Network Working Group
Internet-Draft
Expires: August 30, 2006

N. Modadugu
Stanford University
E. Rescorla
Network Resonance
February 26, 2006

AES Counter Mode Cipher Suites for TLS and DTLS
draft-ietf-tls-ctr-00.txt

Status of this Memo

Copyright Notice

Abstract

This document describes the use of the Advanced Encryption Standard (AES) Counter Mode for use as a Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) confidentiality mechanism.

Table of Contents

1. Introduction

Transport Layer Security [3] provides channel-oriented security for application layer protocols. In TLS, cryptographic algorithms are specified in "Cipher Suites, which consist of a group of algorithms to be used together."

Cipher suites supported by TLS are divided into stream and block ciphers. Counter mode ciphers behave like stream ciphers, but are constructed based on a block cipher primitive (that is, counter mode operation of a block cipher results in a stream cipher.) This specification is limited to discussion of the operation of AES in counter mode (AES-CTR.)

Counter mode ciphers (CTR) offer a number of attractive features over other block cipher modes and stream ciphers such as RC4:

Low Bandwidth: AES-CTR provides a saving of 17-32 bytes per record compared to AES-CBC as used in TLS 1.1 and DTLS. 16 bytes are saved from not having to transmit an explicit IV, and another 1-16 bytes are saved from the absence of the padding block.

Random Access: AES-CTR is capable of random access within the key stream. For DTLS, this implies that records can be processed out of order without dependency on packet arrival order, and also without keystream buffering.

Parallelizable: As a consequence of AES-CTR supporting random access within the key stream, the cipher can be easily parallelized.

Multiple mode support: AES-CTR support in TLS/DTLS allows for implementator to support both a stream (CTR) and block (CBC) cipher through the implemention of a single symmetric algorithm.

1.1. 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 [1].

2. Terminology

This document reuses some terminology introduced in [2] and [3]. The term 'counter block' has the same meaning as used in RFC3686, however, the term 'IV', in this document, holds the meaning defined in [3].

3. Encrypting Records with AES Counter Mode

The use of AES-CTR in TLS/DTLS turns out to be fairly straightforward, with the additional benefit that the method of operation in TLS/DTLS mimics, to a large extent, that in IPsec. The primary constraint on the use of counter mode ciphers is that for a given key, a counter block value MUST never be used more than once (see Section 7 of [2] for a detailed explanation.) In TLS/DTLS ensuring that counter block values never repeat during a given session is straightforward as explained in the following sections.

SSL/TLS records encrypted with AES-CTR mode use a CipherSpec.cipher_type of GenericStreamCipher (Section 6.2.3 of [3]).

3.1. TLS

The cipher stream generated by AES-CTR is much like the cipher stream generated by stream ciphers like RC4. For reasons described in Section 7 of [2], a counter block value MUST never be used more than once with a given key. This is achieved by having part of the per-record IV determined by the record sequence number. Although the client and server use the same sequence number space, they use different keys and IVs.

3.1.1. AES Counter Mode

AES counter mode requires the encryptor and decryptor to share a per-record unique counter block. A given counter block MUST never be used more than once with the same key. For a more in-depth

discussion of AES-CTR operation, refer to Section 2.1 of [2]. The following description of AES-CTR mode has been adapted from [2].

To encrypt a payload with AES-CTR, the encryptor partitions the plaintext, PT, into 128-bit blocks. The final block MAY be less than 128 bits.

PT = PT[1] PT[2] ... PT[n]

Each PT block is XORed with a block of the key stream to generate the ciphertext, CT. The AES encryption of each counter block results in 128 bits of key stream.

To construct the counter block, the most significant 48 bits of the counter block are set to the 48 low order bits of the client_write_IV (for the half-duplex stream originated by the client) or the 48 low order bits of the server_write_IV (for the half-duplex stream originated by the server.) The following 64 bits of the counter block are set to record sequence number, and the remaining 16 bits

function as the block counter. The least significant bit of the counter block is initially set to one. This counter value is incremented by one to generate subsequent counter blocks, each resulting in another 128 bits of key stream.

```
struct {
   case client:
       uint48 client_write_IV;  // low order 48-bits
   case server:
       uint48 server_write_IV;  // low order 48-bits
   uint64 seq_num;
   uint16 blk_ctr;
} CtrBlk;
```

The seq_num and blk_ctr fields of the counter block are initialized for each record processed, while the IV is initialized immediately after a key calculation is made (key calculations are made whenver a TLS/DTLS handshake, either full or abbreviated, is executed.) seq_num is set to the sequence number of the record, and blk_ctr is initialized to 1.

