                Using Advanced Encryption Standard (AES) Counter Mode
                 With IPsec Encapsulating Security Payload (ESP)

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.

Abstract

   This document describes the use of Advanced Encryption Standard (AES)
   Counter Mode, with an explicit initialization vector, as an IPsec
   Encapsulating Security Payload (ESP) confidentiality mechanism.

Table of Contents

1. Introduction

   The National Institute of Standards and Technology (NIST) recently
   selected the Advanced Encryption Standard (AES) [AES], also known as
   Rijndael. The AES is a block cipher, and it can be used in many
   different modes. This document describes the use of AES Counter Mode
   (AES-CTR), with an explicit initialization vector (IV), as an IPsec
   Encapsulating Security Payload (ESP) [ESP] confidentiality mechanism.

   This document does not provide an overview of IPsec. However,
   information about how the various components of IPsec and the way in
   which they collectively provide security services is available in
   [ARCH] and [ROADMAP].

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 [STDWORDS].

2. AES Block Cipher

   This section contains a brief description of the relevant
   characteristics of the AES block cipher. Implementation requirements
   are also discussed.

2.1. Counter Mode

   NIST has defined five modes of operation for AES and other FIPS-
   approved block ciphers [MODES]. Each of these modes has different
   characteristics. The five modes are: ECB (Electronic Code Book), CBC
   (Cipher Block Chaining), CFB (Cipher FeedBack), OFB (Output
   FeedBack), and CTR (Counter).

   Only AES Counter mode (AES-CTR) is discussed in this specification.
   AES-CTR requires the encryptor to generate a unique per-packet value,
   and communicate this value to the decryptor. This specification
   calls this per-packet value an initialization vector (IV). The same
   IV and key combination MUST NOT be used more than once. The

encryptor can generate the IV in any manner that ensures uniqueness.
Common approaches to IV generation include incrementing a counter for
each packet and linear feedback shift registers (LFSRs).

This specification calls for the use of a nonce for additional
protection against precomputation attacks. The nonce value need not
be secret. However, the nonce MUST be unpredictable prior to the
establishment of the IPsec security association that is making use of
AES-CTR.

AES-CTR has many properties that make it an attractive encryption
algorithm for in high-speed networking. 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
counter block values. AES-CTR is easy to implement, and AES-CTR can
be pipelined and parallelized. AES-CTR also supports key stream
precomputation.

Pipelining is possible because AES has multiple rounds (see section
2.2). A hardware implementation (and some software implementations)
can create a pipeline by unwinding the loop implied by this round
structure. For example, after a 16-octet block has been input, one
round later another 16-octet block can be input, and so on. In AES-
CTR, these inputs are the sequential counter block values used to
generate the key stream.

Multiple independent AES encrypt implementations can also be used to
improve performance. For example, one could use two AES encrypt
implementations in parallel, to process a sequence of counter block
values, doubling the effective throughput.

The sender can precompute the key stream. Since the key stream does
not depend on any data in the packet, the key stream can be
precomputed once the nonce and IV are assigned. This precomputation
can reduce packet latency. The receiver cannot perform similar
precomputation because the IV will not be known before the packet
arrives.

AES-CTR uses the only AES encrypt operation (for both encryption and
decryption), making AES-CTR implementations smaller than
implementations of many other AES modes.

When used correctly, AES-CTR provides a high level of
confidentiality. Unfortunately, AES-CTR is easy to use incorrectly.
Being a stream cipher, any reuse of the per-packet value, called the
IV, with the same nonce and key is catastrophic. An IV collision
immediately leaks information about the plaintext in both packets.
For this reason, it is inappropriate to use this mode of operation

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. The Internet
Key Exchange (IKE) [IKE] protocol can be used to establish fresh
keys. IKE can also provide the nonce value.

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
function. Implementations MUST use AES-CTR in conjunction with an
authentication function, such as HMAC-SHA-1-96 [HMAC-SHA].

To encrypt a payload with AES-CTR, the encryptor partitions the
plaintext, PT, into 128-bit blocks. The final block need not be 128
bits; it can be less.

