                    Counter with CBC-MAC (CCM)

Status of this Memo

   This memo provides information for the Internet community.  It does
   not specify an Internet standard of any kind.  Distribution of this
   memo is unlimited.

Abstract

   Counter with CBC-MAC (CCM) is a generic authenticated encryption
   block cipher mode.  CCM is defined for use with 128-bit block
   ciphers, such as the Advanced Encryption Standard (AES).

1.  Introduction

   Counter with CBC-MAC (CCM) is a generic authenticated encryption
   block cipher mode.  CCM is only defined for use with 128-bit block
   ciphers, such as AES [AES].  The CCM design principles can easily be
   applied to other block sizes, but these modes will require their own
   specifications.

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.  CCM Mode Specification

   For the generic CCM mode there are two parameter choices.  The first
   choice is M, the size of the authentication field.  The choice of the
   value for M involves a trade-off between message expansion and the
   probability that an attacker can undetectably modify a message.
   Valid values are 4, 6, 8, 10, 12, 14, and 16 octets.  The second

choice is L, the size of the length field.  This value requires a
trade-off between the maximum message size and the size of the Nonce.
Different applications require different trade-offs, so L is a
parameter.  Valid values of L range between 2 octets and 8 octets
(the value L=1 is reserved).

```
Name  Description                             Size    Encoding
----  --------------------------------------  ------  --------
M     Number of octets in authentication field 3 bits (M-2)/2
L     Number of octets in length field         3 bits L-1
```

## 2.1.  Inputs

To authenticate and encrypt a message the following information is
required:

1.  An encryption key K suitable for the block cipher.

2.  A nonce N of 15-L octets.  Within the scope of any encryption key
    K, the nonce value MUST be unique.  That is, the set of nonce
    values used with any given key MUST NOT contain any duplicate
    values.  Using the same nonce for two different messages
    encrypted with the same key destroys the security properties of
    this mode.

3.  The message m, consisting of a string of $l(m)$ octets where $0 <=
    l(m) < 2^{(8L)}$.  The length restriction ensures that $l(m)$ can be
    encoded in a field of L octets.

4.  Additional authenticated data a, consisting of a string of $l(a)$
    octets where $0 <= l(a) < 2^{64}$.  This additional data is
    authenticated but not encrypted, and is not included in the
    output of this mode.  It can be used to authenticate plaintext
    packet headers, or contextual information that affects the
    interpretation of the message.  Users who do not wish to
    authenticate additional data can provide a string of length zero.

The inputs are summarized as:

```
Name  Description                        Size
----  ---------------------------------  -----------------------
K     Block cipher key                   Depends on block cipher
N     Nonce                              15-L octets
m     Message to authenticate and encrypt l(m) octets
a     Additional authenticated data      l(a) octets
```

2.2.  Authentication

   The first step is to compute the authentication field T.  This is
   done using CBC-MAC [MAC].  We first define a sequence of blocks B_0,
   B_1, ..., B_n and then apply CBC-MAC to these blocks.

   The first block B_0 is formatted as follows, where l(m) is encoded in
   most-significant-byte first order:

      Octet Number    Contents
      ------------    ---------
      0               Flags
      1 ... 15-L      Nonce N
      16-L ... 15     l(m)

   Within the first block B_0, the Flags field is formatted as follows:

      Bit Number    Contents
      ----------    ----------------------
      7             Reserved (always zero)
      6             Adata
      5 ... 3       M'
      2 ... 0       L'

   Another way say the same thing is:  Flags = 64*Adata + 8*M' + L'.

   The Reserved bit is reserved for future expansions and should always
   be set to zero.  The Adata bit is set to zero if l(a)=0, and set to
   one if l(a)>0.  The M' field is set to (M-2)/2.  As M can take on the
   even values from 4 to 16, the 3-bit M' field can take on the values
   from one to seven.  The 3-bit field MUST NOT have a value of zero,
   which would correspond to a 16-bit integrity check value.  The L'
   field encodes the size of the length field used to store l(m).  The
   parameter L can take on the values from 2 to 8 (recall, the value L=1
   is reserved).  This value is encoded in the 3-bit L' field using the
   values from one to seven by choosing L' = L-1 (the zero value is
   reserved).

   If l(a)>0 (as indicated by the Adata field), then one or more blocks
   of authentication data are added.  These blocks contain l(a) and a
   encoded in a reversible manner.  We first construct a string that
   encodes l(a).

   If $0 < l(a) < (2^{16} - 2^8)$, then the length field is encoded as two
   octets which contain the value l(a) in most-significant-byte first
   order.

If $(2^{16} - 2^8) <= l(a) < 2^{32}$, then the length field is encoded as
six octets consisting of the octets 0xff, 0xfe, and four octets
encoding l(a) in most-significant-byte-first order.

If $2^{32} <= l(a) < 2^{64}$, then the length field is encoded as ten
octets consisting of the octets 0xff, 0xff, and eight octets encoding
l(a) in most-significant-byte-first order.

The length encoding conventions are summarized in the following
table.  Note that all fields are interpreted in most-significant-byte
first order.

```
   First two octets    Followed by        Comment
   -----------------   ---------------    ------------------------------
   0x0000              Nothing            Reserved
   0x0001 ... 0xFEFF   Nothing            For 0 < l(a) < (2^16 - 2^8)
   0xFF00 ... 0xFFFD   Nothing            Reserved
   0xFFFE              4 octets of l(a)   For (2^16 - 2^8) <= l(a) < 2^32
   0xFFFF              8 octets of l(a)   For 2^32 <= l(a) < 2^64
```

The blocks encoding a are formed by concatenating this string that
encodes l(a) with a itself, and splitting the result into 16-octet
blocks, and then padding the last block with zeroes if necessary.
These blocks are appended to the first block B0.

After the (optional) additional authentication blocks have been
added, we add the message blocks.  The message blocks are formed by
splitting the message m into 16-octet blocks, and then padding the
last block with zeroes if necessary.  If the message m consists of
the empty string, then no blocks are added in this step.

The result is a sequence of blocks B0, B1, ..., Bn.  The CBC-MAC is
computed by:

```
   X_1 := E( K, B_0 )
   X_i+1 := E( K, X_i XOR B_i )  for i=1, ..., n
   T := first-M-bytes( X_n+1 )
```

where E() is the block cipher encryption function, and T is the MAC
value.  CCM was designed with AES in mind for the E() function, but
any 128-bit block cipher can be used.  Note that the last block B_n
is XORed with X_n, and the result is encrypted with the block cipher.
If needed, the ciphertext is truncated to give T.

2.3.  Encryption

   To encrypt the message data we use Counter (CTR) mode.  We first
   define the key stream blocks by:

      $S_i := E( K, A_i )$   for i=0, 1, 2, ...

   The values $A_i$ are formatted as follows, where the Counter field i is
   encoded in most-significant-byte first order:

      Octet Number    Contents
      ------------    ---------
      0               Flags
      1 ... 15-L      Nonce N
      16-L ... 15     Counter i

   The Flags field is formatted as follows:

      Bit Number    Contents
      ----------    ----------------------
      7             Reserved (always zero)
      6             Reserved (always zero)
      5 ... 3       Zero
      2 ... 0       L'

   Another way say the same thing is:  Flags = L'.

   The Reserved bits are reserved for future expansions and MUST be set
   to zero.  Bit 6 corresponds to the Adata bit in the B_0 block, but as
   this bit is not used here, it is reserved and MUST be set to zero.
   Bits 3, 4, and 5 are also set to zero, ensuring that all the A blocks
   are distinct from B_0, which has the non-zero encoding of M in this
   position.  Bits 0, 1, and 2 contain L', using the same encoding as in
   B_0.

   The message is encrypted by XORing the octets of message m with the
   first l(m) octets of the concatenation of S_1, S_2, S_3, ... .  Note
   that S_0 is not used to encrypt the message.

   The authentication value U is computed by encrypting T with the key
   stream block S_0 and truncating it to the desired length.

      U := T XOR first-M-bytes( S_0 )

2.4.  Output

   The final result c consists of the encrypted message followed by the
   encrypted authentication value U.

2.5.  Decryption and Authentication Checking

   To decrypt a message the following information is required:

      1.  The encryption key K.

      2.  The nonce N.

      3.  The additional authenticated data a.

      4.  The encrypted and authenticated message c.

   Decryption starts by recomputing the key stream to recover the
   message m and the MAC value T.  The message and additional
   authentication data is then used to recompute the CBC-MAC value and
   check T.

   If the T value is not correct, the receiver MUST NOT reveal any
   information except for the fact that T is incorrect.  The receiver
   MUST NOT reveal the decrypted message, the value T, or any other
   information.

