IEEE P802.11
Wireless LANs

AES Encryption & Authentication Using CTR Mode & CBC-MAC

Date: January 15, 2002

Authors:

Doug Whiting
Hifn
5973 Avenida Encinas, #110
Carlsbad, CA 92009
Phone: +1 760-827-4502
E-mail: DWhiting@hifn.com

Russ Housley
RSA Laboratories
918 Spring Knoll Drive
Herndon, Virginia 20170
Phone: +1 703-435-1775
E-mail: RHousley@rsasecurity.com

and

Niels Ferguson
MacFergus BV
Bart de Ligtstraat 64
1097 JE Amsterdam, Netherlands
Phone: +31 20 643 0977
E-mail: Niels@ferguson.net

Abstract

Use of the AES is desired as part of the 802.11i, the Specification for Enhanced Security. A combination of counter mode encryption and CBC-MAC authentication is proposed here. These modes have been used and studied for a long time, with well-understood cryptographic properties, and no known patent encumbrances. They provide for good security and performance, whether implemented in hardware or software. Sample source code and test vectors are provided.

1 Motivation

IEEE Std 802.11-1999 is the wireless LAN standard. The new AES has many properties are desirable for securing traffic on wireless networks. However, any encryption algorithm can be used in many different ways, each of which will have different performance and security characteristics. Important criteria for comparison of cryptographic algorithms and protocols include:

- Patent Status
- Size of Implementation
- Power Consumption
- Speed
- Cleartext Integrity Coverage
- Simplicity of Key Management
- Packet Overhead
- Crypto Confidence

These criteria are discussed in a separate presentation (IEEE 802.11-01/634r1). In that presentation, the use of OCB mode is compared to this proposal, called CTR+CBC-MAC. The purpose of this document is to completely specify CTR+CBC-MAC, allowing the transforms to be evaluated and implemented with confidence.

2 *Definitions*

The following variables are used in the description of this cryptographic mode, as discussed below.

Name	Description	Field Size	Comments
C	Block number "counter" within a packet	16 bits	
H	Number of cleartext header octets covered by MIC	8 bits	Up to 255 header octets allowed
L	Length of the input packet	16 bits	Packet size up to 64K octets
M	Size (in octets) of MIC "tag"	5 bits	Recommended value: M = 8
N	Unique packet number	32 bits	Similar to the WEP IV
P	Input packet	L octets	Octets p[0], p[1], … , p[L-1]
A	Authentication (MIC) value, computed over P	M octets	Octets a[0], a[1], … , a[M-1]
Q	Output packet	L+M octets	Octets q[0], q[1], … , q[L+M–1]
RA	Receiver address	48 bits	Octets RA[0], … , RA[5]
TA	Transmitter address	48 bits	Octets TA[0], … , TA[5]
K	AES Encryption Key	128 bits	Octets k[0], k[1], … , k[15]

All key material for a given connection will be derived from mechanisms outside the scope of this document. In general, each pair of stations in the network that desire to communicate will have a unique pairwise key. A new key will also need to be negotiated periodically. This proposal uses AES with a 128-bit key and 128-bit block size.

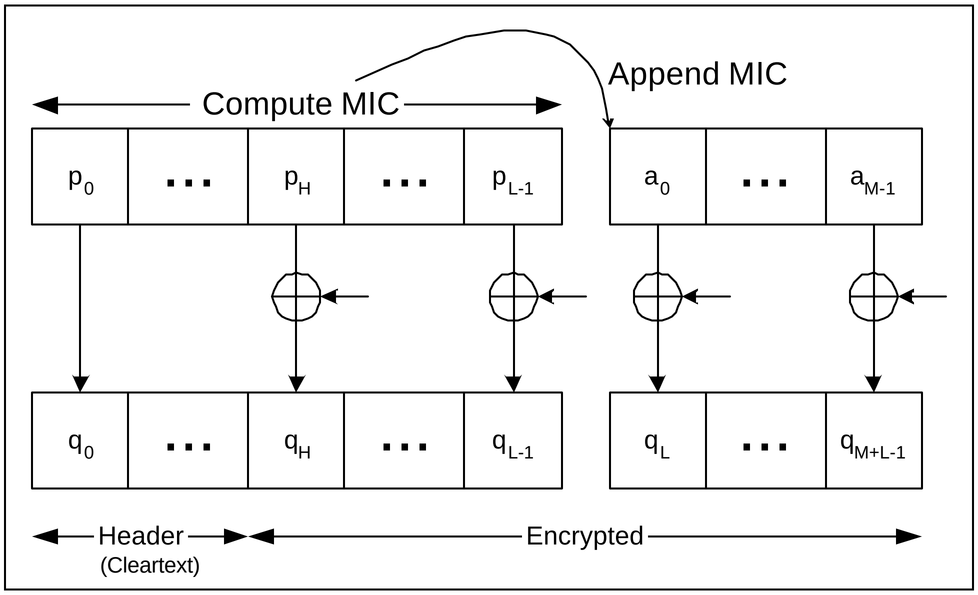

Figure 1. Encryption Process

Before encryption, a packet P has a length of L octets. Of these octets, the first H ≤ L octets (i.e., the "header") are to be covered by authentication but not encrypted. In other words, all the octets in the input packet are covered by CBC-MAC authentication, while only the input data octets p[H], p[H+1] … p[L–1] are encrypted. If any octets in the physical packet are not to be covered by either authentication or encryption, for purposes of this discussion, they are not included in the packet P. Any mutable fields (e.g., the packet "re-transmit" bit) must not be covered by the CBC-MAC or the encryption, in order to avoid having re-encryption.

The length of the MIC authentication value, A, is M octets, and it is generally fixed for each connection. We propose that M=8 octets (64 bits) be used for 802.11i.

To encode a packet, the MIC value is computed over all the octets of the input packet P, and the M octets of MIC value are appended to P. Then the L–H+M octets p[H], p[H+1], … , p[L–1], a[0], a[1], … , a[M–1] are encrypted to produce the output packet Q. In other words, q[j] = p[j] for j=0 to H–1.

The notation <a,b,c,d> = W, where W is a 32-bit integer value, means that the octets a,b,c,d are assigned to the octets of the integer W, in big-endian order. In other words, using C syntax:

```
a = (W >> 24) & 0xFF;
b = (W >> 16) & 0xFF;
c = (W >>  8) & 0xFF;
d =  W        & 0xFF;
```

3 *CBC-MAC Computation (MIC)*

For purposes of computing the MIC value using CBC-MAC, the packet data is logically padded with zeroes, if needed, up to a multiple of the AES block size (i.e., 16 octets). In other words, the values p[k] are assumed to be zero for all k ≥ L. The first AES block of the MAC computation is used to generate the CBC IV. It does not cover any of the packet data, but instead includes the values shown below in Table 1~~Table 1~~.