    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. The most significant 96 bits of the counter
block are set to the nonce value, which is 32 bits, followed by the
per-packet IV value, which is 64 bits. The least significant 32 bits
of the counter block are 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. The encryption of n
plaintext blocks can be summarized as:

    CTRBLK := NONCE || IV || ONE
    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:

```
CTRBLK := NONCE || IV || ONE
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))
```

2.2. Key Size and Rounds

AES supports three key sizes: 128 bits, 192 bits, and 256 bits. The
default key size is 128 bits, and all implementations MUST support
this key size. Implementations MAY also support key sizes of 192
bits and 256 bits.

AES uses a different number of rounds for each of the defined key
sizes. When a 128-bit key is used, implementations MUST use 10
rounds. When a 192-bit key is used, implementations MUST use 12
rounds. When a 256-bit key is used, implementations MUST use 14
rounds.

2.3. Block Size

The AES has a block size of 128 bits (16 octets). As such, 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. However, to provide limited traffic flow
confidentiality, padding MAY be included, as specified in [ESP].

3. ESP Payload

The ESP payload is comprised of the IV followed by the ciphertext.
The payload field, as defined in [ESP], is structured as shown in
Figure 1.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Initialization Vector                     |
|                          (8 octets)                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                   Encrypted Payload (variable)                ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                   Authentication Data (variable)              ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
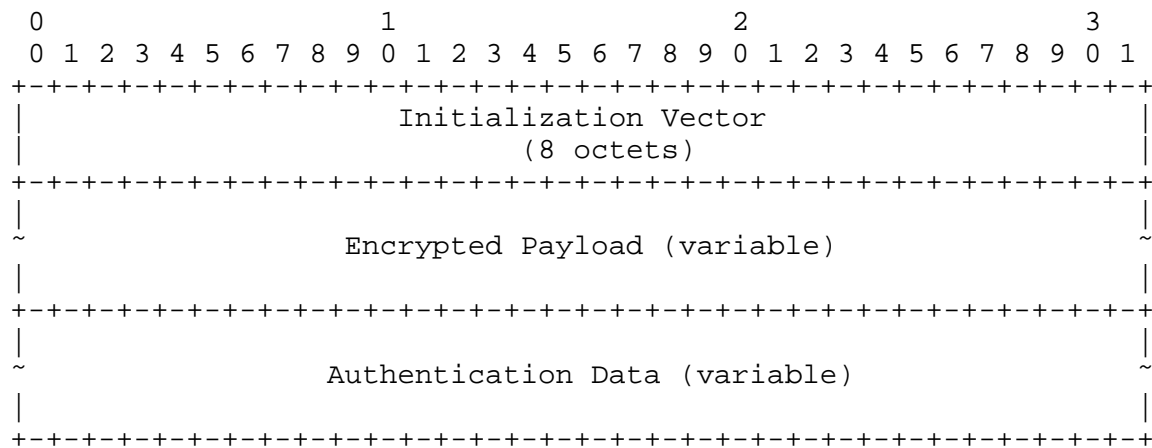
   Figure 1. ESP Payload Encrypted with AES-CTR

3.1. Initialization Vector

   The AES-CTR IV field MUST be eight octets. The IV MUST be chosen by
   the encryptor in a manner that ensures that the same IV value is used
   only once for a given key. The encryptor can generate the IV in any
   manner that ensures uniqueness. Common approaches to IV generation
   include incrementing a counter for each packet and linear feedback
   shift registers (LFSRs).

   Including the IV in each packet ensures that the decryptor can
   generate the key stream needed for decryption, even when some packets
   are lost or reordered.

3.2. Encrypted Payload

   The encrypted payload contains the ciphertext.

   AES-CTR mode does not require plaintext padding. However, ESP does
   require padding to 32-bit word-align the authentication data. The
   padding, Pad Length, and the Next Header MUST be concatenated with
   the plaintext before performing encryption, as described in [ESP].

3.3. Authentication Data

   Since it is trivial to construct a forgery AES-CTR ciphertext from a
   valid AES-CTR ciphertext, AES-CTR implementations MUST employ a non-
   NULL ESP authentication method. HMAC-SHA-1-96 [HMAC-SHA] is a likely
   choice.