2.6.  Restrictions

   To preserve security, implementations need to limit the total amount
   of data that is encrypted with a single key; the total number of
   block cipher encryption operations in the CBC-MAC and encryption
   together cannot exceed $2^{61}$.  (This allows nearly $2^{64}$ octets to be
   encrypted and authenticated using CCM.  This is roughly 16 million
   terabytes, which should be more than enough for most applications.)
   In an environment where this limit might be reached, the sender MUST
   ensure that the total number of block cipher encryption operations in
   the CBC-MAC and encryption together does not exceed $2^{61}$.  Receivers
   that do not expect to decrypt the same message twice MAY also check
   this limit.

   The recipient MUST verify the CBC-MAC before releasing any
   information such as the plaintext.  If the CBC-MAC verification
   fails, the receiver MUST destroy all information, except for the fact
   that the CBC-MAC verification failed.

3.  Security Proof

   Jakob Jonsson has developed a security proof of CCM [PROOF].  The
   resulting paper was presented at the SAC 2002 conference.  The proof
   shows that CCM provides a level of confidentiality and authenticity
   that is in line with other proposed authenticated encryption modes,
   such as OCB mode [OCB].

4.  Rationale

   The main difficulty in specifying this mode is the trade-off between
   nonce size and counter size.  For a general mode we want to support
   large messages.  Some applications use only small messages, but would
   rather have a larger nonce.  Introducing the L parameter solves this
   issue.  The parameter M gives the traditional trade-off between
   message expansion and probability of forgery.  For most applications,
   we recommend choosing M at least 8.

   The CBC-MAC is computed over a sequence of blocks that encode the
   relevant data in a unique way.  Given the block sequence it is easy
   to recover N, M, L, m, and a.  The length encoding of a was chosen to
   be simple and efficient when a is empty and when a is small.  We
   expect that many implementations will limit the maximum size of a.

   CCM encryption is a straightforward application of CTR mode [MODES].
   As some implementations will support a variable length counter field,
   we have ensured that the least significant octet of the counter is at
   one end of the field.  This also ensures that the counter is aligned
   on the block boundary.

   By encrypting T we avoid CBC-MAC collision attacks.  If the block
   cipher behaves as a pseudo-random permutation, then the key stream is
   indistinguishable from a random string.  Thus, the attacker gets no
   information about the CBC-MAC results.  The only avenue of attack
   that is left is a differential-style attack, which has no significant
   chance of success if the block cipher is a pseudo-random permutation.

   To simplify implementation we use the same block cipher key for the
   encryption and authentication functions.  In our design this is not a
   problem.  All the A blocks are different, and they are different from
   the B_0 block.  If the block cipher behaves like a random
   permutation, then the outputs are independent of each other, up to
   the insignificant limitation that they are all different.  The only
   cases where the inputs to the block cipher can overlap are an
   intermediate value in the CBC-MAC and one of the other encryptions.
   As all the intermediate values of the CBC-MAC computation are
   essentially random (because the block cipher behaves like a random
   permutation) the probability of such a collision is very small.  Even
   if there is a collision, these values only affect T, which is
   encrypted so that an attacker cannot deduce any information, or
   detect any collision.

Care has been taken to ensure that the blocks used by the
authentication function match up with the blocks used by the
encryption function.  This should simplify hardware implementations,
and reduce the amount of byte-shifting required by software
implementations.

5.  Nonce Suggestions

The main requirement is that, within the scope of a single key, the
nonce values are unique for each message.  A common technique is to
number messages sequentially, and to use this number as the nonce.
Sequential message numbers are also used to detect replay attacks and
to detect message reordering, so in many situations (such as IPsec
ESP [ESP]) the sequence numbers are already available.

Users of CCM, and all other block cipher modes, should be aware of
precomputation attacks.  These are effectively collision attacks on
the cipher key.  Let us suppose the key K is 128 bits, and the same
nonce value N' is used with many different keys.  The attacker
chooses a particular nonce N'.  She chooses $2^{64}$ different keys at
random and computes a table entry for each K value, generating a pair
of the form $(K,S\_1)$.  (Given the key and the nonce, computing $S\_1$ is
easy.)  She then waits for messages to be sent with nonce N'.  We
will assume the first 16 bytes of each message are known so that she
can compute $S\_1$ for each message.  She looks in her table for a pair
with a matching $S\_1$ value.  She can expect to find a match after
checking about $2^{64}$ messages.  Once a match is found, the other part
of the matched pair is the key in question.  The total workload of
the attacker is only $2^{64}$ steps, rather than the expected $2^{128}$
steps.  Similar precomputation attacks exist for all block cipher
modes.

The main weapon against precomputation attacks is to use a larger
key.  Using a 256-bit key forces the attacker to perform at least
$2^{128}$ precomputations, which is infeasible.  In situations where
using a large key is not possible or desirable (for example, due to
the resulting performance impact), users can use part of the nonce to
reduce the number of times any specific nonce value is used with
different keys.  If there is room in the nonce, the sender could add
a few random bytes, and send these random bytes along with the
message.  This makes the precomputation attack much harder, as the
attacker now has to precompute a table for each of the possible
random values.  An alternative is to use something like the sender's
Ethernet address.  Note that due to the widespread use of DHCP and
NAT, IP addresses are rarely unique.  Including the Ethernet address
forces the attacker to perform the precomputation specifically for a
specific source address, and the resulting table could not be used to
attack anyone else.  Although these solutions can all work, they need

careful analysis and almost never entirely prevent these attacks.
Where possible, we recommend using a larger key, as this solves all
the problems.

6.  Efficiency and Performance

Performance depends on the speed of the block cipher implementation.
In hardware, for large packets, the speed achievable for CCM is
roughly the same as that achievable with the CBC encryption mode.

Encrypting and authenticating an empty message, without any
additional authentication data, requires two block cipher encryption
operations.  For each block of additional authentication data one
additional block cipher encryption operation is required (if one
includes the length encoding).  Each message block requires two block
cipher encryption operations.  The worst-case situation is when both
the message and the additional authentication data are a single
octet.  In this case, CCM requires five block cipher encryption
operations.

CCM results in the minimal possible message expansion; the only bits
added are the authentication bits.

Both the CCM encryption and CCM decryption operations require only
the block cipher encryption function.  In AES, the encryption and
decryption algorithms have some significant differences.  Thus, using
only the encrypt operation can lead to a significant savings in code
size or hardware size.

In hardware, CCM can compute the message authentication code and
perform encryption in a single pass.  That is, the implementation
does not have to complete calculation of the message authentication
code before encryption can begin.

CCM was designed for use in the packet processing environment.  The
authentication processing requires the message length to be known at
the beginning of the operation, which makes one-pass processing
difficult in some environments.  However, in almost all environments,
message or packet lengths are known in advance.

7.  Summary of Properties

   Security Function
      authenticated encryption

   Error Propagation
      none

   Synchronization
      same nonce used by sender and recipient

   Parallelizability
      encryption can be parallelized, but authentication cannot

   Keying Material Requirements
      one key

   Counter/IV/Nonce Requirements
      counter and nonce are part of the counter block

   Memory Requirements
      requires memory for encrypt operation of the underlying block
      cipher, plaintext, ciphertext (expanded for CBC-MAC), and a per-
      packet counter (an integer; at most L octets in size)

   Pre-processing Capability
      encryption key stream can be precomputed, but authentication
      cannot

   Message Length Requirements
      octet aligned message of arbitrary length, up to $2^{(8*L)}$ octets,
      and octet aligned arbitrary additional authenticated data, up to
      $2^{64}$ octets

   Ciphertext Expansion
      4, 6, 8, 10, 12, 14, or 16 octets depending on size of MAC
      selected