Field	Description	Size
RA	"Low" bits of receiver address (octets RA[4] and RA[5])	16 bits
TA	Transmitter address (octets TA[0] to TA[5])	48 bits
N	Unique packet number	32 bits
H	Number of cleartext header octets covered by MIC	8 bits
000	Indicates CBC-MAC, not encryption	3 bits
M	Size (in octets) of MIC "tag"	5 bits
L	Length (in octets) of the input packet	16 bits

Table 1. Bit Fields of CBC-MAC IV Input Block

These values are used in network order (big-endian) in the AES block. In other words, if we define the plaintext of this first encryption as the octets t[j] for j=0 to 15, this plaintext block is generated as follows, using a C-like pseudo-code syntax:

```
<t[ 0],t[ 1],t[ 2],t[ 3]> = <RA[4],RA[5],TA[0],TA[1]>;
<t[ 4],t[ 5],t[ 6],t[ 7]> = <TA[2],TA[3],TA[4],TA[5]>;
<t[ 8],t[ 9],t[10],t[11]> = N;      /* packet sequence number */
<t[12],t[13],t[14],t[15]> = (H << 24) + (M << 16) + L;
```

This plaintext block is then encrypted, using AES with key K, to produce the CBC IV for the remainder of the (padded) packet data:

```
<t[0],t[1], … ,t[15] > = AES_Encrypt(K,<t[0],t[1], … ,t[15]>);
```

Now, CBC-MAC proceeds over the plaintext data as follows:

```
for (i=0;i<L;i+=16)
  {
  for (j=0;j<16;j++) t[j]^= p[i+j]; /* zero pad p[] if necessary */
  <t[0],t[1], … ,t[15]>   = AES_Encrypt(K,<t[0],t[1], … ,t[15]>);
  }
```

Using the final AES encryption result, the MAC value, A, is then defined by a[j] = t[j], for j=0 to M–1. These M octets of MIC are then appended to the packet data by p[L+j] = a[j], for j=0 to M–1.

4 *Counter Mode Encryption*

The first step in counter mode encryption consists of setting the counter plaintext “template” similar to the first step of the CBC-MAC, as follows:

Field	Description	Size
RA	“Low” bits of receiver address (octets RA[4] to RA[5])	16 bits
TA	Transmitter address (octets TA[0] to TA[5])	48 bits
N	Unique packet number	32 bits
H	Number of cleartext header octets covered by MIC	8 bits
100	Indicates encryption, not CBC-MAC	3 bits
M	Size (in octets) of MIC “tag”	5 bits
C	Block number “counter” within the packet	16 bits

Table 2~~2~~. Bit Fields of Counter Mode Encryption Counter

In other words,

```
<t[ 0],t[ 1],t[ 2],t[ 3]> = <RA[4],RA[5],TA[0],TA[1]>;
<t[ 4],t[ 5],t[ 6],t[ 7]> = <TA[2],TA[3],TA[4],TA[5]>;
<t[ 8],t[ 9],t[10],t[11]> = N;      /* packet sequence number */
<t[12],t[13],t[14],t[15]> = (H << 24) + ((M+0x80) << 16);
```

Then the packet data, including the MIC, is encrypted, using the following steps:

```
for (i=0;i<L+M-H;i+=16)
  {
  C = i/16;       /* block counter C */
  t[14] = C/256;
  t[15] = C%256;
  <u[0],u[1], … ,u[15]>   = AES_Encrypt(K,<t[0],t[1], … ,t[15]>);
  for (j= 0;j<16;j++)
     if (i+j < L+M-H)
        p[H+i+j] ^= u[j];
  }
```

5 *Implementation Notes and Options*

Note that the upper 64 bits of the AES input blocks do not change during the lifetime of the connection. The purpose of these “salt” bits is to minimize the impact of certain classes of attacks (e.g., Hellman80, Biham96). For 802.11i, we have chosen them as addresses. Another possible approach is to choose these salt bits at random when the connection is established, possibly by hashing connection attributes, such as the receiver/transmitter addresses. These salt bits do not have to remain secret.

6 *Security Considerations*

Jakob Jonsson from RSA Laboratories is working on a security proof of CTR+CBC-MAC. While the proof is not finished yet, initial results indicate that CTR+CBC-MAC has roughly the same security bound as OCB mode.