4. Counter Block Format

   Each packet conveys the IV that is necessary to construct the
   sequence of counter blocks used to generate the key stream necessary
   to decrypt the payload. The AES counter block cipher block is 128
   bits. Figure 2 shows the format of the counter block.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            Nonce                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Initialization Vector (IV)                  |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Block Counter                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
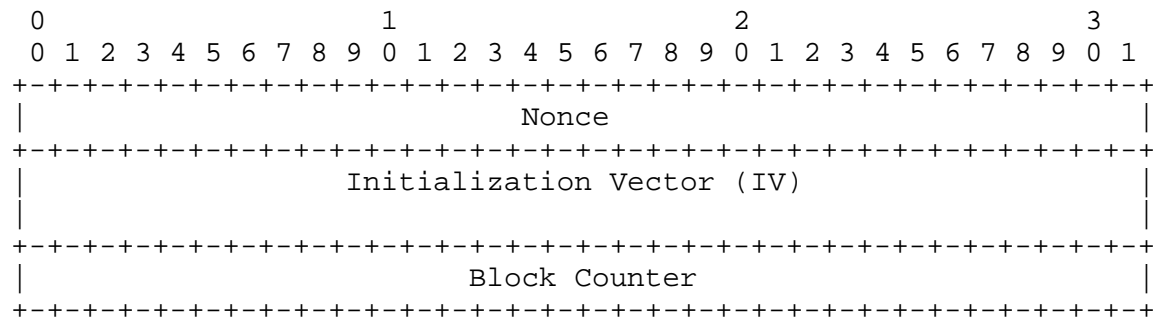
   Figure 2. Counter Block Format

   The components of the counter block are as follows:

   Nonce
      The Nonce field is 32 bits. As the name implies, the nonce is a
      single use value. That is, a fresh nonce value MUST be assigned
      for each security association. It MUST be assigned at the
      beginning of the security association. The nonce value need not
      be secret, but it MUST be unpredictable prior to the beginning of
      the security association.

   Initialization Vector
      The IV field is 64 bits. As described in section 3.1, the IV MUST
      be chosen by the encryptor in a manner that ensures that the same
      IV value is used only once for a given key.

   Block Counter
      The block counter field is the least significant 32 bits of the
      counter block. The block counter begins with the value of one,
      and it is incremented to generate subsequent portions of the key
      stream. The block counter is a 32-bit big-endian integer value.

   Using the encryption process described in section 2.1, this
   construction permits each packet to consist of up to:

      $(2^{32})-1$ blocks = 4,294,967,295 blocks
                       = 68,719,476,720 octets

This construction can produce enough key stream for each packet
sufficient to handle any IPv6 jumbogram [JUMBO].

5. IKE Conventions

This section describes the conventions used to generate keying
material and nonces for use with AES-CTR using the Internet Key
Exchange (IKE) [IKE] protocol. The identifiers and attributes needed
to negotiate a security association which uses AES-CTR are also
defined.

5.1. Keying Material and Nonces

As described in section 2.1, implementations MUST use fresh keys with
AES-CTR. IKE can be used to establish fresh keys. This section
describes the conventions for obtaining the unpredictable nonce value
from IKE. Note that this convention provides a nonce value that is
secret as well as unpredictable.

IKE makes use of a pseudo-random function (PRF) to derive keying
material. The PRF is used iteratively to derive keying material of
arbitrary size, called KEYMAT. Keying material is extracted from the
output string without regard to boundaries.

The size of the requested KEYMAT MUST be four octets longer than is
needed for the associated AES key. The keying material is used as
follows:

AES-CTR with a 128 bit key
   The KEYMAT requested for each AES-CTR key is 20 octets. The first
   16 octets are the 128-bit AES key, and the remaining four octets
   are used as the nonce value in the counter block.

AES-CTR with a 192 bit key
   The KEYMAT requested for each AES-CTR key is 28 octets. The first
   24 octets are the 192-bit AES key, and the remaining four octets
   are used as the nonce value in the counter block.

AES-CTR with a 256 bit key
   The KEYMAT requested for each AES-CTR key is 36 octets. The first
   32 octets are the 256-bit AES key, and the remaining four octets
   are used as the nonce value in the counter block.