8.  Test Vectors

   These test vectors use AES for the block cipher [AES].  In each of
   these test vectors, the least significant sixteen bits of the counter
   block is used for the block counter, and the nonce is 13 octets.
   Some of the test vectors include a eight octet authentication value,
   and others include a ten octet authentication value.

```
=============== Packet Vector #1 ==================
AES Key =   C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =     00 00 00 03  02 01 00 A0  A1 A2 A3 A4  A5
Total packet length = 31. [Input with 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: 59 00 00 00  03 02 01 00  A0 A1 A2 A3  A4 A5 00 17
CBC IV out:EB 9D 55 47  73 09 55 AB  23 1E 0A 2D  FE 4B 90 D6
After xor: EB 95 55 46  71 0A 51 AE  25 19 0A 2D  FE 4B 90 D6   [hdr]
After AES: CD B6 41 1E  3C DC 9B 4F  5D 92 58 B6  9E E7 F0 91
After xor: C5 BF 4B 15  30 D1 95 40  4D 83 4A A5  8A F2 E6 86   [msg]
After AES: 9C 38 40 5E  A0 3C 1B C9  04 B5 8B 40  C7 6C A2 EB
After xor: 84 21 5A 45  BC 21 05 C9  04 B5 8B 40  C7 6C A2 EB   [msg]
After AES: 2D C6 97 E4  11 CA 83 A8  60 C2 C4 06  CC AA 54 2F
CBC-MAC  : 2D C6 97 E4  11 CA 83 A8
CTR Start: 01 00 00 00  03 02 01 00  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: 50 85 9D 91  6D CB 6D DD  E0 77 C2 D1  D4 EC 9F 97
CTR[0002]: 75 46 71 7A  C6 DE 9A FF  64 0C 9C 06  DE 6D 0D 8F
CTR[MAC ]: 3A 2E 46 C8  EC 33 A5 48
Total packet length = 39. [Authenticated and Encrypted Output]
            00 01 02 03  04 05 06 07  58 8C 97 9A  61 C6 63 D2
            F0 66 D0 C2  C0 F9 89 80  6D 5F 6B 61  DA C3 84 17
            E8 D1 2C FD  F9 26 E0


=============== Packet Vector #2 ==================
AES Key =   C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =     00 00 00 04  03 02 01 A0  A1 A2 A3 A4  A5
Total packet length = 32. [Input with 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: 59 00 00 00  04 03 02 01  A0 A1 A2 A3  A4 A5 00 18
CBC IV out:F0 C2 54 D3  CA 03 E2 39  70 BD 24 A8  4C 39 9E 77
After xor: F0 CA 54 D2  C8 00 E6 3C  76 BA 24 A8  4C 39 9E 77   [hdr]
After AES: 48 DE 8B 86  28 EA 4A 40  00 AA 42 C2  95 BF 4A 8C
After xor: 40 D7 81 8D  24 E7 44 4F  10 BB 50 D1  81 AA 5C 9B   [msg]
After AES: 0F 89 FF BC  A6 2B C2 4F  13 21 5F 16  87 96 AA 33
After xor: 17 90 E5 A7  BA 36 DC 50  13 21 5F 16  87 96 AA 33   [msg]
After AES: F7 B9 05 6A  86 92 6C F3  FB 16 3D C4  99 EF AA 11
CBC-MAC  : F7 B9 05 6A  86 92 6C F3
CTR Start: 01 00 00 00  04 03 02 01  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: 7A C0 10 3D  ED 38 F6 C0  39 0D BA 87  1C 49 91 F4
CTR[0002]: D4 0C DE 22  D5 F9 24 24  F7 BE 9A 56  9D A7 9F 51
CTR[MAC ]: 57 28 D0 04  96 D2 65 E5
Total packet length = 40. [Authenticated and Encrypted Output]
            00 01 02 03  04 05 06 07  72 C9 1A 36  E1 35 F8 CF
            29 1C A8 94  08 5C 87 E3  CC 15 C4 39  C9 E4 3A 3B
            A0 91 D5 6E  10 40 09 16
```

```
      =============== Packet Vector #3 =================
      AES Key =  C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
      Nonce =    00 00 00 05  04 03 02 A0  A1 A2 A3 A4  A5
      Total packet length = 33. [Input with 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: 59 00 00 00  05 04 03 02  A0 A1 A2 A3  A4 A5 00 19
      CBC IV out:6F 8A 12 F7  BF 8D 4D C5  A1 19 6E 95  DF F0 B4 27
      After xor: 6F 82 12 F6  BD 8E 49 C0  A7 1E 6E 95  DF F0 B4 27   [hdr]
      After AES: 37 E9 B7 8C  C2 20 17 E7  33 80 43 0C  BE F4 28 24
      After xor: 3F E0 BD 87  CE 2D 19 E8  23 91 51 1F  AA E1 3E 33   [msg]
      After AES: 90 CA 05 13  9F 4D 4E CF  22 6F E9 81  C5 9E 2D 40
      After xor: 88 D3 1F 08  83 50 50 D0  02 6F E9 81  C5 9E 2D 40   [msg]
      After AES: 73 B4 67 75  C0 26 DE AA  41 03 97 D6  70 FE 5F B0
      CBC-MAC  : 73 B4 67 75  C0 26 DE AA
      CTR Start: 01 00 00 00  05 04 03 02  A0 A1 A2 A3  A4 A5 00 01
      CTR[0001]: 59 B8 EF FF  46 14 73 12  B4 7A 1D 9D  39 3D 3C FF
      CTR[0002]: 69 F1 22 A0  78 C7 9B 89  77 89 4C 99  97 5C 23 78
      CTR[MAC ]: 39 6E C0 1A  7D B9 6E 6F
      Total packet length = 41. [Authenticated and Encrypted Output]
                 00 01 02 03  04 05 06 07  51 B1 E5 F4  4A 19 7D 1D
                 A4 6B 0F 8E  2D 28 2A E8  71 E8 38 BB  64 DA 85 96
                 57 4A DA A7  6F BD 9F B0  C5

      =============== Packet Vector #4 =================
      AES Key =  C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
      Nonce =    00 00 00 06  05 04 03 A0  A1 A2 A3 A4  A5
      Total packet length = 31. [Input with 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: 59 00 00 00  06 05 04 03  A0 A1 A2 A3  A4 A5 00 13
      CBC IV out:06 65 2C 60  0E F5 89 63  CA C3 25 A9  CD 3E 2B E1
      After xor: 06 69 2C 61  0C F6 8D 66  CC C4 2D A0  C7 35 2B E1   [hdr]
      After AES: A0 75 09 AC  15 C2 58 86  04 2F 80 60  54 FE A6 86
      After xor: AC 78 07 A3  05 D3 4A 95  10 3A 96 77  4C E7 BC 9D   [msg]
      After AES: 64 4C 09 90  D9 1B 83 E9  AB 4B 8E ED  06 6F F5 BF
      After xor: 78 51 17 90  D9 1B 83 E9  AB 4B 8E ED  06 6F F5 BF   [msg]
      After AES: 4B 4F 4B 39  B5 93 E6 BF  B0 B2 C2 B7  0F 29 CD 7A
      CBC-MAC  : 4B 4F 4B 39  B5 93 E6 BF
      CTR Start: 01 00 00 00  06 05 04 03  A0 A1 A2 A3  A4 A5 00 01
      CTR[0001]: AE 81 66 6A  83 8B 88 6A  EE BF 4A 5B  32 84 50 8A
      CTR[0002]: D1 B1 92 06  AC 93 9E 2F  B6 DD CE 10  A7 74 FD 8D
      CTR[MAC ]: DD 87 2A 80  7C 75 F8 4E
      Total packet length = 39. [Authenticated and Encrypted Output]
                 00 01 02 03  04 05 06 07  08 09 0A 0B  A2 8C 68 65
                 93 9A 9A 79  FA AA 5C 4C  2A 9D 4A 91  CD AC 8C 96
                 C8 61 B9 C9  E6 1E F1
```

```
=============== Packet Vector #5 ==================
AES Key =    C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =      00 00 00 07  06 05 04 A0  A1 A2 A3 A4  A5
Total packet length = 32. [Input with 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: 59 00 00 00  07 06 05 04  A0 A1 A2 A3  A4 A5 00 14
CBC IV out:00 4C 50 95  45 80 3C 48  51 CD E1 3B  56 C8 9A 85
After xor: 00 40 50 94  47 83 38 4D  57 CA E9 32  5C C3 9A 85   [hdr]
After AES: E2 B8 F7 CE  49 B2 21 72  84 A8 EA 84  FA AD 67 5C
After xor: EE B5 F9 C1  59 A3 33 61  90 BD FC 93  E2 B4 7D 47   [msg]
After AES: 3E FB 36 72  25 DB 11 01  D3 C2 2F 0E  CA FF 44 F3
After xor: 22 E6 28 6D  25 DB 11 01  D3 C2 2F 0E  CA FF 44 F3   [msg]
After AES: 48 B9 E8 82  55 05 4A B5  49 0A 95 F9  34 9B 4B 5E
CBC-MAC  : 48 B9 E8 82  55 05 4A B5
CTR Start: 01 00 00 00  07 06 05 04  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: D0 FC F5 74  4D 8F 31 E8  89 5B 05 05  4B 7C 90 C3