Appendix A. Test Vectors

```
Borland C compiler [Jan 15 2002 14:21:05].
Random seed = 1011133266
AES KAT Vectors:
Key:        00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
PT:         80 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
CT:         3A D7 8E 72  6C 1E C0 2B  7E BF E9 2B  23 D9 EC 34

Key:        80 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
PT:         00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
CT:         0E DD 33 D3  C6 21 E5 46  45 5B D8 BA  14 18 BE C8

NIST AES Vectors: OK
=============== Packet Vector #1 ==================
AES Key:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000000
      RA = 40 41 42 43  44 45
Packet length =   31. [Input (8 cleartext header octets)]
            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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 00  08 08 00 1F
CBC IV out:92 53 8F B6  05 8A 6B 64  70 33 6D 29  78 45 29 B9
CBC[0000]: 31 44 F3 5A  7D 6F 98 9E  38 01 ED C7  BB F3 9D 35
CBC[0010]: 97 5C 94 C1  88 5B 0B 3B  36 9A 21 09  DE D0 76 2C
Packet length =   39. [MIC appended (8 octets)]
            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 97
            5C 94 C1 88  5B 0B 3B
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 00  08 88 00 00
CTR[0001]: 38 5F 10 73  E4 B7 C5 13  0C B6 93 EB  60 B6 12 AB
CTR[0002]: EE FF 8C 88  01 DC DE E1  4D 83 A1 00  30 46 2C 02
Packet length =   39. [Encrypted]
            00 01 02 03  04 05 06 07  30 56 1A 78  E8 BA CB 1C
            1C A7 81 F8  74 A3 04 BC  F6 E6 96 93  1D C1 C0 76
            11 17 60 88  6B 4D 17
=============== Packet Vector #2 ==================
AES Key:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000001
      RA = 40 41 42 43  44 45
Packet length =   32. [Input (8 cleartext header octets)]
            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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 01  08 08 00 20
CBC IV out:B8 0E 48 EB  93 39 61 8B  A7 D9 38 62  31 7D E8 67
CBC[0000]: E9 C0 09 91  34 E1 71 5F  3B BC A5 38  73 E4 77 0B
CBC[0010]: 17 1D D1 9F  60 2D 86 7B  B9 B0 A4 36  D1 FD 6D E0
Packet length =   40. [MIC appended (8 octets)]
            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
            17 1D D1 9F  60 2D 86 7B
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 01  08 88 00 00
CTR[0001]: F3 30 03 80  D3 20 A6 3B  2D C0 00 26  7C 8E 47 5C
CTR[0002]: F3 D4 6F C3  BB 42 8C FA  C6 08 88 F7  6C A9 55 F6
Packet length =   40. [Encrypted]
            00 01 02 03  04 05 06 07  FB 39 09 8B  DF 2D A8 34
            3D D1 12 35  68 9B 51 4B  EB CD 75 D8  A7 5F 92 E5
            D1 15 59 68  0C 84 D3 8D
=============== Packet Vector #3 ==================
AES Key:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000002
      RA = 40 41 42 43  44 45
Packet length =   33. [Input (8 cleartext header octets)]
            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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 02  08 08 00 21
CBC IV out:D8 E1 CA 18  B1 42 7D D2  AA 8D CF 4F  ED DE 91 FE
CBC[0000]: A8 9B 62 44  92 B3 0A 20  40 CF 44 47  68 28 AB DF
CBC[0010]: 35 88 5F D9  67 F4 E2 BC  0D 7E DC 1C  1E B7 22 1A
CBC[0020]: 58 97 6F 9E  BD DB C5 6E  E6 57 E3 4F  3C 9F D0 F5
Packet length =   41. [MIC appended (8 octets)]
            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 58 97 6F  9E BD DB C5  6E
```

```
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 02  08 88 00 00
CTR[0001]: 66 D3 2A BB  26 21 AD 48  31 AD 32 45  7C 7E CD C0
CTR[0002]: 2C 0C B1 5E  90 DB 51 66  8E CF 4D D2  8A 22 66 8F
CTR[0003]: 44 2D 2D 59  8F CE 04 99  BF B1 70 CA  76 BE A1 8A
Packet length =   41. [Encrypted]
           00 01 02 03  04 05 06 07  6E DA 20 B0  2A 2C A3 47
           21 BC 20 56  68 6B DB D7  34 15 AB 45  8C C6 4F 79
           AE 97 DA BD  14 9F BD 4A  2A
=============== Packet Vector #4 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000003
      RA = 40 41 42 43  44 45
Packet length =   31. [Input (12 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 03  0C 08 00 1F
CBC IV out:B6 27 27 8B  F7 B8 61 88  A9 73 85 81  DC DB E1 CB
CBC[0000]: D4 C8 65 D9  0A A7 CF 80  2D 02 92 44  DC C0 8E 5A
CBC[0010]: CB 8F 38 A7  EC D7 85 56  79 1F 4D 96  69 6C F5 80
Packet length =   39. [MIC appended (8 octets)]
           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 CB
           8F 38 A7 EC  D7 85 56
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 03  0C 88 00 00
CTR[0001]: 38 4D E7 0C  AF 1A 93 A5  BD F8 6F 57  C1 C4 8A A0
CTR[0002]: AB 24 96 99  25 F4 3E 68  75 AC 5B AF  9C AE 0C F2
Packet length =   39. [Encrypted]
           00 01 02 03  04 05 06 07  08 09 0A 0B  34 40 E9 03
           BF 0B 81 B6  A9 ED 79 40  D9 DD 90 BB  B7 39 88 52
           AA CC 99 84  A2 29 0D
=============== Packet Vector #5 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000004
      RA = 40 41 42 43  44 45
Packet length =   32. [Input (12 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 04  0C 08 00 20
CBC IV out:07 F6 59 95  05 3D EA DA  6D E3 35 16  59 41 C0 37
CBC[0000]: 32 39 EB BD  DB D8 9D A2  9B 26 AD 0E  28 1C D9 2F
CBC[0010]: 51 86 16 AB  91 49 F4 87  88 76 4F 82  A4 CE DB 16
Packet length =   40. [MIC appended (8 octets)]
           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
           51 86 16 AB  91 49 F4 87
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 04  0C 88 00 00
CTR[0001]: 0F 5C 40 9F  7A 29 AF 74  0F 67 F7 EE  F0 EB 3A 9C