5.2. Phase 1 Identifier

   This document does not specify the conventions for using AES-CTR for
   IKE Phase 1 negotiations. For AES-CTR to be used in this manner, a
   separate specification is needed, and an Encryption Algorithm
   Identifier needs to be assigned.

5.3. Phase 2 Identifier

   For IKE Phase 2 negotiations, IANA has assigned an ESP Transform
   Identifier of 13 for AES-CTR with an explicit IV.

5.4. Key Length Attribute

   Since the AES supports three key lengths, the Key Length attribute
   MUST be specified in the IKE Phase 2 exchange [DOI]. The Key Length
   attribute MUST have a value of 128, 192, or 256.

6. Test Vectors

   This section contains nine test vectors, which can be used to confirm
   that an implementation has correctly implemented AES-CTR. The first
   three test vectors use AES with a 128 bit key; the next three test
   vectors use AES with a 192 bit key; and the last three test vectors
   use AES with a 256 bit key.

   Test Vector #1: Encrypting 16 octets using AES-CTR with 128-bit key
   AES Key   : AE 68 52 F8 12 10 67 CC 4B F7 A5 76 55 77 F3 9E
   AES-CTR IV   : 00 00 00 00 00 00 00 00
   Nonce   : 00 00 00 30
   Plaintext String : 'Single block msg'
   Plaintext   : 53 69 6E 67 6C 65 20 62 6C 6F 63 6B 20 6D 73 67
   Counter Block (1): 00 00 00 30 00 00 00 00 00 00 00 00 00 00 00 01
   Key Stream   (1): B7 60 33 28 DB C2 93 1B 41 0E 16 C8 06 7E 62 DF
   Ciphertext   : E4 09 5D 4F B7 A7 B3 79 2D 61 75 A3 26 13 11 B8