CTR[0002]: 72 A0 D4 21  9F 0D E1 D4  04 83 BC 2D  3D 0C FC 2A
CTR[MAC ]: 19 51 D7 85  28 99 67 26
Total packet length = 40. [Authenticated and Encrypted Output]
             00 01 02 03  04 05 06 07  08 09 0A 0B  DC F1 FB 7B
             5D 9E 23 FB  9D 4E 13 12  53 65 8A D8  6E BD CA 3E
             51 E8 3F 07  7D 9C 2D 93

=============== Packet Vector #6 ==================
AES Key =    C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =      00 00 00 08  07 06 05 A0  A1 A2 A3 A4  A5
Total packet length = 33. [Input with 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: 59 00 00 00  08 07 06 05  A0 A1 A2 A3  A4 A5 00 15
CBC IV out:04 72 DA 4C  6F F6 0A 63  06 52 1A 06  04 80 CD E5
After xor: 04 7E DA 4D  6D F5 0E 66  00 55 12 0F  0E 8B CD E5   [hdr]
After AES: 64 4C 36 A5  A2 27 37 62  0B 89 F1 D7  BF F2 73 D4
After xor: 68 41 38 AA  B2 36 25 71  1F 9C E7 C0  A7 EB 69 CF   [msg]
After AES: 41 E1 19 CD  19 24 CE 77  F1 2F A6 60  C1 6E BB 4E
After xor: 5D FC 07 D2  39 24 CE 77  F1 2F A6 60  C1 6E BB 4E   [msg]
After AES: A5 27 D8 15  6A C3 59 BF  1C B8 86 E6  2F 29 91 29
CBC-MAC  : A5 27 D8 15  6A C3 59 BF
CTR Start: 01 00 00 00  08 07 06 05  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: 63 CC BE 1E  E0 17 44 98  45 64 B2 3A  8D 24 5C 80
CTR[0002]: 39 6D BA A2  A7 D2 CB D4  B5 E1 7C 10  79 45 BB C0
CTR[MAC ]: E5 7D DC 56  C6 52 92 2B
Total packet length = 41. [Authenticated and Encrypted Output]
             00 01 02 03  04 05 06 07  08 09 0A 0B  6F C1 B0 11
             F0 06 56 8B  51 71 A4 2D  95 3D 46 9B  25 70 A4 BD
             87 40 5A 04  43 AC 91 CB  94
```

```
=============== Packet Vector #7 ==================
AES Key =  C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =    00 00 00 09  08 07 06 A0  A1 A2 A3 A4  A5
Total packet length = 31. [Input with 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: 61 00 00 00  09 08 07 06  A0 A1 A2 A3  A4 A5 00 17
CBC IV out:60 06 C5 72  DA 23 9C BF  A0 5B 0A DE  D2 CD A8 1E
After xor: 60 0E C5 73  D8 20 98 BA  A6 5C 0A DE  D2 CD A8 1E   [hdr]
After AES: 41 7D E2 AE  94 E2 EA D9  00 FC 44 FC  D0 69 52 27
After xor: 49 74 E8 A5  98 EF E4 D6  10 ED 56 EF  C4 7C 44 30   [msg]
After AES: 2A 6C 42 CA  49 D7 C7 01  C5 7D 59 FF  87 16 49 0E
After xor: 32 75 58 D1  55 CA D9 01  C5 7D 59 FF  87 16 49 0E   [msg]
After AES: 89 8B D6 45  4E 27 20 BB  D2 7E F3 15  7A 7C 90 B2
CBC-MAC  : 89 8B D6 45  4E 27 20 BB  D2 7E
CTR Start: 01 00 00 00  09 08 07 06  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: 09 3C DB B9  C5 52 4F DA  C1 C5 EC D2  91 C4 70 AF
CTR[0002]: 11 57 83 86  E2 C4 72 B4  8E CC 8A AD  AB 77 6F CB
CTR[MAC ]: 8D 07 80 25  62 B0 8C 00  A6 EE
Total packet length = 41. [Authenticated and Encrypted Output]
           00 01 02 03  04 05 06 07  01 35 D1 B2  C9 5F 41 D5
           D1 D4 FE C1  85 D1 66 B8  09 4E 99 9D  FE D9 6C 04
           8C 56 60 2C  97 AC BB 74  90

=============== Packet Vector #8 ==================
AES Key =  C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =    00 00 00 0A  09 08 07 A0  A1 A2 A3 A4  A5
Total packet length = 32. [Input with 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: 61 00 00 00  0A 09 08 07  A0 A1 A2 A3  A4 A5 00 18
CBC IV out:63 A3 FA E4  6C 79 F3 FA  78 38 B8 A2  80 36 B6 0B
After xor: 63 AB FA E5  6E 7A F7 FF  7E 3F B8 A2  80 36 B6 0B   [hdr]
After AES: 1C 99 1A 3D  B7 60 79 27  34 40 79 1F  AD 8B 5B 02
After xor: 14 90 10 36  BB 6D 77 28  24 51 6B 0C  B9 9E 4D 15   [msg]
After AES: 14 19 E8 E8  CB BE 75 58  E1 E3 BE 4B  6C 9F 82 E3
After xor: 0C 00 F2 F3  D7 A3 6B 47  E1 E3 BE 4B  6C 9F 82 E3   [msg]
After AES: E0 16 E8 1C  7F 7B 8A 38  A5 38 F2 CB  5B B6 C1 F2
CBC-MAC  : E0 16 E8 1C  7F 7B 8A 38  A5 38
CTR Start: 01 00 00 00  0A 09 08 07  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: 73 7C 33 91  CC 8E 13 DD  E0 AA C5 4B  6D B7 EB 98
CTR[0002]: 74 B7 71 77  C5 AA C5 3B  04 A4 F8 70  8E 92 EB 2B
CTR[MAC ]: 21 6D AC 2F  8B 4F 1C 07  91 8C
Total packet length = 42. [Authenticated and Encrypted Output]
           00 01 02 03  04 05 06 07  7B 75 39 9A  C0 83 1D D2
           F0 BB D7 58  79 A2 FD 8F  6C AE 6B 6C  D9 B7 DB 24
           C1 7B 44 33  F4 34 96 3F  34 B4
```

```
=============== Packet Vector #9 ==================
AES Key =   C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =     00 00 00 0B  0A 09 08 A0  A1 A2 A3 A4  A5
Total packet length = 33. [Input with 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: 61 00 00 00  0B 0A 09 08  A0 A1 A2 A3  A4 A5 00 19
CBC IV out:4F 2C 86 11  1E 08 2A DD  6B 44 21 3A  B5 13 13 16
After xor: 4F 24 86 10  1C 0B 2E D8  6D 43 21 3A  B5 13 13 16  [hdr]
After AES: F6 EC 56 87  3C 57 12 DC  9C C5 3C A8  D4 D1 ED 0A
After xor: FE E5 5C 8C  30 5A 1C D3  8C D4 2E BB  C0 C4 FB 1D  [msg]
After AES: 17 C1 80 A5  31 53 D4 C3  03 85 0C 95  65 80 34 52
After xor: 0F D8 9A BE  2D 4E CA DC  23 85 0C 95  65 80 34 52  [msg]
After AES: 46 A1 F6 E2  B1 6E 75 F8  1C F5 6B 1A  80 04 44 1B
CBC-MAC  : 46 A1 F6 E2  B1 6E 75 F8  1C F5
CTR Start: 01 00 00 00  0B 0A 09 08  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: 8A 5A 10 6B  C0 29 9A 55  5B 93 6B 0B  0E A0 DE 5A
CTR[0002]: EA 05 FD E2  AB 22 5C FE  B7 73 12 CB  88 D9 A5 4A
CTR[MAC ]: AC 3D F1 07  DA 30 C4 86  43 BB
Total packet length = 43. [Authenticated and Encrypted Output]
            00 01 02 03  04 05 06 07  82 53 1A 60  CC 24 94 5A
            4B 82 79 18  1A B5 C8 4D  F2 1C E7 F9  B7 3F 42 E1
            97 EA 9C 07  E5 6B 5E B1  7E 5F 4E

=============== Packet Vector #10 ==================
AES Key =   C0 C1 C2 C3  C4 C5 C6 C7  C8 C9 CA CB  CC CD CE CF
Nonce =     00 00 00 0C  0B 0A 09 A0  A1 A2 A3 A4  A5
Total packet length = 31. [Input with 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: 61 00 00 00  0C 0B 0A 09  A0 A1 A2 A3  A4 A5 00 13
CBC IV out:7F B8 0A 32  E9 80 57 46  EC 31 6C 3A  B2 A2 EB 5D
After xor: 7F B4 0A 33  EB 83 53 43  EA 36 64 33  B8 A9 EB 5D  [hdr]
After AES: 7E 96 96 BF  F1 56 D6 A8  6E AC F5 7B  7F 23 47 5A
After xor: 72 9B 98 B0  E1 47 C4 BB  7A B9 E3 6C  67 3A 5D 41  [msg]
After AES: 8B 4A EE 42  04 24 8A 59  FA CC 88 66  57 66 DD 72
After xor: 97 57 F0 42  04 24 8A 59  FA CC 88 66  57 66 DD 72  [msg]
After AES: 41 63 89 36  62 ED D7 EB  CD 6E 15 C1  89 48 62 05
CBC-MAC  : 41 63 89 36  62 ED D7 EB  CD 6E
CTR Start: 01 00 00 00  0C 0B 0A 09  A0 A1 A2 A3  A4 A5 00 01
CTR[0001]: 0B 39 2B 9B  05 66 97 06  3F 12 56 8F  2B 13 A1 0F
CTR[0002]: 07 89 65 25  23 40 94 3B  9E 69 B2 56  CC 5E F7 31
CTR[MAC ]: 17 09 20 76  09 A0 4E 72  45 B3