CTR[0002]: 13 F6 DB 3F  9A AE 53 58  9B 4A 27 05  70 A2 01 C4
Packet length =   40. [Encrypted]
           00 01 02 03  04 05 06 07  08 09 0A 0B  03 51 4E 90
           6A 38 BD 67  1B 72 E1 F9  E8 F2 20 87  0F EB C5 20
           CB 28 45 F3  0A 03 D3 82
=============== Packet Vector #6 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000005
      RA = 40 41 42 43  44 45
Packet length =   33. [Input (12 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 05  0C 08 00 21
CBC IV out:88 E5 92 F7  CD BA C6 FF  71 1B 8A 16  F0 B8 80 52
CBC[0000]: D7 D4 66 C8  A7 2B 25 0E  19 25 88 39  0F 29 6B 60
CBC[0010]: 62 F1 0B A4  FB 16 D6 BD  37 2C BF B7  7A 41 47 5E
CBC[0020]: 22 DD 11 82  F3 F3 71 6A  B3 B3 D0 57  CE A3 AB 94
Packet length =   41. [MIC appended (8 octets)]
           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 22 DD 11  82 F3 F3 71  6A
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 05  0C 88 00 00
CTR[0001]: A0 FC 14 C2  CF 2A 49 19  B7 34 9D BB  67 D9 E0 5D
CTR[0002]: 9C 04 BC FB  B6 65 D8 C5  34 57 23 2D  B3 93 D9 DA
Packet length =   41. [Encrypted]
           00 01 02 03  04 05 06 07  08 09 0A 0B  AC F1 1A CD
           DF 3B 5B 0A  A3 21 8B AC  7F C0 FA 46  80 19 A2 E4
           96 47 05 D4  B6 A4 D0 5C  D9
=============== Packet Vector #7 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000006
      RA = 40 41 42 43  44 45
Packet length =   31. [Input (8 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 06  08 0A 00 1F
CBC IV out:6A 4D BD 4A  C7 AB C8 15  1E 52 1F 05  89 85 15 0D
CBC[0000]: 9B DC 54 AC  81 BC FE F2  35 71 97 A1  25 DF BF 6E
CBC[0010]: 2B 1F 2C D1  81 BF BC F3  82 92 00 3A  36 A3 F3 35
Packet length =   41. [MIC appended (10 octets)]
           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 2B
           1F 2C D1 81  BF BC F3 82  92
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 06  08 8A 00 00
CTR[0001]: 19 28 23 4E  C2 4C F4 AD  BA 20 97 0D  5E 49 AC 12
CTR[0002]: AC 2D 02 A6  18 93 BD 0B  FC F6 56 95  12 46 4B 91
CTR[0003]: FE 86 CB C9  55 C6 C2 9F  CB 4F F5 A2  BD B1 B5 85
Packet length =   41. [Encrypted]
           00 01 02 03  04 05 06 07  11 21 29 45  CE 41 FA A2
           AA 31 85 1E  4A 5C BA 05  B4 34 18 BD  04 8E A3 20
           E3 DA 87 14  AD FA B8 13  6C
=============== Packet Vector #8 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000007
      RA = 40 41 42 43  44 45
Packet length =   32. [Input (8 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 07  08 0A 00 20
CBC IV out:BA 3A 15 3B  D5 C4 BC A6  B5 D5 80 1F  1E F2 21 6E
CBC[0000]: FE CC A4 4E  DB 0E E7 E6  F1 5E 4C 96  21 A1 40 92
CBC[0010]: 7C F6 38 9B  D8 15 CF 70  BC 3F 95 02  68 76 51 40
Packet length =   42. [MIC appended (10 octets)]
           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
           7C F6 38 9B  D8 15 CF 70  BC 3F
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 07  08 8A 00 00
CTR[0001]: 29 F2 B0 65  0B 6C CB 4D  DB 72 F8 E4  DD 25 AF AC
CTR[0002]: 17 90 34 43  EC EB C1 36  CC F2 3E EE  C9 9D 45 7B
CTR[0003]: B8 CB 13 A6  25 6C 6F 67  4B D8 40 01  3F 6E 06 24
Packet length =   42. [Encrypted]
           00 01 02 03  04 05 06 07  21 FB BA 6E  07 61 C5 42
           CB 63 EA F7  C9 30 B9 BB  0F 89 2E 58  F0 F6 DF 29
           B0 04 06 75  11 88 8A 0B  04 F4
=============== Packet Vector #9 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000008
      RA = 40 41 42 43  44 45
Packet length =   33. [Input (8 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 08  08 0A 00 21
CBC IV out:72 86 4A EC  B1 7C DA 26  F9 86 9A FB  37 AD D7 8A
CBC[0000]: 51 20 24 77  8B E3 18 C1  BE 77 BA 21  0A 1B 89 F8
CBC[0010]: 68 2D A0 5A  0D D5 02 43  6B 73 C7 1C  38 BA A9 20
CBC[0020]: B6 C5 45 DB  A2 20 8C D7  32 7D 61 D0  11 02 ED 6B
Packet length =   43. [MIC appended (10 octets)]
           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 B6 C5 45  DB A2 20 8C  D7 32 7D
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 08  08 8A 00 00
CTR[0001]: 08 A8 1A FD  2E 91 80 F5  79 D1 DF 85  B7 FC 1F C9
CTR[0002]: 71 73 06 F5  97 36 28 BB  89 A3 BC 97  B0 83 DE DE
CTR[0003]: 95 04 6C 31  CD FB F1 AC  94 90 A2 C4  AF 42 DE 93
Packet length =   43. [Encrypted]
           00 01 02 03  04 05 06 07  00 A1 10 F6  22 9C 8E FA
           69 C0 CD 96  A3 E9 09 DE  69 6A 1C EE  8B 2B 36 A4
           A9 15 79 D2  6B 21 FE 52  42 36 11
=============== Packet Vector #10 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 00000009
      RA = 40 41 42 43  44 45
Packet length =   31. [Input (12 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 09  0C 0A 00 1F
CBC IV out:3D F8 F0 B1  3D 9B 26 E1  62 98 04 72  1C 89 44 8A
CBC[0000]: E0 AB 2E C6  77 73 AE EB  2F 3D 56 B4  24 E5 B5 0E
CBC[0010]: 66 7A 8B EE  43 3F 86 53  37 6B AE 00  23 2C 7E ED
Packet length =   41. [MIC appended (10 octets)]
           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 66
           7A 8B EE 43  3F 86 53 37  6B
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 09  0C 8A 00 00
CTR[0001]: DF 69 65 EF  26 C0 BF 52  5F 60 E0 BC  05 46 74 12
CTR[0002]: A4 07 B7 F2  0B CE D1 54  8E 4A 1E A0  60 2E B0 7F
Packet length =   41. [Encrypted]
           00 01 02 03  04 05 06 07  08 09 0A 0B  D3 64 6B E0
           36 D1 AD 41  4B 75 F6 AB  1D 5F 6E 09  B8 1A A9 94
           71 45 3F 17  B1 CC 4D 97  0B
=============== Packet Vector #11 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 0000000A
      RA = 40 41 42 43  44 45