   Test Vector #2: Encrypting 32 octets using AES-CTR with 128-bit key
   AES Key   : 7E 24 06 78 17 FA E0 D7 43 D6 CE 1F 32 53 91 63
   AES-CTR IV   : C0 54 3B 59 DA 48 D9 0B
   Nonce   : 00 6C B6 DB
   Plaintext   : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
                     : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
   Counter Block (1): 00 6C B6 DB C0 54 3B 59 DA 48 D9 0B 00 00 00 01
   Key Stream   (1): 51 05 A3 05 12 8F 74 DE 71 04 4B E5 82 D7 DD 87
   Counter Block (2): 00 6C B6 DB C0 54 3B 59 DA 48 D9 0B 00 00 00 02
   Key Stream   (2): FB 3F 0C EF 52 CF 41 DF E4 FF 2A C4 8D 5C A0 37
   Ciphertext   : 51 04 A1 06 16 8A 72 D9 79 0D 41 EE 8E DA D3 88
                     : EB 2E 1E FC 46 DA 57 C8 FC E6 30 DF 91 41 BE 28

```
   Test Vector #3: Encrypting 36 octets using AES-CTR with 128-bit key
   AES Key            : 76 91 BE 03 5E 50 20 A8 AC 6E 61 85 29 F9 A0 DC
   AES-CTR IV         : 27 77 7F 3F  4A 17 86 F0
   Nonce              : 00 E0 01 7B
   Plaintext          : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
                      : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
                      : 20 21 22 23
   Counter Block (1): 00 E0 01 7B 27 77 7F 3F 4A 17 86 F0 00 00 00 01
   Key Stream    (1): C1 CE 4A AB 9B 2A FB DE C7 4F 58 E2 E3 D6 7C D8
   Counter Block (2): 00 E0 01 7B 27 77 7F 3F 4A 17 86 F0 00 00 00 02
   Key Stream    (2): 55 51 B6 38 CA 78 6E 21 CD 83 46 F1 B2 EE 0E 4C
   Counter Block (3): 00 E0 01 7B 27 77 7F 3F 4A 17 86 F0 00 00 00 03
   Key Stream    (3): 05 93 25 0C 17 55 36 00 A6 3D FE CF 56 23 87 E9
   Ciphertext         : C1 CF 48 A8 9F 2F FD D9 CF 46 52 E9 EF DB 72 D7
                      : 45 40 A4 2B DE 6D 78 36 D5 9A 5C EA AE F3 10 53
                      : 25 B2 07 2F


   Test Vector #4: Encrypting 16 octets using AES-CTR with 192-bit key
   AES Key            : 16 AF 5B 14 5F C9 F5 79 C1 75 F9 3E 3B FB 0E ED
                      : 86 3D 06 CC FD B7 85 15
   AES-CTR IV         : 36 73 3C 14 7D 6D 93 CB
   Nonce              : 00 00 00 48
   Plaintext String : 'Single block msg'
   Plaintext          : 53 69 6E 67 6C 65 20 62 6C 6F 63 6B 20 6D 73 67
   Counter Block (1): 00 00 00 48 36 73 3C 14 7D 6D 93 CB 00 00 00 01
   Key Stream    (1): 18 3C 56 28 8E 3C E9 AA 22 16 56 CB 23 A6 9A 4F
   Ciphertext         : 4B 55 38 4F E2 59 C9 C8 4E 79 35 A0 03 CB E9 28


   Test Vector #5: Encrypting 32 octets using AES-CTR with 192-bit key
   AES Key            : 7C 5C B2 40 1B 3D C3 3C 19 E7 34 08 19 E0 F6 9C
                      : 67 8C 3D B8 E6 F6 A9 1A
   AES-CTR IV         : 02 0C 6E AD C2 CB 50 0D
   Nonce              : 00 96 B0 3B
   Plaintext          : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
                      : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
   Counter Block (1): 00 96 B0 3B 02 0C 6E AD C2 CB 50 0D 00 00 00 01
   Key Stream    (1): 45 33 41 FF 64 9E 25 35 76 D6 A0 F1 7D 3C C3 90
   Counter Block (2): 00 96 B0 3B 02 0C 6E AD C2 CB 50 0D 00 00 00 02
   Key Stream    (2): 94 81 62 0F 4E C1 B1 8B E4 06 FA E4 5E E9 E5 1F
   Ciphertext         : 45 32 43 FC 60 9B 23 32 7E DF AA FA 71 31 CD 9F
                      : 84 90 70 1C 5A D4 A7 9C FC 1F E0 FF 42 F4 FB 00
```

```
   Test Vector #6: Encrypting 36 octets using AES-CTR with 192-bit key
   AES Key           : 02 BF 39 1E E8 EC B1 59 B9 59 61 7B 09 65 27 9B
                     : F5 9B 60 A7 86 D3 E0 FE