Total packet length = 41. [Authenticated and Encrypted Output]
            00 01 02 03  04 05 06 07  08 09 0A 0B  07 34 25 94
            15 77 85 15  2B 07 40 98  33 0A BB 14  1B 94 7B 56
            6A A9 40 6B  4D 99 99 88  DD
```

```
=============== Packet Vector #11 ==================
AES Key =   C0 C1 C2 C3   C4 C5 C6 C7   C8 C9 CA CB   CC CD CE CF
Nonce =     00 00 00 0D   0C 0B 0A A0   A1 A2 A3 A4   A5
Total packet length = 32. [Input with 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: 61 00 00 00   0D 0C 0B 0A   A0 A1 A2 A3   A4 A5 00 14
CBC IV out:B0 84 85 79   51 D2 FA 42   76 EF 3A D7   14 B9 62 87
After xor: B0 88 85 78   53 D1 FE 47   70 E8 32 DE   1E B2 62 87   [hdr]
After AES: C9 B3 64 7E   D8 79 2A 5C   65 B7 CE CC   19 0A 97 0A
After xor: C5 BE 6A 71   C8 68 38 4F   71 A2 D8 DB   01 13 8D 11   [msg]
After AES: 34 0F 69 17   FA B9 19 D6   1D AC D0 35   36 D6 55 8B
After xor: 28 12 77 08   FA B9 19 D6   1D AC D0 35   36 D6 55 8B   [msg]
After AES: 6B 5E 24 34   12 CC C2 AD   6F 1B 11 C3   A1 A9 D8 BC
CBC-MAC  : 6B 5E 24 34   12 CC C2 AD   6F 1B
CTR Start: 01 00 00 00   0D 0C 0B 0A   A0 A1 A2 A3   A4 A5 00 01
CTR[0001]: 6B 66 BC 0C   90 A1 F1 12   FC BE 6F 4E   12 20 77 BC
CTR[0002]: 97 9E 57 2B   BE 65 8A E5   CC 20 11 83   2A 9A 9B 5B
CTR[MAC ]: 9E 64 86 DD   02 B6 49 C1   6D 37
Total packet length = 42. [Authenticated and Encrypted Output]
            00 01 02 03   04 05 06 07   08 09 0A 0B   67 6B B2 03
            80 B0 E3 01   E8 AB 79 59   0A 39 6D A7   8B 83 49 34
            F5 3A A2 E9   10 7A 8B 6C   02 2C

=============== Packet Vector #12 ==================
AES Key =   C0 C1 C2 C3   C4 C5 C6 C7   C8 C9 CA CB   CC CD CE CF
Nonce =     00 00 00 0E   0D 0C 0B A0   A1 A2 A3 A4   A5
Total packet length = 33. [Input with 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: 61 00 00 00   0E 0D 0C 0B   A0 A1 A2 A3   A4 A5 00 15
CBC IV out:5F 8E 8D 02   AD 95 7C 5A   36 14 CF 63   40 16 97 4F
After xor: 5F 82 8D 03   AF 96 78 5F   30 13 C7 6A   4A 1D 97 4F   [hdr]
After AES: 63 FA BD 69   B9 55 65 FF   54 AA F4 60   88 7D EC 9F
After xor: 6F F7 B3 66   A9 44 77 EC   40 BF E2 77   90 64 F6 84   [msg]
After AES: 5A 76 5F 0B   93 CE 4F 6A   B4 1D 91 30   18 57 6A D7
After xor: 46 6B 41 14   B3 CE 4F 6A   B4 1D 91 30   18 57 6A D7   [msg]
After AES: 9D 66 92 41   01 08 D5 B6   A1 45 85 AC   AF 86 32 E8
CBC-MAC  : 9D 66 92 41   01 08 D5 B6   A1 45
CTR Start: 01 00 00 00   0E 0D 0C 0B   A0 A1 A2 A3   A4 A5 00 01
CTR[0001]: CC F2 AE D9   E0 4A C9 74   E6 58 55 B3   2B 94 30 BF
CTR[0002]: A2 CA AC 11   63 F4 07 E5   E5 F6 E3 B3   79 0F 79 F8
CTR[MAC ]: 50 7C 31 57   63 EF 78 D3   77 9E
Total packet length = 43. [Authenticated and Encrypted Output]
            00 01 02 03   04 05 06 07   08 09 0A 0B   C0 FF A0 D6
            F0 5B DB 67   F2 4D 43 A4   33 8D 2A A4   BE D7 B2 0E
            43 CD 1A A3   16 62 E7 AD   65 D6 DB
```

```
       =============== Packet Vector #13 ==================
       AES Key =  D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
       Nonce =    00 41 2B 4E  A9 CD BE 3C  96 96 76 6C  FA
       Total packet length = 31. [Input with 8 cleartext header octets]
                  0B E1 A8 8B  AC E0 18 B1  08 E8 CF 97  D8 20 EA 25
                  84 60 E9 6A  D9 CF 52 89  05 4D 89 5C  EA C4 7C
       CBC IV in: 59 00 41 2B  4E A9 CD BE  3C 96 96 76  6C FA 00 17
       CBC IV out:33 AE C3 1A  1F B7 CC 35  E5 DA D2 BA  C0 90 D9 A3
       After xor: 33 A6 C8 FB  B7 3C 60 D5  FD 6B D2 BA  C0 90 D9 A3   [hdr]
       After AES: B7 56 CA 1E  5B 42 C6 9C  58 E3 0A F5  2B F7 7C FD
       After xor: BF BE 05 89  83 62 2C B9  DC 83 E3 9F  F2 38 2E 74   [msg]
       After AES: 33 3D 3A 3D  07 B5 3C 7B  22 0E 96 1A  18 A9 A1 9E
       After xor: 36 70 B3 61  ED 71 40 7B  22 0E 96 1A  18 A9 A1 9E   [msg]
       After AES: 14 BD DB 6B  F9 01 63 4D  FB 56 51 83  BC 74 93 F7
       CBC-MAC  : 14 BD DB 6B  F9 01 63 4D
       CTR Start: 01 00 41 2B  4E A9 CD BE  3C 96 96 76  6C FA 00 01
       CTR[0001]: 44 51 B0 11  7A 84 82 BF  03 19 AE C1  59 5E BD DA
       CTR[0002]: 83 EB 76 E1  3A 44 84 7F  92 20 09 07  76 B8 25 C5
       CTR[MAC ]: F3 31 2C A0  F5 DC B4 FE
       Total packet length = 39. [Authenticated and Encrypted Output]
                  0B E1 A8 8B  AC E0 18 B1  4C B9 7F 86  A2 A4 68 9A
                  87 79 47 AB  80 91 EF 53  86 A6 FF BD  D0 80 F8 E7
                  8C F7 CB 0C  DD D7 B3

       =============== Packet Vector #14 ==================
       AES Key =  D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
       Nonce =    00 33 56 8E  F7 B2 63 3C  96 96 76 6C  FA
       Total packet length = 32. [Input with 8 cleartext header octets]
                  63 01 8F 76  DC 8A 1B CB  90 20 EA 6F  91 BD D8 5A
                  FA 00 39 BA  4B AF F9 BF  B7 9C 70 28  94 9C D0 EC
       CBC IV in: 59 00 33 56  8E F7 B2 63  3C 96 96 76  6C FA 00 18
       CBC IV out:42 0D B1 50  BB 0C 44 DA  83 E4 52 09  55 99 67 E3
       After xor: 42 05 D2 51  34 7A 98 50  98 2F 52 09  55 99 67 E3   [hdr]
       After AES: EA D1 CA 56  02 02 09 5C  E6 12 B0 D2  18 A0 DD 44
       After xor: 7A F1 20 39  93 BF D1 06  1C 12 89 68  53 0F 24 FB   [msg]
       After AES: 51 77 41 69  C3 DE 6B 24  13 27 74 90  F5 FF C5 62
       After xor: E6 EB 31 41  57 42 BB C8  13 27 74 90  F5 FF C5 62   [msg]
       After AES: D4 CC 3B 82  DF 9F CC 56  7E E5 83 61  D7 8D FB 5E
       CBC-MAC  : D4 CC 3B 82  DF 9F CC 56
       CTR Start: 01 00 33 56  8E F7 B2 63  3C 96 96 76  6C FA 00 01
       CTR[0001]: DC EB F4 13  38 3C 66 A0  5A 72 55 EF  98 D7 FF AD
       CTR[0002]: 2F 54 2C BA  15 D6 6C DF  E1 EC 46 8F  0E 68 A1 24
       CTR[MAC ]: 11 E2 D3 9F  A2 E8 0C DC
       Total packet length = 40. [Authenticated and Encrypted Output]
                  63 01 8F 76  DC 8A 1B CB  4C CB 1E 7C  A9 81 BE FA
                  A0 72 6C 55  D3 78 06 12  98 C8 5C 92  81 4A BC 33
                  C5 2E E8 1D  7D 77 C0 8A
```

```
=============== Packet Vector #15 ==================
AES Key =   D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =     00 10 3F E4  13 36 71 3C  96 96 76 6C  FA
Total packet length = 33. [Input with 8 cleartext header octets]
            AA 6C FA 36  CA E8 6B 40  B9 16 E0 EA  CC 1C 00 D7
            DC EC 68 EC  0B 3B BB 1A  02 DE 8A 2D  1A A3 46 13
            2E
CBC IV in: 59 00 10 3F  E4 13 36 71  3C 96 96 76  6C FA 00 19
CBC IV out:B3 26 49 FF  D5 9F 56 0F  02 2D 11 E2  62 C5 BE EA
After xor: B3 2E E3 93  2F A9 9C E7  69 6D 11 E2  62 C5 BE EA   [hdr]
After AES: 82 50 9E E5  B2 FF DB CA  9B D0 2E 20  6B 3F B7 AD