Packet length =   32. [Input (12 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 0A  0C 0A 00 20
CBC IV out:A7 6C 6B E2  D5 FB 5F 8D  0C 25 06 07  52 00 5E 5A
CBC[0000]: 18 6F 0F 6B  E2 DC A7 F3  45 7D FF 27  9A 1B A8 3D
CBC[0010]: 4D 62 28 50  7D 61 E0 E4  1F 9F FE 07  53 DB D1 57
Packet length =   42. [MIC appended (10 octets)]
           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
           4D 62 28 50  7D 61 E0 E4  1F 9F
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 0A  0C 8A 00 00
CTR[0001]: 18 CC 6D 52  47 25 C6 B2  80 89 26 CE  44 1D 52 A2
CTR[0002]: 02 F6 F1 F2  A5 F4 7F 99  F8 D6 A1 9F  21 29 DE 19
Packet length =   42. [Encrypted]
           00 01 02 03  04 05 06 07  08 09 0A 0B  14 C1 63 5D
           57 34 D4 A1  94 9C 30 D9  5C 04 48 B9  1E EB EF ED
           E8 96 57 C9  85 B7 41 7B  3E B6
=============== Packet Vector #12 ==================
AES Key:   00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
      TA = 00 01 02 03  04 05      seqNum = 0000000B
      RA = 40 41 42 43  44 45
Packet length =   33. [Input (12 cleartext header octets)]
           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
CBC IV in: 44 45 00 01  02 03 04 05  00 00 00 0B  0C 0A 00 21
CBC IV out:09 3F 87 48  63 03 5C 18  63 4F BE A7  99 C7 AA E0
CBC[0000]: A9 DB B4 1B  9E 18 26 F9  38 0E 02 F0  0C D9 CD 34
CBC[0010]: 16 D5 D4 AD  8C 6E A8 EF  54 39 20 23  0A C2 64 95
CBC[0020]: 15 1D 21 97  FC 66 DA CD  2E 75 2A B3  3E E5 8D 82
Packet length =   43. [MIC appended (10 octets)]
           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 15 1D 21  97 FC 66 DA  CD 2E 75
CTR Start: 44 45 00 01  02 03 04 05  00 00 00 0B  0C 8A 00 00
CTR[0001]: 0B A6 14 2A  D4 EC 93 6D  74 64 CD F9  96 29 0E 2A
CTR[0002]: A3 71 49 C0  6C C3 A9 C6  38 ED E4 AE  9D 57 D1 7A
Packet length =   43. [Encrypted]
           00 01 02 03  04 05 06 07  08 09 0A 0B  07 AB 1A 25
           C4 FD 81 7E  60 71 DB EE  8E 30 14 31  BF 6C 57 DF
           4C D6 B4 E7  AF 11 82 74  50 79 A4
=============== Packet Vector #13 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = B8105D9F
      RA = 17 6C 4D 48  7A 4D
Packet length =   31. [Input (8 cleartext header octets)]
           FC C5 33 32  18 18 84 CA  48 DD 6D 54  90 17 3E 1C
           26 D1 97 D2  56 43 B9 60  28 72 AC A0  12 B0 4C
CBC IV in: 7A 4D 51 98  33 6D 97 4A  B8 10 5D 9F  08 08 00 1F
CBC IV out:E3 78 6F 0D  1D D1 3A 0F  2F 79 C3 8E  C8 3A 8A F7
CBC[0000]: 20 B4 C0 79  98 DC 27 1B  80 D4 6B DC  17 DF 4F 89
```

```
CBC[0010]: DD DB F7 60  3A 42 B3 EF  0F F5 41 B2  B9 BB 7C 82
Packet length =   39. [MIC appended (8 octets)]
           FC C5 33 32  18 18 84 CA  48 DD 6D 54  90 17 3E 1C
           26 D1 97 D2  56 43 B9 60  28 72 AC A0  12 B0 4C DD
           DB F7 60 3A  42 B3 EF
CTR Start: 7A 4D 51 98  33 6D 97 4A  B8 10 5D 9F  08 88 00 00
CTR[0001]: 90 04 73 9B  36 71 F0 45  23 4F FE 20  3E 08 B3 40
CTR[0002]: 54 F6 BF DB  E9 A4 56 D6  8C 01 75 74  0A CE 55 EC
Packet length =   39. [Encrypted]
           FC C5 33 32  18 18 84 CA  D8 D9 1E CF  A6 66 CE 59
           05 9E 69 F2  68 4B 0A 20  7C 84 13 7B  FB 14 1A 0B
           57 F6 15 4E  48 7D BA
=============== Packet Vector #14 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 30FF8C53
      RA = 17 6C 4D 48  7A 4D
Packet length =   32. [Input (8 cleartext header octets)]
           94 85 6A 4C  DE A6 14 ED  72 7F 4D 33  63 FA 64 98
           DE A7 38 52  A7 22 F5 D3  22 54 E7 5D  68 97 68 86
CBC IV in: 7A 4D 51 98  33 6D 97 4A  30 FF 8C 53  08 08 00 20
CBC IV out:01 48 0E 95  BD 38 58 86  D7 5E 5F 05  CC 2F AB 94
CBC[0000]: 68 12 6B D0  A4 8D 06 01  55 D0 39 14  CC 30 B3 06
CBC[0010]: 72 A8 65 DF  35 91 1D 70  4B 74 82 A9  7A A9 B0 3C
Packet length =   40. [MIC appended (8 octets)]
           94 85 6A 4C  DE A6 14 ED  72 7F 4D 33  63 FA 64 98
           DE A7 38 52  A7 22 F5 D3  22 54 E7 5D  68 97 68 86
           72 A8 65 DF  35 91 1D 70
CTR Start: 7A 4D 51 98  33 6D 97 4A  30 FF 8C 53  08 88 00 00
CTR[0001]: 08 88 9D E8  29 3A DB D1  72 F3 C3 2A  5E 18 F2 2E
CTR[0002]: CE 45 A8 D3  82 D4 9C 53  66 B7 8A 43  DD 1A 84 EE
Packet length =   40. [Encrypted]
           94 85 6A 4C  DE A6 14 ED  7A F7 D0 DB  4A C0 BF 49
           AC 54 FB 78  F9 3A 07 FD  EC 11 4F 8E  EA 43 F4 D5
           14 1F EF 9C  E8 8B 99 9E
=============== Packet Vector #15 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = C3EAAF67
      RA = 17 6C 4D 48  7A 4D
Packet length =   33. [Input (8 cleartext header octets)]
           B0 80 4C 3C  EF DC D0 49  52 C8 7E 31  BE CE 12 5F
           CD 6A 44 62  BC 65 40 71  B0 24 97 DA  B8 87 41 E1
           27
CBC IV in: 7A 4D 51 98  33 6D 97 4A  C3 EA AF 67  08 08 00 21
CBC IV out:C5 D0 C1 48  66 EA D0 E9  24 51 14 F8  4C A9 AA 33
CBC[0000]: D1 C8 22 9B  23 1D B4 77  83 7F 01 62  35 8A B1 4B
CBC[0010]: F7 5F 29 F0  93 E1 58 09  CC 22 FB B0  B1 9F 42 72
CBC[0020]: 88 AB BE B2  A8 1E B2 F8  E4 9F 3F 91  CE 97 30 C2
Packet length =   41. [MIC appended (8 octets)]
           B0 80 4C 3C  EF DC D0 49  52 C8 7E 31  BE CE 12 5F
           CD 6A 44 62  BC 65 40 71  B0 24 97 DA  B8 87 41 E1
           27 88 AB BE  B2 A8 1E B2  F8
CTR Start: 7A 4D 51 98  33 6D 97 4A  C3 EA AF 67  08 88 00 00
CTR[0001]: 53 33 53 D6  B5 E3 D6 88  B5 CA D3 28  F6 73 77 5C
CTR[0002]: 12 58 F6 5D  44 5A 1A 25  4F 2A F7 A4  F1 E4 A9 20
CTR[0003]: 89 B2 3E 96  D8 49 EE 8F  AC 17 5E F5  BB 66 2A F8
Packet length =   41. [Encrypted]
           B0 80 4C 3C  EF DC D0 49  01 FB 2D E7  0B 2D C4 D7