Note that the block counter does not overflow since the maximum TLS/

DTLS record size is 14 KB and 16 bits of blk_ctr allow the generation of 1MB of keying material per record.

The encryption of n plaintext blocks can be summarized as:

```
FOR i := 1 to n-1 DO
  CT[i] := PT[i] XOR AES(CtrBlk)
  CtrBlk := CtrBlk + 1
END
CT[n] := PT[n] XOR TRUNC(AES(CtrBlk))
```

The AES() function performs AES encryption with the fresh key.

The TRUNC() function truncates the output of the AES encrypt operation to the same length as the final plaintext block, returning the most significant bits.

Decryption is similar. The decryption of n ciphertext blocks can be summarized as:

```
FOR i := 1 to n-1 DO
  PT[i] := CT[i] XOR AES(CtrBlk)
  CtrBlk := CtrBlk + 1
END
PT[n] := CT[n] XOR TRUNC(AES(CtrBlk))
```


For TLS, no part of the counter block need be transmitted, since the client_write_IV and server_write_IV are derived during the key calculation phase, and the record sequence number is implicit.

3.2. DTLS

The operation of AES-CTR in DTLS is the same as in TLS, with the only difference being the inclusion of the epoch in the counter block. The counter block is constructed as follows for DTLS:

```
struct {
   case client:
       uint48 client_write_IV;  // low order 48-bits
   case server:
       uint48 server_write_IV;  // low order 48-bits
```

```
        uint16 epoch;
        uint48 seq_num;
        uint16 blk_ctr;
    } CtrBlk;
```

The epoch and record sequence number used for generating the counter block are extracted from the received record.

3.3. Padding

Stream ciphers in TLS and DTLS do not require plaintext padding.

3.4. Session Resumption

TLS supports session resumption via caching of session ID's and connection parameters on both client and server. While resumed sessions use the same master secret that was originally negotiated, a resumed session uses new keys that are derived, in part, using fresh client_random and server_random parameters. As a result resumed sessions do not use the same encryption keys or IVs as the original session.

4. Design Rationale

An alternate design for the construction of the counter block would be the use of an explicit 'record tag' (as a substitute for the implicit record sequence number) that could potentially be generated via an LFSR. Such a design, however, suffers two major drawbacks when used in the TLS or DTLS protocol, without offering any significant benefit: (1) in both TLS and DTLS inclusion of such a tag would incur a bandwidth cost, (2) all TLS and DTLS associations have per-record sequence numbers which can be used to ensure counter

uniqueness.

5. Security Considerations

See Section 7. of [2].

5.1. Maximum Key Lifetime

TLS/DTLS sessions employing AES-CTR MUST be renegotiated before sequence numbers repeat. In the case of TLS, this implies a maximum of 2^64 records per session, while for DTLS the maximum is 2^48 (with the remaining bits reserved for epoch.)

6. IANA Considerations

IANA has assigned the following values for AES-CTR mode ciphers:

```
CipherSuite TLS_RSA_WITH_AES_128_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DH_DSS_WITH_AES_128_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DH_RSA_WITH_AES_128_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DHE_DSS_WITH_AES_128_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DHE_RSA_WITH_AES_128_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DH_anon_WITH_AES_128_CTR_SHA = { 0xXX, 0xXX };

CipherSuite TLS_RSA_WITH_AES_256_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DH_DSS_WITH_AES_256_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DH_RSA_WITH_AES_256_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DHE_DSS_WITH_AES_256_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DHE_RSA_WITH_AES_256_CTR_SHA = { 0xXX, 0xXX };
CipherSuite TLS_DH_anon_WITH_AES_256_CTR_SHA = { 0xXX, 0xXX };
```

7. Normative References

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

[2] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, January 2004.

[3] Dierks, T. and E. Rescorla, "The TLS Protocol Version 1.1", draft-ietf-tls-rfc2246-bis-13 (work in progress), June 2005.

Authors' Addresses

Nagendra Modadugu
Stanford University
353 Serra Mall
Stanford, CA 94305
USA

Email: nagendra@cs.stanford.edu

Eric Rescorla
Network Resonance
2483 E. Bayshore Rd., #212
Palo Alto, CA 94303
USA

Email: ekr@networkresonance.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the

Internet Society.