After xor: 3B 46 7E 0F  7E E3 DB 1D  47 3C 46 CC  60 04 0C B7   [msg]
After AES: 80 46 0E 4C  08 3A D0 3F  B9 A9 13 BE  E4 DE 2F 66
After xor: 82 98 84 61  12 99 96 2C  97 A9 13 BE  E4 DE 2F 66   [msg]
After AES: 47 29 CB 00  31 F1 81 C1  92 68 4B 89  A4 71 50 E7
CBC-MAC  : 47 29 CB 00  31 F1 81 C1
CTR Start: 01 00 10 3F  E4 13 36 71  3C 96 96 76  6C FA 00 01
CTR[0001]: 08 C4 DA C8  EC C1 C0 7B  4C E1 F2 4C  37 5A 47 EE
CTR[0002]: A7 87 2E 6C  6D C4 4E 84  26 02 50 4C  3F A5 73 C5
CTR[MAC ]: E0 5F B2 6E  EA 83 B4 C7
Total packet length = 41. [Authenticated and Encrypted Output]
            AA 6C FA 36  CA E8 6B 40  B1 D2 3A 22  20 DD C0 AC
            90 0D 9A A0  3C 61 FC F4  A5 59 A4 41  77 67 08 97
            08 A7 76 79  6E DB 72 35  06

=============== Packet Vector #16 ==================
AES Key =   D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =     00 76 4C 63  B8 05 8E 3C  96 96 76 6C  FA
Total packet length = 31. [Input with 12 cleartext header octets]
            D0 D0 73 5C  53 1E 1B EC  F0 49 C2 44  12 DA AC 56
            30 EF A5 39  6F 77 0C E1  A6 6B 21 F7  B2 10 1C
CBC IV in: 59 00 76 4C  63 B8 05 8E  3C 96 96 76  6C FA 00 13
CBC IV out:AB DC 4E C9  AA 72 33 97  DF 2D AD 76  33 DE 3B 0D
After xor: AB D0 9E 19  D9 2E 60 89  C4 C1 5D 3F  F1 9A 3B 0D   [hdr]
After AES: 62 86 F6 2F  23 42 63 B0  1C FD 8C 37  40 74 81 EB
After xor: 70 5C 5A 79  13 AD C6 89  73 8A 80 D6  E6 1F A0 1C   [msg]
After AES: 88 95 84 18  CF 79 CA BE  EB C0 0C C4  86 E6 01 F7
After xor: 3A 85 98 18  CF 79 CA BE  EB C0 0C C4  86 E6 01 F7   [msg]
After AES: C1 85 92 D9  84 CD 67 80  63 D1 D9 6D  C1 DF A1 11
CBC-MAC  : C1 85 92 D9  84 CD 67 80
CTR Start: 01 00 76 4C  63 B8 05 8E  3C 96 96 76  6C FA 00 01
CTR[0001]: 06 08 FF 95  A6 94 D5 59  F4 0B B7 9D  EF FA 41 DF
CTR[0002]: 80 55 3A 75  78 38 04 A9  64 8B 68 DD  7F DC DD 7A
CTR[MAC ]: 5B EA DB 4E  DF 07 B9 2F
Total packet length = 39. [Authenticated and Encrypted Output]
            D0 D0 73 5C  53 1E 1B EC  F0 49 C2 44  14 D2 53 C3
            96 7B 70 60  9B 7C BB 7C  49 91 60 28  32 45 26 9A
            6F 49 97 5B  CA DE AF
```

```
================ Packet Vector #17 ==================
AES Key = D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =     00 F8 B6 78  09 4E 3B 3C  96 96 76 6C  FA
Total packet length = 32. [Input with 12 cleartext header octets]
             77 B6 0F 01  1C 03 E1 52  58 99 BC AE  E8 8B 6A 46
             C7 8D 63 E5  2E B8 C5 46  EF B5 DE 6F  75 E9 CC 0D
CBC IV in: 59 00 F8 B6  78 09 4E 3B  3C 96 96 76  6C FA 00 14
CBC IV out:F4 68 FE 5D  B1 53 0B 7A  5A A5 FB 27  40 CF 6E 33
After xor: F4 64 89 EB  BE 52 17 79  BB F7 A3 BE  FC 61 6E 33   [hdr]
After AES: 23 29 0E 0B  33 45 9A 83  32 2D E4 06  86 67 10 04
After xor: CB A2 64 4D  F4 C8 F9 66  1C 95 21 40  69 D2 CE 6B   [msg]
After AES: 8F BE D4 0F  8B 89 B7 B8  20 D5 5F E0  3C E2 43 11
After xor: FA 57 18 02  8B 89 B7 B8  20 D5 5F E0  3C E2 43 11   [msg]
After AES: 6A DB 15 B6  71 81 B2 E2  2B E3 4A F2  B2 83 E2 29
CBC-MAC  : 6A DB 15 B6  71 81 B2 E2
CTR Start: 01 00 F8 B6  78 09 4E 3B  3C 96 96 76  6C FA 00 01
CTR[0001]: BD CE 95 5C  CF D3 81 0A  91 EA 77 A6  A4 5B C0 4C
CTR[0002]: 43 2E F2 32  AE 36 D8 92  22 BF 63 37  E6 B2 6C E8
CTR[MAC ]: 1C F7 19 C1  35 7F CC DE
Total packet length = 40. [Authenticated and Encrypted Output]
             77 B6 0F 01  1C 03 E1 52  58 99 BC AE  55 45 FF 1A
             08 5E E2 EF  BF 52 B2 E0  4B EE 1E 23  36 C7 3E 3F
             76 2C 0C 77  44 FE 7E 3C


================ Packet Vector #18 ==================
AES Key = D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =     00 D5 60 91  2D 3F 70 3C  96 96 76 6C  FA
Total packet length = 33. [Input with 12 cleartext header octets]
             CD 90 44 D2  B7 1F DB 81  20 EA 60 C0  64 35 AC BA
             FB 11 A8 2E  2F 07 1D 7C  A4 A5 EB D9  3A 80 3B A8
             7F
CBC IV in: 59 00 D5 60  91 2D 3F 70  3C 96 96 76  6C FA 00 15
CBC IV out:BA 37 74 54  D7 20 A4 59  25 97 F6 A3  D1 D6 BA 67
After xor: BA 3B B9 C4  93 F2 13 46  FE 16 D6 49  B1 16 BA 67   [hdr]
After AES: 81 6A 20 20  38 D0 A6 30  CB E0 B7 3C  39 BB CE 05
After xor: E5 5F 8C 9A  C3 C1 0E 1E  E4 E7 AA 40  9D 1E 25 DC   [msg]
After AES: 6D 5C 15 FD  85 2D 5C 3C  E3 03 3D 85  DA 57 BD AC
After xor: 57 DC 2E 55  FA 2D 5C 3C  E3 03 3D 85  DA 57 BD AC   [msg]
After AES: B0 4A 1C 23  BC 39 B6 51  76 FD 5B FF  9B C1 28 5E
CBC-MAC  : B0 4A 1C 23  BC 39 B6 51
CTR Start: 01 00 D5 60  91 2D 3F 70  3C 96 96 76  6C FA 00 01
CTR[0001]: 64 A2 C5 56  50 CE E0 4C  7A 93 D8 EE  F5 43 E8 8E
CTR[0002]: 18 E7 65 AC  B7 B0 E9 AF  09 2B D0 20  6C A1 C8 3C
CTR[MAC ]: F7 43 82 79  5C 49 F3 00
Total packet length = 41. [Authenticated and Encrypted Output]
             CD 90 44 D2  B7 1F DB 81  20 EA 60 C0  00 97 69 EC
             AB DF 48 62  55 94 C5 92  51 E6 03 57  22 67 5E 04
             C8 47 09 9E  5A E0 70 45  51
```

```
=============== Packet Vector #19 ==================
AES Key =  D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =    00 42 FF F8  F1 95 1C 3C  96 96 76 6C  FA
Total packet length = 31. [Input with 8 cleartext header octets]
           D8 5B C7 E6  9F 94 4F B8  8A 19 B9 50  BC F7 1A 01
           8E 5E 67 01  C9 17 87 65  98 09 D6 7D  BE DD 18
CBC IV in: 61 00 42 FF  F8 F1 95 1C  3C 96 96 76  6C FA 00 17
CBC IV out:44 F7 CC 9C  2B DD 2F 45  F6 38 25 6B  73 6E 1D 7A
After xor: 44 FF 14 C7  EC 3B B0 D1  B9 80 25 6B  73 6E 1D 7A   [hdr]
After AES: 57 C3 73 F8  00 AA 5F CC  7B CF 1D 1B  DD BB 4C 52
After xor: DD DA CA A8  BC 5D 45 CD  F5 91 7A 1A  14 AC CB 37   [msg]
After AES: 42 4E 93 72  72 C8 79 B6  11 C7 A5 9F  47 8D 9F D8
After xor: DA 47 45 0F  CC 15 61 B6  11 C7 A5 9F  47 8D 9F D8   [msg]
After AES: 9A CB 03 F8  B9 DB C8 D2  D2 D7 A4 B4  95 25 08 67
CBC-MAC  : 9A CB 03 F8  B9 DB C8 D2  D2 D7
CTR Start: 01 00 42 FF  F8 F1 95 1C  3C 96 96 76  6C FA 00 01
CTR[0001]: 36 38 34 FA  28 83 3D B7  55 66 0D 98  65 0D 68 46
CTR[0002]: 35 E9 63 54  87 16 72 56  3F 0C 08 AF  78 44 31 A9
CTR[MAC ]: F9 B7 FA 46  7B 9B 40 45  14 6D
Total packet length = 41. [Authenticated and Encrypted Output]
           D8 5B C7 E6  9F 94 4F B8  BC 21 8D AA  94 74 27 B6
           DB 38 6A 99  AC 1A EF 23  AD E0 B5 29  39 CB 6A 63
           7C F9 BE C2  40 88 97 C6  BA

=============== Packet Vector #20 ==================
AES Key =  D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =    00 92 0F 40  E5 6C DC 3C  96 96 76 6C  FA
Total packet length = 32. [Input with 8 cleartext header octets]
           74 A0 EB C9  06 9F 5B 37  17 61 43 3C  37 C5 A3 5F
           C1 F3 9F 40  63 02 EB 90  7C 61 63 BE  38 C9 84 37