           78 A0 97 4A  4A 16 37 2D  A2 7C 61 87  FC DD 5B C4
           68 A2 5C 1A  43 4C B7 92  71
=============== Packet Vector #16 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 2DF690A8
      RA = 17 6C 4D 48  7A 4D
Packet length =   31. [Input (12 cleartext header octets)]
           95 8F 9B 6B  69 0B 86 3A  9E 6D 66 D4  59 D7 E9 D1
           0E E3 A1 2E  9B F6 A6 7A  16 96 D3 1E  DA 5A DB
CBC IV in: 7A 4D 51 98  33 6D 97 4A  2D F6 90 A8  0C 08 00 1F
CBC IV out:F2 14 0D 47  1A BD 45 5D  EF DC 2D 46  1D 92 43 B6
CBC[0000]: EB 2C 23 FD  77 C7 50 9D  86 02 69 C7  78 C8 C4 5E
CBC[0010]: 64 8D A3 4D  25 D0 80 74  D8 6C 8A A6  B2 D9 27 FC
Packet length =   39. [MIC appended (8 octets)]
           95 8F 9B 6B  69 0B 86 3A  9E 6D 66 D4  59 D7 E9 D1
           0E E3 A1 2E  9B F6 A6 7A  16 96 D3 1E  DA 5A DB 64
           8D A3 4D 25  D0 80 74
CTR Start: 7A 4D 51 98  33 6D 97 4A  2D F6 90 A8  0C 88 00 00
```

```
CTR[0001]: 61 D3 AD 80  41 6B B5 D5  34 CD 51 BF  02 51 ED 0C
CTR[0002]: 7D D8 25 A9  D2 C6 13 5A  88 52 9B 4C  E0 BF A9 D9
Packet length =   39. [Encrypted]
           95 8F 9B 6B  69 0B 86 3A  9E 6D 66 D4  38 04 44 51
           4F 88 14 FB  AF 3B F7 C5  14 C7 3E 12  A7 82 FE CD
           5F 65 5E 7F  58 D2 EF
=============== Packet Vector #17 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 686613C7
      RA = 17 6C 4D 48  7A 4D
Packet length =   32. [Input (12 cleartext header octets)]
           EE 35 B1 CE  8B 99 34 8E  84 4A 1E 5D  E0 32 AD 31
           9A AB 07 57  5F 60 69 B8  B0 F4 42 A6  4C D6 A3 76
CBC IV in: 7A 4D 51 98  33 6D 97 4A  68 66 13 C7  0C 08 00 20
CBC IV out:C5 5A F3 F6  8F C6 C5 2A  C5 F0 33 9B  F3 68 BB 3A
CBC[0000]: D8 F8 B9 A4  38 B5 65 FF  BA 44 81 16  6A 52 24 2C
CBC[0010]: BA 9A A6 B9  41 9F E2 94  7B 1C CE C3  3A BD 10 47
Packet length =   40. [MIC appended (8 octets)]
           EE 35 B1 CE  8B 99 34 8E  84 4A 1E 5D  E0 32 AD 31
           9A AB 07 57  5F 60 69 B8  B0 F4 42 A6  4C D6 A3 76
           BA 9A A6 B9  41 9F E2 94
CTR Start: 7A 4D 51 98  33 6D 97 4A  68 66 13 C7  0C 88 00 00
CTR[0001]: 2B FF 08 02  D6 A5 11 4A  D8 91 43 8F  63 6D DF 4A
CTR[0002]: D2 FF F9 79  A8 D5 65 5A  28 94 26 EB  9B FA 30 D4
Packet length =   40. [Encrypted]
           EE 35 B1 CE  8B 99 34 8E  84 4A 1E 5D  CB CD A5 33
           4C 0E 16 1D  87 F1 2A 37  D3 99 9D EC  9E 29 5A 0F
           12 4F C3 E3  69 0B C4 7F
=============== Packet Vector #18 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 75912FEC
      RA = 17 6C 4D 48  7A 4D
Packet length =   33. [Input (12 cleartext header octets)]
           1B E5 C8 F8  D7 4C D9 CA  F5 21 35 E5  5C 0D 04 58
           93 7C 1A 98  BA F5 13 8C  BB F8 E5 6D  A9 6A 4D 94
           C6
CBC IV in: 7A 4D 51 98  33 6D 97 4A  75 91 2F EC  0C 08 00 21
CBC IV out:C9 87 FD 36  BA 3B 59 9D  CE AD 4A 43  F7 A1 E6 30
CBC[0000]: CC 64 13 9D  5C 1E 6D 4D  70 80 FA EA  20 3C 34 A4
CBC[0010]: C1 76 E1 5A  7C FF 41 03  F8 0D 41 10  0A E8 40 BA
CBC[0020]: 9F 05 72 AF  53 BF 6F 32  EB CD 9E BF  2C 83 D8 EA
Packet length =   41. [MIC appended (8 octets)]
           1B E5 C8 F8  D7 4C D9 CA  F5 21 35 E5  5C 0D 04 58
           93 7C 1A 98  BA F5 13 8C  BB F8 E5 6D  A9 6A 4D 94
           C6 9F 05 72  AF 53 BF 6F  32
CTR Start: 7A 4D 51 98  33 6D 97 4A  75 91 2F EC  0C 88 00 00
CTR[0001]: EE 54 AF 28  33 25 28 42  19 E1 E9 DA  3A 2A 45 51
CTR[0002]: 9B F1 DA 7C  38 A2 87 0A  6D 47 C0 FF  98 61 45 0F
Packet length =   41. [Encrypted]
           1B E5 C8 F8  D7 4C D9 CA  F5 21 35 E5  B2 59 AB 70
           A0 59 32 DA  A3 14 FA 56  81 D2 A0 3C  32 9B 97 E8
           FE 3D 82 78  C2 14 7F 90  AA
=============== Packet Vector #19 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 111900AF
      RA = 17 6C 4D 48  7A 4D
Packet length =   31. [Input (8 cleartext header octets)]
           F8 E4 4B 74  59 2C 12 2F  58 79 07 B9  C4 78 F0 24
           D7 50 6C B4  23 66 4C 58  14 C1 D8 79  8D CD 41
CBC IV in: 7A 4D 51 98  33 6D 97 4A  11 19 00 AF  08 0A 00 1F
CBC IV out:11 2E 18 D6  8E 74 16 92  32 F1 39 CA  7B 5B B7 1E
CBC[0000]: 30 20 56 E4  5C D9 AC CE  A8 E0 1D 3E  F9 09 4C 70
CBC[0010]: 34 84 69 E5  9F 41 FC A4  90 2C 5B 09  11 2A 88 FF
Packet length =   41. [MIC appended (10 octets)]
           F8 E4 4B 74  59 2C 12 2F  58 79 07 B9  C4 78 F0 24
           D7 50 6C B4  23 66 4C 58  14 C1 D8 79  8D CD 41 34
           84 69 E5 9F  41 FC A4 90  2C
CTR Start: 7A 4D 51 98  33 6D 97 4A  11 19 00 AF  08 8A 00 00
CTR[0001]: 74 0E AE DB  4E E9 45 01  0B 17 EF 39  2D 38 3F A6
CTR[0002]: E1 4F 3A F4  2A 28 44 4F  53 B7 AA 22  05 0C 44 AF
CTR[0003]: 0C EA BF 2D  2F CF 21 72  E2 09 5B 53  F8 11 D2 F5
Packet length =   41. [Encrypted]
           F8 E4 4B 74  59 2C 12 2F  2C 77 A9 62  8A 91 B5 25
           DC 47 83 8D  0E 5E 73 FE  F5 8E E2 8D  A7 E5 05 7B
           D7 DE 4F BD  44 F0 E0 3F  20
```

```
=============== Packet Vector #20 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 75FBD7E1
      RA = 17 6C 4D 48  7A 4D
Packet length =   32. [Input (8 cleartext header octets)]
           23 E9 14 CE  1A 8D 18 5D  6A 81 8E 4B  97 62 D4 BD
           25 61 16 0A  29 13 E7 BA  1F 91 AD 0D  51 C3 7B D6
CBC IV in: 7A 4D 51 98  33 6D 97 4A  75 FB D7 E1  08 0A 00 20
CBC IV out:B0 72 3B BB  26 10 03 D7  E6 75 10 95  78 1E E0 AC
CBC[0000]: C3 E9 50 46  29 E9 48 B3  1B 9B 00 E2  40 2C 2D 2F
CBC[0010]: DE FB 46 CA  8A 93 70 C6  0C E1 E6 B0  15 9C 75 AE
Packet length =   42. [MIC appended (10 octets)]
           23 E9 14 CE  1A 8D 18 5D  6A 81 8E 4B  97 62 D4 BD