   AES-CTR IV        : 5C BD 60 27 8D CC 09 12
   Nonce             : 00 07 BD FD
   Plaintext         : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
                     : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
                     : 20 21 22 23
   Counter Block (1) : 00 07 BD FD 5C BD 60 27 8D CC 09 12 00 00 00 01
   Key Stream    (1) : 96 88 3D C6 5A 59 74 28 5C 02 77 DA D1 FA E9 57
   Counter Block (2) : 00 07 BD FD 5C BD 60 27 8D CC 09 12 00 00 00 02
   Key Stream    (2) : C2 99 AE 86 D2 84 73 9F 5D 2F D2 0A 7A 32 3F 97
   Counter Block (3) : 00 07 BD FD 5C BD 60 27 8D CC 09 12 00 00 00 03
   Key Stream    (3) : 8B CF 2B 16 39 99 B2 26 15 B4 9C D4 FE 57 39 98
   Ciphertext        : 96 89 3F C5 5E 5C 72 2F 54 0B 7D D1 DD F7 E7 58
                     : D2 88 BC 95 C6 91 65 88 45 36 C8 11 66 2F 21 88
                     : AB EE 09 35

   Test Vector #7: Encrypting 16 octets using AES-CTR with 256-bit key
   AES Key           : 77 6B EF F2 85 1D B0 6F 4C 8A 05 42 C8 69 6F 6C
                     : 6A 81 AF 1E EC 96 B4 D3 7F C1 D6 89 E6 C1 C1 04
   AES-CTR IV        : DB 56 72 C9 7A A8 F0 B2
   Nonce             : 00 00 00 60
   Plaintext String  : 'Single block msg'
   Plaintext         : 53 69 6E 67 6C 65 20 62 6C 6F 63 6B 20 6D 73 67
   Counter Block (1) : 00 00 00 60 DB 56 72 C9 7A A8 F0 B2 00 00 00 01
   Key Stream    (1) : 47 33 BE 7A D3 E7 6E A5 3A 67 00 B7 51 8E 93 A7
   Ciphertext        : 14 5A D0 1D BF 82 4E C7 56 08 63 DC 71 E3 E0 C0

   Test Vector #8: Encrypting 32 octets using AES-CTR with 256-bit key
   AES Key           : F6 D6 6D 6B D5 2D 59 BB 07 96 36 58 79 EF F8 86
                     : C6 6D D5 1A 5B 6A 99 74 4B 50 59 0C 87 A2 38 84
   AES-CTR IV        : C1 58 5E F1 5A 43 D8 75
   Nonce             : 00 FA AC 24
   Plaintext         : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
                     : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
   Counter block (1) : 00 FA AC 24 C1 58 5E F1 5A 43 D8 75 00 00 00 01
   Key stream    (1) : F0 5F 21 18 3C 91 67 2B 41 E7 0A 00 8C 43 BC A6
   Counter block (2) : 00 FA AC 24 C1 58 5E F1 5A 43 D8 75 00 00 00 02
   Key stream    (2) : A8 21 79 43 9B 96 8B 7D 4D 29 99 06 8F 59 B1 03
   Ciphertext        : F0 5E 23 1B 38 94 61 2C 49 EE 00 0B 80 4E B2 A9
                     : B8 30 6B 50 8F 83 9D 6A 55 30 83 1D 93 44 AF 1C
```

```
Test Vector #9: Encrypting 36 octets using AES-CTR with 256-bit key
AES Key            : FF 7A 61 7C E6 91 48 E4 F1 72 6E 2F 43 58 1D E2
                   : AA 62 D9 F8 05 53 2E DF F1 EE D6 87 FB 54 15 3D
AES-CTR IV         : 51 A5 1D 70 A1 C1 11 48
Nonce              : 00 1C C5 B7
Plaintext          : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
                   : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
                   : 20 21 22 23
Counter block (1): 00 1C C5 B7 51 A5 1D 70 A1 C1 11 48 00 00 00 01
Key stream    (1): EB 6D 50 81 19 0E BD F0 C6 7C 9E 4D 26 C7 41 A5
Counter block (2): 00 1C C5 B7 51 A5 1D 70 A1 C1 11 48 00 00 00 02
Key stream    (2): A4 16 CD 95 71 7C EB 10 EC 95 DA AE 9F CB 19 00
Counter block (3): 00 1C C5 B7 51 A5 1D 70 A1 C1 11 48 00 00 00 03
Key stream    (3): 3E E1 C4 9B C6 B9 CA 21 3F 6E E2 71 D0 A9 33 39
Ciphertext         : EB 6C 52 82 1D 0B BB F7 CE 75 94 46 2A CA 4F AA
                   : B4 07 DF 86 65 69 FD 07 F4 8C C0 B5 83 D6 07 1F
                   : 1E C0 E6 B8
```