CBC IV in: 61 00 92 0F  40 E5 6C DC  3C 96 96 76  6C FA 00 18
CBC IV out:60 CB 21 CE  40 06 50 AE  2A D2 BE 52  9F 5F 0F C2
After xor: 60 C3 55 6E  AB CF 56 31  71 E5 BE 52  9F 5F 0F C2   [hdr]
After AES: 03 20 64 14  35 32 5D 95  C8 A2 50 40  93 28 DA 9B
After xor: 14 41 27 28  02 F7 FE CA  09 51 CF 00  F0 2A 31 0B   [msg]
After AES: B9 E8 87 95  ED F7 F0 08  15 15 F0 14  E2 FE 0E 48
After xor: C5 89 E4 2B  D5 3E 74 3F  15 15 F0 14  E2 FE 0E 48   [msg]
After AES: 8F AD 0C 23  E9 63 7E 87  FA 21 45 51  1B 47 DE F1
CBC-MAC  : 8F AD 0C 23  E9 63 7E 87  FA 21
CTR Start: 01 00 92 0F  40 E5 6C DC  3C 96 96 76  6C FA 00 01
CTR[0001]: 4F 71 A5 C1  12 42 E3 7D  29 F0 FE E4  1B E1 02 5F
CTR[0002]: 34 2B D3 F1  7C B7 7B C1  79 0B 05 05  61 59 27 2C
CTR[MAC ]: 7F 09 7B EF  C6 AA C1 D3  73 65
Total packet length = 42. [Authenticated and Encrypted Output]
           74 A0 EB C9  06 9F 5B 37  58 10 E6 FD  25 87 40 22
           E8 03 61 A4  78 E3 E9 CF  48 4A B0 4F  44 7E FF F6
           F0 A4 77 CC  2F C9 BF 54  89 44
```

```
=============== Packet Vector #21 ==================
AES Key =   D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =     00 27 CA 0C  71 20 BC 3C  96 96 76 6C  FA
Total packet length = 33. [Input with 8 cleartext header octets]
            44 A3 AA 3A  AE 64 75 CA  A4 34 A8 E5  85 00 C6 E4
            15 30 53 88  62 D6 86 EA  9E 81 30 1B  5A E4 22 6B
            FA
CBC IV in: 61 00 27 CA  0C 71 20 BC  3C 96 96 76  6C FA 00 19
CBC IV out:43 07 C0 73  A8 9E E1 D5  05 27 B2 9A  62 48 D6 D2
After xor: 43 0F 84 D0  02 A4 4F B1  70 ED B2 9A  62 48 D6 D2   [hdr]
After AES: B6 0B C6 F5  84 01 75 BC  01 27 70 F1  11 8D 75 10
After xor: 12 3F 6E 10  01 01 B3 58  14 17 23 79  73 5B F3 FA   [msg]
After AES: 7D 5E 64 92  CE 2C B9 EA  7E 4C 4A 09  09 89 C8 FB
After xor: E3 DF 54 89  94 C8 9B 81  84 4C 4A 09  09 89 C8 FB   [msg]
After AES: 68 5F 8D 79  D2 2B 9B 74  21 DF 4C 3E  87 BA 0A AF
CBC-MAC  : 68 5F 8D 79  D2 2B 9B 74  21 DF
CTR Start: 01 00 27 CA  0C 71 20 BC  3C 96 96 76  6C FA 00 01
CTR[0001]: 56 8A 45 9E  40 09 48 67  EB 85 E0 9E  6A 2E 64 76
CTR[0002]: A6 00 AA 92  92 03 54 9A  AE EF 2C CC  59 13 7A 57
CTR[MAC ]: 25 1E DC DD  3F 11 10 F3  98 11
Total packet length = 43. [Authenticated and Encrypted Output]
            44 A3 AA 3A  AE 64 75 CA  F2 BE ED 7B  C5 09 8E 83
            FE B5 B3 16  08 F8 E2 9C  38 81 9A 89  C8 E7 76 F1
            54 4D 41 51  A4 ED 3A 8B  87 B9 CE

=============== Packet Vector #22 ==================
AES Key =   D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =     00 5B 8C CB  CD 9A F8 3C  96 96 76 6C  FA
Total packet length = 31. [Input with 12 cleartext header octets]
            EC 46 BB 63  B0 25 20 C3  3C 49 FD 70  B9 6B 49 E2
            1D 62 17 41  63 28 75 DB  7F 6C 92 43  D2 D7 C2
CBC IV in: 61 00 5B 8C  CB CD 9A F8  3C 96 96 76  6C FA 00 13
CBC IV out:91 14 AD 06  B6 CC 02 35  76 9A B6 14  C4 82 95 03
After xor: 91 18 41 40  0D AF B2 10  56 59 8A 5D  39 F2 95 03   [hdr]
After AES: 29 BD 7C 27  83 E3 E8 D3  C3 5C 01 F4  4C EC BB FA
After xor: 90 D6 35 C5  9E 81 FF 92  A0 74 74 2F  33 80 29 B9   [msg]
After AES: 4E DA F4 0D  21 0B D4 5F  FE 97 90 B9  AA EC 34 4C
After xor: 9C 0D 36 0D  21 0B D4 5F  FE 97 90 B9  AA EC 34 4C   [msg]
After AES: 21 9E F8 90  EA 64 C2 11  A5 37 88 83  E1 BA 22 0D
CBC-MAC  : 21 9E F8 90  EA 64 C2 11  A5 37
CTR Start: 01 00 5B 8C  CB CD 9A F8  3C 96 96 76  6C FA 00 01
CTR[0001]: 88 BC 19 42  80 C1 FA 3E  BE FC EF FB  4D C6 2D 54
CTR[0002]: 3E 59 7D A5  AE 21 CC A4  00 9E 4C 0C  91 F6 22 49
CTR[MAC ]: 5C BC 30 98  66 02 A9 F4  64 A0
Total packet length = 41. [Authenticated and Encrypted Output]
            EC 46 BB 63  B0 25 20 C3  3C 49 FD 70  31 D7 50 A0
            9D A3 ED 7F  DD D4 9A 20  32 AA BF 17  EC 8E BF 7D
            22 C8 08 8C  66 6B E5 C1  97
```

```
=============== Packet Vector #23 ==================
AES Key = D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =    00 3E BE 94  04 4B 9A 3C  96 96 76 6C  FA
Total packet length = 32. [Input with 12 cleartext header octets]
           47 A6 5A C7  8B 3D 59 42  27 E8 5E 71  E2 FC FB B8
           80 44 2C 73  1B F9 51 67  C8 FF D7 89  5E 33 70 76
CBC IV in: 61 00 3E BE  94 04 4B 9A  3C 96 96 76  6C FA 00 14
CBC IV out:0F 70 3F 5A  54 2C 44 6E  8B 74 A3 73  9B 48 B9 61
After xor: 0F 7C 78 FC  0E EB CF 53  D2 36 84 9B  C5 39 B9 61  [hdr]
After AES: 40 5B ED 29  D0 98 AE 91  DB 68 78 F3  68 B8 73 85
After xor: A2 A7 16 91  50 DC 82 E2  C0 91 29 94  A0 47 A4 0C  [msg]
After AES: 3D 03 29 3C  FD 81 1B 37  01 51 FB C7  85 6B 7A 74
After xor: 63 30 59 4A  FD 81 1B 37  01 51 FB C7  85 6B 7A 74  [msg]
After AES: 66 4F 27 16  3E 36 0F 72  62 0D 4E 67  7C E0 61 DE
CBC-MAC  : 66 4F 27 16  3E 36 0F 72  62 0D
CTR Start: 01 00 3E BE  94 04 4B 9A  3C 96 96 76  6C FA 00 01
CTR[0001]: 0A 7E 0A 63  53 C8 CF 9E  BC 3B 6E 63  15 9A D0 97
CTR[0002]: EA 20 32 DA  27 82 6E 13  9E 1E 72 5C  5B 0D 3E BF
CTR[MAC ]: B9 31 27 CA  F0 F1 A1 20  FA 70
Total packet length = 42. [Authenticated and Encrypted Output]
           47 A6 5A C7  8B 3D 59 42  27 E8 5E 71  E8 82 F1 DB
           D3 8C E3 ED  A7 C2 3F 04  DD 65 07 1E  B4 13 42 AC
           DF 7E 00 DC  CE C7 AE 52  98 7D

=============== Packet Vector #24 ==================
AES Key = D7 82 8D 13  B2 B0 BD C3  25 A7 62 36  DF 93 CC 6B
Nonce =    00 8D 49 3B  30 AE 8B 3C  96 96 76 6C  FA
Total packet length = 33. [Input with 12 cleartext header octets]
           6E 37 A6 EF  54 6D 95 5D  34 AB 60 59  AB F2 1C 0B
           02 FE B8 8F  85 6D F4 A3  73 81 BC E3  CC 12 85 17
           D4
CBC IV in: 61 00 8D 49  3B 30 AE 8B  3C 96 96 76  6C FA 00 15
CBC IV out:67 AC E4 E8  06 77 7A D3  27 1D 0B 93  4C 67 98 15
After xor: 67 A0 8A DF  A0 98 2E BE  B2 40 3F 38  2C 3E 98 15  [hdr]
After AES: 35 58 F8 7E  CA C2 B4 39  B6 7E 75 BB  F1 5E 69 08
After xor: 9E AA E4 75  C8 3C 0C B6  33 13 81 18  82 DF D5 EB  [msg]
After AES: 54 E4 7B 62  22 F0 BB 87  17 D0 71 6A  EB AF 19 9E
After xor: 98 F6 FE 75  F6 F0 BB 87  17 D0 71 6A  EB AF 19 9E  [msg]
After AES: 23 E3 30 50  BC 57 DC 2C  3D 3E 7C 94  77 D1 49 71
CBC-MAC  : 23 E3 30 50  BC 57 DC 2C  3D 3E
CTR Start: 01 00 8D 49  3B 30 AE 8B  3C 96 96 76  6C FA 00 01
CTR[0001]: 58 DB 19 B3  88 9A A3 8B  3C A4 0B 16  FF 42 2C 73
CTR[0002]: C3 2F 24 3D  65 DC 7E 9F  4B 02 16 AB  7F B9 6B 4D
CTR[MAC ]: 4E 2D AE D2  53 F6 B1 8A  1D 67
Total packet length = 43. [Authenticated and Encrypted Output]
           6E 37 A6 EF  54 6D 95 5D  34 AB 60 59  F3 29 05 B8
           8A 64 1B 04  B9 C9 FF B5  8C C3 90 90  0F 3D A1 2A
           B1 6D CE 9E  82 EF A1 6D  A6 20 59
```