           25 61 16 0A  29 13 E7 BA  1F 91 AD 0D  51 C3 7B D6
           DE FB 46 CA  8A 93 70 C6  0C E1
CTR Start: 7A 4D 51 98  33 6D 97 4A  75 FB D7 E1  08 8A 00 00
CTR[0001]: 14 DC D9 BF  A9 76 73 2E  81 4B DC 9C  41 A4 6E 54
CTR[0002]: 3A 40 A2 60  EB EE 06 51  9A 57 D9 D5  DB DC 5E E9
CTR[0003]: 25 23 20 B7  61 34 34 BD  31 2F E2 2A  3B A8 94 2E
Packet length =   42. [Encrypted]
           23 E9 14 CE  1A 8D 18 5D  7E 5D 57 F4  3E 14 A7 93
           A4 2A CA 96  68 B7 89 EE  25 D1 0F 6D  BA 2D 7D 87
           44 AC 9F 1F  51 4F 2E 2F  29 C2
=============== Packet Vector #21 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 305A175C
      RA = 17 6C 4D 48  7A 4D
Packet length =   33. [Input (8 cleartext header octets)]
           A4 EB 63 B1  F6 A8 07 CC  30 A8 64 AB  2F 60 7C 2D
           44 11 B6 AB  01 62 3D 1E  8B E4 D0 B0  8C 72 08 D5
           A6
CBC IV in: 7A 4D 51 98  33 6D 97 4A  30 5A 17 5C  08 0A 00 21
CBC IV out:37 DE E3 9C  4B EF 1A 85  DB E9 7C C9  58 4F 8B 23
CBC[0000]: AD 28 08 B7  B1 B4 86 49  E2 64 89 42  9E E2 CA CC
CBC[0010]: 29 66 42 1F  AA 60 A5 AF  83 E2 EC 56  37 8A 6D 27
CBC[0020]: 72 8F 12 61  10 0F 0D B8  8F 72 72 1D  04 DC 8E 2E
Packet length =   43. [MIC appended (10 octets)]
           A4 EB 63 B1  F6 A8 07 CC  30 A8 64 AB  2F 60 7C 2D
           44 11 B6 AB  01 62 3D 1E  8B E4 D0 B0  8C 72 08 D5
           A6 72 8F 12  61 10 0F 0D  B8 8F 72
CTR Start: 7A 4D 51 98  33 6D 97 4A  30 5A 17 5C  08 8A 00 00
CTR[0001]: 0E F6 9B 57  E1 72 41 88  BF E3 7B 36  54 2B 0E C1
CTR[0002]: 84 0B E6 04  99 09 57 0A  E1 62 DF 59  90 F6 50 35
CTR[0003]: 98 58 77 C2  1C 72 AA 28  E8 A6 F1 AC  8D E1 E5 01
Packet length =   43. [Encrypted]
           A4 EB 63 B1  F6 A8 07 CC  3E 5E FF FC  CE 12 3D A5
           FB F2 CD 9D  55 49 33 DF  0F EF 36 B4  15 7B 5F DF
           47 10 50 4B  F1 E6 5F 38  20 D7 05
=============== Packet Vector #22 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 673AF43C
      RA = 17 6C 4D 48  7A 4D
Packet length =   31. [Input (12 cleartext header octets)]
           6E 45 1C EB  FF FF D7 6E  1E FF A8 E6  6F AA 23 3A
           08 2D 7B 33  8E 7B 3D 85  3D 40 D3 E3  90 30 38
CBC IV in: 7A 4D 51 98  33 6D 97 4A  67 3A F4 3C  0C 0A 00 1F
CBC IV out:94 30 BB D2  5E D7 F3 8C  B1 C1 2A 10  D6 64 F1 B9
CBC[0000]: DF 29 8D D5  EF E4 43 E7  6A 1C 25 7F  90 D6 49 3B
CBC[0010]: D9 59 B6 3F  C7 BF 84 D4  53 FC 41 1C  37 70 BE 16
Packet length =   41. [MIC appended (10 octets)]
           6E 45 1C EB  FF FF D7 6E  1E FF A8 E6  6F AA 23 3A
           08 2D 7B 33  8E 7B 3D 85  3D 40 D3 E3  90 30 38 D9
           59 B6 3F C7  BF 84 D4 53  FC
CTR Start: 7A 4D 51 98  33 6D 97 4A  67 3A F4 3C  0C 8A 00 00
CTR[0001]: 95 C9 E1 02  1B 72 EB 6F  0A 71 02 2E  04 A0 01 62
CTR[0002]: 7C 99 24 51  08 2A 12 EB  1F F2 95 B5  87 FB C1 F8
Packet length =   41. [Encrypted]
           6E 45 1C EB  FF FF D7 6E  1E FF A8 E6  FA 63 C2 38
           13 5F 90 5C  84 0A 3F AB  39 E0 D2 81  EC A9 1C 88
           51 9C 2D 2C  A0 76 41 E6  7B
=============== Packet Vector #23 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = D929DDA2
      RA = 17 6C 4D 48  7A 4D
Packet length =   32. [Input (12 cleartext header octets)]
```

```
           17 99 E4 7D  8B 8A A0 07  E4 17 23 C4  F4 8C 5E DC
           C6 8D B9 F2  F0 7A 01 6A  49 5E 25 66  40 31 80 FD
CBC IV in: 7A 4D 51 98  33 6D 97 4A  D9 29 DD A2  0C 0A 00 20
CBC IV out:88 E5 C8 A8  9E 66 A7 0F  39 99 58 E2  3B 48 DF 46
CBC[0000]: 27 9D AC B7  56 F9 CE 40  18 A0 66 4A  50 65 E1 0E
CBC[0010]: 65 5B 1B 0F  EB 70 CD 97  02 1F 10 15  2C 77 47 8C
Packet length =   42. [MIC appended (10 octets)]
           17 99 E4 7D  8B 8A A0 07  E4 17 23 C4  F4 8C 5E DC
           C6 8D B9 F2  F0 7A 01 6A  49 5E 25 66  40 31 80 FD
           65 5B 1B 0F  EB 70 CD 97  02 1F
CTR Start: 7A 4D 51 98  33 6D 97 4A  D9 29 DD A2  0C 8A 00 00
CTR[0001]: 48 B5 87 39  E7 BE CD A0  FC 7A FF EA  D3 3C 4C 16
CTR[0002]: DC BB 4C E3  11 6A DE 9A  70 36 11 91  39 F7 27 1F
Packet length =   42. [Encrypted]
           17 99 E4 7D  8B 8A A0 07  E4 17 23 C4  BC 39 D9 E5
           21 33 74 52  0C 00 FE 80  9A 62 69 70  9C 8A CC 1E
           74 31 C5 95  9B 46 DC 06  3B E8
=============== Packet Vector #24 ==================
AES Key:   98 07 44 7D  42 1F BD B0  97 FE 6C E2  87 DE 1E 7C
      TA = 51 98 33 6D  97 4A      seqNum = 52E1B7EB
      RA = 17 6C 4D 48  7A 4D
Packet length =   33. [Input (12 cleartext header octets)]
           FB B8 3D B7  DC E1 0B 13  D8 88 1B 60  60 04 6F 4A
           9D E9 4C AB  B5 82 87 08  34 85 F6 F8  3D 0F 16 23
           E6
CBC IV in: 7A 4D 51 98  33 6D 97 4A  52 E1 B7 EB  0C 0A 00 21
CBC IV out:9A 95 CF 11  2C 33 7B BB  1E 36 C0 2A  44 B4 78 88
CBC[0000]: D7 35 4B 0E  5E C8 7E BB  71 0A 7F 3B  AB AD 9F 43
CBC[0010]: B9 A4 FA CD  C0 50 D0 2A  65 8B BA 1B  32 4A 21 EC
CBC[0020]: F1 AA 34 E8  57 17 7A B2  F0 E4 E0 E9  FF B5 DF F7
Packet length =   43. [MIC appended (10 octets)]
           FB B8 3D B7  DC E1 0B 13  D8 88 1B 60  60 04 6F 4A
           9D E9 4C AB  B5 82 87 08  34 85 F6 F8  3D 0F 16 23
           E6 F1 AA 34  E8 57 17 7A  B2 F0 E4
CTR Start: 7A 4D 51 98  33 6D 97 4A  52 E1 B7 EB  0C 8A 00 00
CTR[0001]: A8 33 A2 04  8F 3C F9 D8  35 B6 46 55  4F B3 DE 21
CTR[0002]: E2 B5 15 15  BB 88 07 5E  48 21 B8 E7  73 E5 DC B4
Packet length =   43. [Encrypted]
           FB B8 3D B7  DC E1 0B 13  D8 88 1B 60  C8 37 CD 4E
           12 D5 B5 73  80 34 C1 5D  7B 36 28 D9  DF BA 03 36
           5D 79 AD 6A  A0 76 AF 9D  C1 15 38
```