7. Security Considerations

   When used properly, AES-CTR mode provides strong confidentiality.
   Bellare, Desai, Jokipii, Rogaway show in [BDJR] that the privacy
   guarantees provided by counter mode are at least as strong as those
   for CBC mode when using the same block cipher.

   Unfortunately, it is very easy to misuse this counter mode. If
   counter block values are ever used for more that one packet with the
   same key, then the same key stream will be used to encrypt both
   packets, and the confidentiality guarantees are voided.

   What happens if the encryptor XORs the same key stream with two
   different plaintexts? Suppose two plaintext byte 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.

Therefore, stream ciphers, including AES-CTR, should not be used with
static keys. It is inappropriate to use AES-CTR with static keys.
Extraordinary measures would be needed to prevent reuse of a counter
block value with the static key across power cycles. To be safe, ESP
implementations MUST use fresh keys with AES-CTR. The Internet Key
Exchange (IKE) protocol [IKE] can be used to establish fresh keys.
IKE can also be used to establish the nonce at the beginning of the
security association.

When IKE is used to establish fresh keys between two peer entities,
separate keys are established for the two traffic flows. When a
mechanism other than IKE is used to establish fresh keys, and that
mechanism establishes only a single key to encrypt packets, then
there is a high probability that the peers will select the same IV
values for some packets. Thus, to avoid counter block collisions,

ESP implementations that permit use of the same key for encrypting
outbound traffic and decrypting incoming traffic with the same peer
MUST ensure that the two peers assign different Nonce values to the
security association.

Data forgery is trivial with CTR mode. The demonstration of this
attack is similar to the key stream reuse discussion above. If a
known plaintext byte sequence P1, P2, P3 is encrypted with key stream
K1, K2, K3, then the attacker can replace the plaintext with one of
his 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)

Accordingly, ESP implementations MUST use of AES-CTR in conjunction
with ESP authentication.

Additionally, 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. Since ESP with Enhanced Sequence Numbers allows
for up to 2^64 packets in a single security association, there is
real potential for more than 2^64 blocks to be encrypted with one
key. Therefore, implementations SHOULD generate a fresh key before
2^64 blocks are encrypted with the same key. Note that ESP with 32-
bit Sequence Numbers will not exceed 2^64 blocks even if all of the
packets are maximum-length IPv6 jumbograms [JUMBO].

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
any other block cipher mode) 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. Therefore, implementations
that employ 128-bit AES keys should take precautions to make the
precomputation attacks more difficult. The unpredictable nonce value
in the counter block significantly increases the size of the table
that the attacker must compute to mount a successful attack.

8. Design Rationale

In the development of this specification, the use of the ESP sequence
number field instead of an explicit IV field was considered. This
selection is not a cryptographic security issue, as either approach
will prevent counter block collisions.

In a very conservative model of encryption security, at most 2^64
blocks ought to be encrypted with AES-CTR under a single key. Under
this constraint, no more than 64 bits are needed to identify each
packet within a security association. Since the ESP extended
sequence number is 64 bits, it is an obvious candidate for use as an
implicit IV. This would dictate a single method for the assignment
of per-packet value in the counter block. The use of an explicit IV
does not dictate such a method, which is desirable for several
reasons.

1. Only the encryptor can ensure that the value is not used for more
   than one packet, so there is no advantage to selecting a mechanism
   that allows the decryptor to determine whether counter block
   values collide. Damage from the collision is done, whether the
   decryptor detects it or not.

2. Allows adders, LFSRs, and any other technique that meets the time
   budget of the encryptor, so long as the technique results in a
   unique value for each packet. Adders are simple and
   straightforward to implement, but due to carries, they do not
   execute in constant time. LFSRs offer an alternative that
   executes in constant time.

3. Complexity is in control of the implementer. Further, the
   decision made by the implementer of the encryptor does not make
   the decryptor more (or less) complex.

4. When the encryptor has more than one cryptographic hardware
   device, an IV prefix can be assigned to each device, ensuring that
   collisions will not occur. Yet, since the decryptor does not need
   to examine IV structure, the decryptor is unaffected by the IV
   structure selected by the encryptor. One cannot make use of the
   same technique with the ESP sequence numbers, because the
   semantics for them require sequential value generation.

5. Assurance boundaries are very important to implementations that
   will be evaluated against the FIPS Pub 140-1 or FIPS Pub 140-2
   [SECRQMTS]. The assignment of the per-packet counter block value
   needs to be inside the assurance boundary. Some implementations
   assign the sequence number inside the assurance boundary, but
   others do not. A sequence number collision does not have the dire
   consequences, but, as described in section 6, a collision in
   counter block values has disastrous consequences.

6. Coupling with the sequence number is possible in those
   architectures where the sequence number assignment is performed
   within the assurance boundary. In this situation, the sequence
   number and the IV field will contain the same value.

7. Decoupling from the sequence number is possible in those
   architectures where the sequence number assignment is performed
   outside the assurance boundary.