9.  Intellectual Property Statements

   The authors hereby explicitly release any intellectual property
   rights to CCM to the public domain.  Further, the authors are not
   aware of any patent or patent application anywhere in the world that
   covers CCM mode.  It is our belief that CCM is a simple combination
   of well-established techniques, and we believe that CCM is obvious to
   a person of ordinary skill in the arts.

10.  Security Considerations

   We claim that this block cipher mode is secure against attackers
   limited to $2^{128}$ steps of operation if the key K is 256 bits or
   larger.  There are fairly generic precomputation attacks against all
   block cipher modes that allow a meet-in-the-middle attack on the key
   K.  If these attacks can be made, then the theoretical strength of
   this, and any other, block cipher mode is limited to $2^{(n/2)}$ where n
   is the number of bits in the key.  The strength of the authentication
   is of course limited by M.

   Users of smaller key sizes (such as 128-bits) should take precautions
   to make the precomputation attacks more difficult.  Repeated use of
   the same nonce value (with different keys of course) ought to be
   avoided.  One solution is to include a random value within the nonce.
   Of course, a packet counter is also needed within the nonce.  Since
   the nonce is of limited size, a random value in the nonce provides a
   limited amount of additional security.

11.  References

   This section provides normative and informative references.

11.1.  Normative References

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

11.2.  Informative References

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

   [CCM]       Whiting, D., Housley, R. and N. Ferguson, "AES Encryption
               & Authentication Using CTR Mode & CBC-MAC," IEEE P802.11
               doc 02/001r2, May 2002.

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

   [MAC]          NIST, FIPS PUB 113, "Computer Data Authentication," May
                  1985.

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

   [OCB]          Rogaway, P., Bellare, M., Black, J. and T, Krovetz, "OCB:
                  A block-Cipher Mod of Operation for Efficient
                  Authenticated Encryption," 8th ACM Conference on Computer
                  and Communications Security, pp 196-295, ACM Press, 2001.

   [PROOF]        Jonsson, J., "On the Security of CTR + CBC-MAC," SAC 2002
                  -- Ninth Annual Workshop on Selected Areas of
                  Cryptography, Workshop Record version, 2002.  Final
                  version to appear in the LNCS Proceedings.

## 12.  Acknowledgement

13.  Authors' Addresses

Doug Whiting
Hifn
5973 Avenida Encinas, #110
Carlsbad, CA 92009
USA

EMail: dwhiting@hifn.com


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

EMail: housley@vigilsec.com


Niels Ferguson
MacFergus BV
Bart de Ligtstraat 64
1097 JE Amsterdam
Netherlands

EMail: niels@macfergus.com

14.  Full Copyright Statement

Acknowledgement