279]. The following ciphersuites are defined:

```
CipherSuite TLS_PSK_WITH_AES_128_CCM       = {0xC0,0xA4}
CipherSuite TLS_PSK_WITH_AES_256_CCM       = {0xC0,0xA5)
CipherSuite TLS_DHE_PSK_WITH_AES_128_CCM   = {0xC0,0xA6}
CipherSuite TLS_DHE_PSK_WITH_AES_256_CCM   = {0xC0,0xA7}
CipherSuite TLS_PSK_WITH_AES_128_CCM_8     = {0xC0,0xA8}
CipherSuite TLS_PSK_WITH_AES_256_CCM_8     = {0xC0,0xA9)
CipherSuite TLS_PSK_DHE_WITH_AES_128_CCM_8 = {0xC0,0xAA}
CipherSuite TLS_PSK_DHE_WITH_AES_256_CCM_8 = {0xC0,0xAB}
```

The "nonce" input to the AEAD algorithm is defined as in Section 3.

These ciphersuites make use of the default TLS 1.2 Pseudorandom Function (PRF), which uses HMAC with the SHA-256 hash function. The PSK and DHE_PSK key exchange is performed as defined in [RFC5487].

5. TLS Versions

These ciphersuites make use of the authenticated encryption with additional data (AEAD) defined in TLS 1.2 [RFC5288]. Earlier versions of TLS do not have support for AEAD; for instance, the TLSCiphertext structure does not have the "aead" option in TLS 1.1. Consequently, these ciphersuites MUST NOT be negotiated in older versions of TLS. Clients MUST NOT offer these cipher suites if they do not offer TLS 1.2 or later. Servers that select an earlier version of TLS MUST NOT select one of these cipher suites. Because TLS has no way for the client to indicate that it supports TLS 1.2 but not earlier, a non-compliant server might potentially negotiate TLS 1.1 or earlier and select one of the cipher suites in this document. Clients MUST check the TLS version and generate a fatal "illegal_parameter" alert if they detect an incorrect version.

6. New AEAD Algorithms

The following AEAD algorithms are defined:

```
AEAD_AES_128_CCM_8     = 18
AEAD_AES_256_CCM_8     = 19
```

6.1. AES-128-CCM with an 8-Octet Integrity Check Value (ICV)

The AEAD_AES_128_CCM_8 authenticated encryption algorithm is identical to the AEAD_AES_128_CCM algorithm (see Section 5.3 of [RFC5116]), except that it uses 8 octets for authentication, instead of the full 16 octets used by AEAD_AES_128_CCM. The AEAD_AES_128_CCM_8 ciphertext consists of the ciphertext output of the CCM encryption operation concatenated with the 8-octet authentication tag output of the CCM encryption operation. Test cases are provided in [CCM]. The input and output lengths are the same as those for AEAD_AES_128_CCM. An AEAD_AES_128_CCM_8 ciphertext is exactly 8 octets longer than its corresponding plaintext.

6.2. AES-256-CCM with a 8-Octet Integrity Check Value (ICV)

The AEAD_AES_256_CCM_8 authenticated encryption algorithm is identical to the AEAD_AES_256_CCM algorithm (see Section 5.4 of [RFC5116]), except that it uses 8 octets for authentication, instead of the full 16 octets used by AEAD_AES_256_CCM. The AEAD_AES_256_CCM_8 ciphertext consists of the ciphertext output of the CCM encryption operation concatenated with the 8-octet authentication tag output of the CCM encryption operation. Test cases are provided in [CCM]. The input and output lengths are as for AEAD_AES_128_CCM. An AEAD_AES_128_CCM_8 ciphertext is exactly 8 octets longer than its corresponding plaintext.

7. IANA Considerations

IANA has assigned the values for the ciphersuites defined in Sections 3 and 4 from the "TLS Cipher Suite" registry and the values of the AEAD algorithms defined in Section 6 from the "AEAD Algorithms" registry.

8. Security Considerations

8.1. Perfect Forward Secrecy

The perfect forward secrecy properties of RSA-based TLS ciphersuites are discussed in the security analysis of [RFC5246]. It should be noted that only the ephemeral Diffie-Hellman-based ciphersuites are capable of providing perfect forward secrecy.

8.2. Counter Reuse

AES-CCM security requires that the counter is never reused. The IV construction in Section 3 is designed to prevent counter reuse.

9. Acknowledgements

This document borrows heavily from [RFC5288]. Thanks are due to Stephen Farrell and Robert Cragie for their input.

10. References

10.1. Normative References

[AES] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)", FIPS 197, November 2001.

[CCM] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality", SP 800-38C, May 2004.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, August 2008.

[RFC5487] Badra, M., "Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode", RFC 5487, March 2009.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

10.2. Informative References

[IEEE802154] Institute of Electrical and Electronics Engineers, "Wireless Personal Area Networks", IEEE Standard 802.15.4-2006, 2006.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.

Authors' Addresses

David McGrew
Cisco Systems
13600 Dulles Technology Drive
Herndon, VA 20171
USA

EMail: mcgrew@cisco.com

Daniel V. Bailey
RSA, Security Division of EMC
174 Middlesex Tpke.
Bedford, MA 01463
USA

EMail: dbailey@rsa.com