The use of an explicit IV field directly follows from the decoupling
of the sequence number and the per-packet counter block value. The
overhead associated with 64 bits for the IV field is acceptable.
This overhead is significantly less than the overhead associated with
Cipher Block Chaining (CBC) mode. As normally employed, CBC requires

a full block for the IV and, on average, half of a block for padding.
AES-CTR with an explicit IV has about one-third of the overhead as
AES-CBC, and the overhead is constant for each packet.

The inclusion of the nonce provides a weak countermeasure against
precomputation attacks. For this countermeasure to be effective, the
attacker must not be able to predict the value of the nonce well in
advance of security association establishment. The use of long keys
provides a strong countermeasure to precomputation attacks, and AES
offers key sizes that thwart these attacks for many decades to come.

A 28-bit block counter value is sufficient for the generation of a
key stream to encrypt the largest possible IPv6 jumbogram [JUMBO];
however, a 32-bit field is used. This size is convenient for both
hardware and software implementations.

9. IANA Considerations

   IANA has assigned 13 as the ESP transform number for AES-CTR with an
   explicit IV.

10. Intellectual Property Statement

   The IETF takes no position regarding the validity or scope of any
   intellectual property 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; neither does it represent that it
   has made any effort to identify any such rights. Information on the
   IETF's procedures with respect to rights in standards-track and
   standards-related documentation can be found in BCP-11. Copies of
   claims of rights made available for publication 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 implementors or users of this specification can
   be obtained from the IETF Secretariat.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights which may cover technology that may be required to practice
   this standard. Please address the information to the IETF Executive
   Director.

11. Acknowledgements

   This document is the result of extensive discussions and compromises.
   While not all of the participants are completely satisfied with the
   outcome, the document is better for their contributions.

The author thanks the members of the IPsec working group for their
contributions to the design, with special mention of the efforts of
(in alphabetical order) Steve Bellovin, David Black, Niels Ferguson,
Charlie Kaufman, Steve Kent, Tero Kivinen, Paul Koning, David McGrew,
Robert Moskowitz, Jesse Walker, and Doug Whiting.

The author thanks and Alireza Hodjat, John Viega, and Doug Whiting
for assistance with the test vectors.

12. References

This section provides normative and informative references.

12.1. Normative References

[AES]   NIST, FIPS PUB 197, "Advanced Encryption Standard (AES)",
               November 2001.

[DOI]   Piper, D., "The Internet IP Security Domain of
               Interpretation for ISAKMP", RFC 2407, November 1998.

[ESP]   Kent, S. and R. Atkinson, "IP Encapsulating Security
               Payload (ESP)", RFC 2406, November 1998.

[MODES]   Dworkin, M., "Recommendation for Block Cipher Modes of
               Operation: Methods and Techniques", NIST Special
               Publication 800-38A, December 2001.

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

12.2. Informative References

[ARCH]   Kent, S. and R. Atkinson, "Security Architecture for the
               Internet Protocol", RFC 2401, November 1998.

[BDJR]   Bellare, M, Desai, A., Jokipii, E. and P. Rogaway, "A
               Concrete Security Treatment of Symmetric Encryption:
               Analysis of the DES Modes of Operation", Proceedings 38th
               Annual Symposium on Foundations of Computer Science,
               1997.

[HMAC-SHA]   Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within
               ESP and AH", RFC 2404, November 1998.

[IKE]   Harkins, D. and D. Carrel, "The Internet Key Exchange
               (IKE)", RFC 2409, November 1998.

    [JUMBO]   Borman, D., Deering, S. and R. Hinden, "IPv6 Jumbograms",
                  RFC 2675, August 1999.

    [ROADMAP]   Thayer, R., Doraswamy, N. and R. Glenn, "IP Security
                  Document Roadmap", RFC 2411, November 1998.

    [SECRQMTS]   National Institute of Standards and Technology. FIPS Pub
                  140-1: Security Requirements for Cryptographic Modules.
                  11 January 1994.

                  National Institute of Standards and Technology. FIPS Pub
                  140-2: Security Requirements for Cryptographic Modules.
                  25 May 2001. [Supercedes FIPS Pub 140-1]

13. Author's Address

    Russell Housley
    Vigil Security, LLC
    918 Spring Knoll Drive
    Herndon, VA 20170
    USA

    EMail: housley@vigilsec.com

14. Full Copyright Statement

Acknowledgement