Appendix B. Test Vector Generation Source Code

```
//
//=================================================================
// Proposed AES CTR/CBC-MAC mode test vector generation
//
// Author:  Doug Whiting, Hifn  (dwhiting@hifn.com)
//
// This code is released to the public domain, on an as-is basis.
//=================================================================
//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <assert.h>
#include "aes_defs.h"       // AES calling interface
#include "aes_vect.h"       // NIST AES test vectors

typedef int BOOL;           // boolean

enum
    {
    BLK_SIZE    =   16,     // # octets in an AES block
    MAX_PACKET  =   3*512   // largest packet size
    };

union block                 // AES cipher block
    {
    u32b  x[BLK_SIZE/4];    // access as 8-bit octets or 32-bit words
    u08b  b[BLK_SIZE];
    };

struct packet
    {
    BOOL    encrypted;      // TRUE if encrypted
    u08b    TA[6],RA[6];    // xmit/rcv address
    int     micLength;      // # octets of MIC appended to plaintext (M)
    int     clrCount;       // # cleartext octets covered by MIC (H)
    block   key;            // the encryption key (K)
    u32b    seqNum;         // unique packet sequence number (e.g., WEP IV)
    int     length;         // # octets in data[]
    u08b    data[MAX_PACKET+2*BLK_SIZE];    // packet contents
    };

struct
    {
    int     cnt;            // how many words left in ct
    block   ptCntr;         // the counter input
    block   ct;             // the ciphertext (prng output)
    } prng;

// return the 32-bit value read to be stored as a big-endian word
u32b   BigEndian(u32b   x)
    {
    static block b={0,0,0,0};

    if (b.x[0]==0)          // first time, figure out endianness
        b.x[0]=0xFF000001;

    if (b.b[0]==0xFF)
        return x;
    if (b.b[0]==1)
        return (x >> 24) + (x << 24) +
              ((x >>  8) & 0x00FF00) + ((x <<  8) & 0xFF0000);
    assert(0);              // should never happen
    return 0;
    }

void InitRand(u32b   seed)
    {
    memset(prng.ptCntr.b,0,BLK_SIZE);
    prng.ptCntr.x[(BLK_SIZE/4)-1]=seed*17;
    prng.cnt=0;             // the pump is dry
    }

// prng: does not use C rand(), so this should be repeatable across platforms
u32b   Random32(void)
    {
    if (prng.cnt == 0)
        {   // use whatever key is currently defined
        prng.cnt=BLK_SIZE/4;
        prng.ptCntr.x[0]++;
        if (prng.ptCntr.x[0]==0)    // ripple carry?
            prng.ptCntr.x[1]++;     // (stop at 64 bits)
        AES_Encrypt(prng.ptCntr.x,prng.ct.x);
        }
    --prng.cnt;
    return BigEndian(prng.ct.x[prng.cnt]);
    }

// display a block
void ShowBlock(const block *blk,const char *prefix,const char *suffix,int a)
    {
    int i;
    printf(prefix,a);
    for (i=0;i<BLK_SIZE;i++)
        printf("%02X%s",blk->b[i],((i&3)==3)?"  ":" ");
    printf(suffix);
    }

void ShowAddr(const packet *p)
    {
    int i;

    printf("      TA = ");
    for (i=0;i<6;i++) printf("%02X%s",p->TA[i],(i==3)?"  ":" ");
    printf("     seqNum = %08X\n",p->seqNum);
    printf("      RA = ");
    for (i=0;i<6;i++) printf("%02X%s",p->RA[i],(i==3)?"  ":" ");
    printf("\n");
    }

// display a packet
void ShowPacket(const packet *p,const char *pComment,int a)
    {
    int i;

    printf("Packet length = %4d. ",p->length);
    printf(pComment,a);
    if (p->encrypted) printf("[Encrypted]");
    for (i=0;i<p->length;i++)
        {
        if ((i & 15) == 0) printf("\n%11s","");
        printf("%02X%s",p->data[i],((i&3)==3)?"  ":" ");
        }
    printf("\n");
    }

// make sure that encrypt/decrypt work according to NIST vectors
void Validate_NIST_AES_Vectors(int verbose)
    {
    int     i;
    block   key,pt,ct,rt;

    printf("AES KAT Vectors:\n");   // known-answer tests
    // variable text (fixed-key) tests
    memcpy(key.b,VT_key,BLK_SIZE);
    AES_SetKey(key.x,BLK_SIZE*8);
    for (i=0;i<sizeof(VT_pt_ct_pairs);i+=2*BLK_SIZE)
         {
         memcpy(pt.b,VT_pt_ct_pairs+i,BLK_SIZE);
         AES_Encrypt(pt.x,ct.x);
         if (memcmp(ct.x,VT_pt_ct_pairs+i+BLK_SIZE,BLK_SIZE))
             {
             printf("Vector miscompare at VT test #%d",i);
             exit(1);
             }
         AES_Decrypt(ct.x,rt.x);    // sanity check on decrypt
         if (memcmp(pt.b,rt.b,BLK_SIZE))
             {
             printf("Decrypt miscompare at VT test #%d",i);
             exit(1);
             }
         if (verbose)   // only do a little if we're "debugging"
             {  printf("\n");    break; }
         else if (i==0)
             {          //  display the first vector
             ShowBlock(&key,"Key:       ","\n",0);
             ShowBlock(&pt ,"PT:        ","\n",0);
             ShowBlock(&ct ,"CT:        ","\n\n",0);
             }
         }

    // variable key (fixed-text) tests
    memcpy(pt.b,VK_pt,BLK_SIZE);
    for (i=0;i<sizeof(VK_key_ct_pairs);i+=2*BLK_SIZE)
         {
         memcpy(key.b,VK_key_ct_pairs+i,BLK_SIZE);
         AES_SetKey(key.x,BLK_SIZE*8);
         AES_Encrypt(pt.x,ct.x);
         if (memcmp(ct.x,VK_key_ct_pairs+i+BLK_SIZE,BLK_SIZE))
             {
             printf("Vector miscompare at VK test #%d",i);
             exit(1);
             }
         AES_Decrypt(ct.x,rt.x);    // sanity check on decrypt
         if (memcmp(pt.b,rt.b,BLK_SIZE))
             {
             printf("Decrypt miscompare at VK test #%d",i);
             exit(1);
             }
         if (verbose)   // only do a little if we're "debugging"
             {  printf("\n");    break; }
         else if (i==0)
             {          //  display the first vector
             ShowBlock(&key,"Key:       ","\n",0);
             ShowBlock(&pt ,"PT:        ","\n",0);
             ShowBlock(&ct ,"CT:        ","\n\n",0);
             }
         }
    printf("NIST AES Vectors: OK\n");   // if we got here, it's all cool
    }

// assumes AES_SetKey is called elsewhere
void Generate_CTR_CBC_Vector(packet *p,int verbose)
    {
    int     i,j,C;
    block   m,x;
    assert(p->length    >= p->clrCount && p->length    <= MAX_PACKET);
    assert(p->micLength >  0           && p->micLength <= BLK_SIZE);

    ShowPacket(p,"[Input (%d cleartext header octets)]",p->clrCount);

    // compute & append the CBC-MAC
    m.b[0]  = p->RA[4];
    m.b[1]  = p->RA[5];
    m.b[2]  = p->TA[0];
    m.b[3]  = p->TA[1];
    m.b[4]  = p->TA[2];
    m.b[5]  = p->TA[3];
    m.b[6]  = p->TA[4];
    m.b[7]  = p->TA[5];
    m.x[2]  = BigEndian(p->seqNum);
    m.x[3]  = BigEndian(p->length + (p->micLength << 16) + (p->clrCount<<24));

    AES_Encrypt(m.x,x.x);               // produce the CBC IV
    ShowBlock(&m,"CBC IV in: ","\n",0);
    if (verbose) ShowBlock(&x,"CBC IV out:","\n",0);
    memset(p->data+p->length,0,BLK_SIZE); // zero pad for purposes of CBC-MAC
    for (i=0;i<p->length;i+=BLK_SIZE)
        {
        for (j=0;j<BLK_SIZE;j++)        // do the CBC xor
```

```
            x.b[j] = x.b[j] ^ p->data[i+j];
        AES_Encrypt(x.x,x.x);           // and encrypt the block in place
        if (verbose) ShowBlock(&x,"CBC[%04X]: ","\n",i);
        }
    for (i=0;i<p->micLength;i++)        // truncate & append MAC to packet
        p->data[p->length+i]=x.b[i];
    p->length+=p->micLength;
    if (verbose) ShowPacket(p,"[MIC appended (%d octets)]",p->micLength);

    // now encrypt the packet+CBC-MAC, using CTR mode
    m.b[13] |= 0x80;                   // set the encrypt bit
    for (i=0;i+p->clrCount < p->length;i++)
        {
        if ((i % BLK_SIZE) == 0)
            {                           // generate new keystream block
            C = i/16;
            m.b[14] = C/256;
            m.b[15] = C%256;
            AES_Encrypt(m.x,x.x);       // then encrypt the counter
            if (verbose && i==0) ShowBlock(&m,"CTR Start: ","\n",0);
            if (verbose) ShowBlock(&x,"CTR[%04X]: " ,"\n",1+(i/BLK_SIZE));
            }
        p->data[i+p->clrCount] ^= x.b[i % BLK_SIZE];    // merge in the keystream
        }
    p->encrypted = 1;
    ShowPacket(p,"",0);                 // show the final encrypted packet
    }

int main(int argc,char *argv[])
    {
    int     i,j,k,len,pktNum,seed;
    packet  p;

    seed = (argc > 1) ? atoi(argv[1]) : (int) time(NULL);
    InitRand(seed);
    printf("%s C compiler [%s %s].\nRandom seed = %d\n",
           COMPILER_ID,__DATE__,__TIME__,seed);

    // first, make sure that our AES code matches NIST KAT vectors
    Validate_NIST_AES_Vectors(_VERBOSE_);

    // generate CTR-CBC vectors for various parameter settings
    for (k=pktNum=0;k<2;k++)
        {   // k==1 --> random vectors. k==0 --> "visually simple" vectors
        for (i=0;i<BLK_SIZE  ;i++)
            p.key.b [i]=(k) ? (u08b) Random32() & 0xFF : i;
        for (i=0;i<6;i++)
            {
            p.TA[i]=(k) ? (u08b) Random32() & 0xFF : i;
            p.RA[i]=(k) ? (u08b) Random32() & 0xFF : i + 0x40;
            }
        AES_SetKey(p.key.x,BLK_SIZE*8);     // run the key schedule

        // now generate the vectors
        for (p.micLength  = 8;p.micLength  <12;p.micLength+=2)
        for (p.clrCount   = 8;p.clrCount   <16;p.clrCount+=4)
        for (len          =32;len          <64;len*=2)
        for (i            =-1;i            < 2;i++)
            {
            p.seqNum = (k) ? Random32() : pktNum;
            p.length = len+i;               // len+i is packet length
            p.encrypted = 0;
            assert(p.length <= MAX_PACKET);
            for (j=0;j<p.length;j++)        // generate random packet contents
                p.data[j]=(k) ? (u08b  ) Random32() & 0xFF : j;
            pktNum++;
            printf("=============== Packet Vector #%d ==================\n",pktNum);
            ShowBlock(&p.key ,"AES Key:   ","\n",0);
            ShowAddr (&p);
            Generate_CTR_CBC_Vector(&p,1);
            }
        }
    return 0;
    }
```