The CCMP protocol is based on Advanced Encryption Standard (AES) encryption algorithm using the Counter Mode with CBC–MAC (CCM) mode of operation. The CCM mode combines Counter (CTR) mode privacy and Cipher Block Chaining Message Authentication Code (CBC-MAC) authentication. These modes have been used and studied for a long time, have well-understood cryptographic properties. They provide good security and performance in either hardware or software.

CCMP employs the AES encryption algorithm using the CCM mode of operation. The CCM mode combines Counter Mode (CTR) for confidentiality and Cipher Block Chaining Message Authentication Code (CBC-MAC) for authentication and integrity. The Advanced Encryption Algorithm (AES) algorithm is defined in FIPS PUB 197. All AES processing used within CCMP uses AES with a 128 bit key and a 128 bit block size. CCMP must use the AES algorithm with a 128 bit key and 128 bit block size. CCM provides other parameters (K, M and L) that must have the values: K=16, M=8 and L=2. CCM requires a fresh temporal key (TK) for every session. CCM also requires a unique nonce value for each frame protected by a given TK, and CCMP uses a 48-bit packet number (PN) for this purpose. Reuse of a packet number (PN) with the same TK voids all security guarantees.

CCMP Encapsulation:

- CCMP encapsulates a plaintext MAC Protocol Data Unit (MPDU) using the following steps:
  1. It first increments the Packet Number (PN), to obtain a fresh PN for each MPDU.
  2. The fields in the MAC header are used to construct the Additional Authentication Data (AAD).
  3. Construct CCM Nonce block (initialization vector) from the PN, A2 and the Priority of the MPDU.
  4. Encode the new PN and the KeyId into the 8 octet CCMP Header.
  5. Run CTR mode AES using the temporal key (TK), AAD, Nonce and MPDU data to form the ciphertext and Message Integrity Check (MIC).
  6. The Encrypted MPDU is formed by concatenating the original MAC Header, the CCMP header, the Encrypted Data and the MIC.

- Figure 1 depicts the CCMP encapsulation process

CCMP MIC Computation:

- CCMP uses AES in the CBC-MAC mode to compute a MIC for the MPDU. The input to this algorithm is:
  1. The plaintext MPDU.
  2. The Initial Block for this MPDU.
  3. The temporal key.
- The output of the algorithm is a MIC value. This can be appended to the MPDU on transmit, and compared with a received MIC at the receiver.
- The algorithm first encrypts the Initial Block to produce the CBC mode Initialization vector (IV). Next it computes the CBC-MAC over the IEEE 802.11 header length (Hlen), selected parts of the IEEE 802.11 MPDU header, and the plaintext MPDU data.
- Figure 2 depicts the MIC calculation process.

**Figure 1. CCMP Encapsulation Process**

**Figure 2. MIC calculation**

---

**Notes**
1. Discard most significant 8 octets
2. Pad n zeroes to most significant end of field such that: (field length + n) = 16

---

**Key:**
- AES: AES block cipher, using 128-bit key K
- MAC IV: 16 octet (at least) data field
- PN: Priority
- AAD: Additional Authentication Data
- MIC: MIC (8 octets)
- CCM: CCM encryption
- A3, A2: Priority
- Data: Plaintext MPDU
- Key: 128-bit key K

**Plaintext MPDU:**
- Construct AAD
- Construct Nonce
- Construct CCMP header
- Increment PN
- MAC header
- Encrypted Data, MIC
- Encrypted MPDU

**Encrypted MPDU:**
- MAC header
- Construct AAD
- Construct Nonce
- Construct CCMP header
- Increment PN

**Encrypted MPDU:**
- MAC header
- Construct AAD
- Construct Nonce
- Construct CCMP header
- Increment PN

CCMP CTR-mode encryption:

- CCMP uses AES in Counter Mode to encrypt and decrypt the MPDU data and MIC. The input to this algorithm is:
  1. The MPDU data field, with MIC appended. On transmission, the data field with MIC is plaintext, while on reception both are ciphertext.
  2. The Counter for this MPDU
  3. The temproal key.

- The CTR Preload contains one flag byte, one byte of QoS information, a six bytes address field, a six byte packet number and a two byte counter.
- The output of the algorithm is an encrypted MPDU data field on transmit and a decrypted MPDU data field with MIC on reception.

Figure 3 depicts the encryption process.
CCMP De-capsulation Process:

- CCMP (AES-CTR and CBC-MAC) requires only AES encryption operations and not AES decryption operations. The decapsulation process succeeds when the calculated MIC matches the MIC value received in the Encrypted MPDU.
- Figure 4 shows the CCMP decapsulation process.

![Figure 4. CCMP De-capsulation](image)

**CCMP Performance:**

- The following table 1 summarizes the number of MIPS required to encode 1 megabit of user data using CCMP with 64-bit Co-Processor:

<table>
<thead>
<tr>
<th>Implementation</th>
<th>128-bit key</th>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCMP with 64-bit Co-Processor</td>
<td>355.46875</td>
<td>19843</td>
</tr>
</tbody>
</table>

  **Table 1**

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