IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adding Support for Suite B | | | | |
| Date: 2012-09-10 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | Aruba Networks | 1322 Crossman ave, Sunnyvale, CA | +1 408 227 4500 | dharkins at arubanetworks dot com |
| Joe Salowey | Cisco Systems | 2901 3rd ave, Seattle, WA | +1 408 256 3380 | jsalowey at cisco dot com |
| Jouni Malinen | Qualcomm | 5775 Morehouse Dr, San Diego, CA | +1 858 651 6645 | jouni at qca dot qualcomm dot com |

Abstract

This document defines the text modifications necessary to add support for Suite B, which includes GCM-256, to the draft and proposes resolution of CID 6513 from LB188, as well as CID 4261 from LB187 which invited text and a discussion on the topic of GCM-256 and Suite B (here is that text).

***Instruct the editor to modify Tables 8-99 and 8-100 as indicated:***

* Cipher suites

|  |  |  |
| --- | --- | --- |
| * Table 8-99 Cipher suite selectors | | |
| OUI | Suite type | Meaning |
| 00-0F-AC | 8 | GCMP-128 – default for a DMG STA |
| 00-0F-AC | <ANA-1> | GCMP-256 |
| 00-0F-AC | <ANA-2> | CCMP-256 |
| 00-0F-AC | <ANA-3> | BIP-GMAC-128 |
| 00-0F-AC | <ANA-4> | BIP-GMAC-256 |

|  |  |  |  |
| --- | --- | --- | --- |
| * Table 8-100 Cipher suite usage | | | |
| Cipher suite selector | GTK | PTK | IGTK |
| GCMP-128 | Yes | Yes | No |
| GCMP-256 | Yes | Yes | No |
| CCMP-256 | Yes | Yes | No |
| BIP-GMAC-128 | No | No | Yes |
| BIP-GMAC-256 | No | No | Yes |

***Instruct the editor to modify section 8.4.2.27.3 as indicated:***

* AKM suites

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * Table 8-101 AKM suite selectors | | | | |
| OUI | Suite type | Meaning | | |
| Authentication type | Key management type | Key derivation type |
| 00-0F-AC | <ANA-5> | Authentication negotiated over IEEE 802.1X or using PMKSA caching as defined in 11.5.9.3 (Cached PMKSAs and RSNA key management) using a Suite B compliant EAP method supporting EC of GF(p=256) | RSNA key management as defined in 11.6 (Keys and key distribution) or using PMKSA caching as defined in 11.5.9.3 (Cached PMKSAs and RSNA key management) with HMAC-SHA256 | Defined in 11.6.1.7.2 using HMAC-SHA256 |
| 00-0F-AC | <ANA-6> | Authentication negotiated over IEEE 802.1X or using PMKSA caching as defined in 11.5.9.3 (Cached PMKSAs and RSNA key management) using a Suite B compliant EAP method supporting EC of GP(p=384) | RSNA key management as defined in 11.6 (Keys and key distribution) or using PMKSA caching as defined in 11.5.9.3 (Cached PMKSAs and RSNA key management) with HMAC-SHA384 | Defined in 11.6.1.7.2 using HMAC-SHA384 |
| 00-0F-AC | <ANA-7> | FT authentication negotiated over IEEE 802.1X | FT key management as defined in 11.6.1.7 with HMAC-SHA384 | Defined in 11.6.1.7.2 using HMAC-SHA384 |

.

The AKM suite selector value 00-0F-AC:<ANA-5> shall only be used with Cipher suite selector values 00-0F-AC:8 (GCMP-128) and 00-0F-AC:<ANA-3> (BIP-GMAC-128); and AKM suite selector value 00-0F-AC:<ANA-6> shall only be used with Cipher suite selector values 00-0F-AC:<ANA-1> (GCMP-256), 00-0F-AC:<ANA-2>, and 00-0F-AC:<ANA-4> (BIP-GMAC-256).

***Instruct the editor to modify section 8.4.2.57 as indicated:***

**8.4.2.57 Management MIC element**

The Management MIC element (MME) provides message integrity and protects group addressed robust management frames from forgery and replay. Figure 8-247 shows the MME format.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element ID | Length | KeyID | IPN | MIC |
| 1 | 1 | 2 | 6 | 8 or 16 |

Figure 8-247—Management MIC element format

The value of the Element ID field is 76 decimal (4c hex).

The Length field is set to 16 or 24.

The Key ID field identifies the IGTK used to compute the MIC. Bits 0–11 define a value in the range 0– 4095. Bits 12–15 are reserved. The IGTK Key ID is either 4 or 5. The remaining Key IDs are reserved.

The IPN field contains a 6 octet value, interpreted as a 48-bit unsigned integer and used to detect replay of protected group addressed robust management frames.

The MIC field contains a message integrity code calculated over the robust management frame as specified in 11.4.4.5 and 11.4.4.6. The length of the MIC field depends on the specific cipher negotiated, either BIP (8 octets) or BIP-GMAC-128 (16 octets), or BIP-GMAC-256 (16 octets).

***Instruct the editor to modify the following sub-sections of 11.4.3 as indicated:***

* CTR with CBC-MAC Protocol (CCMP)
* General

Subclause CTR with CBC-MAC Protocol (CCMP) specifies the CCMP, which provides data confidentiality, authentication, integrity, and replay protection. CCMP is mandatory for RSN compliance.

CCMP is based on the CCM of the AES encryption algorithm. CCM combines CTR for data confidentiality and CBC-MAC for authentication and integrity. CCM protects the integrity of both the MPDU Data field and selected portions of the IEEE 802.11 MPDU header.

The AES algorithm is defined in FIPS PUB 197-2001. AES processing used within CCMP uses AES with either a 128-bit key or a 256-bit key.

CCM is defined in IETF RFC 3610. CCM is a generic mode that can be used with any block-oriented encryption algorithm. CCM has two parameters (*M* and *L*). CCMP with a 128-bit key uses the following values for the CCM parameters:

* *M* = 8; indicating that the MIC is 8 octets.
* *L* = 2; indicating that the Length field is 2 octets, which is sufficient to hold the length of the largest possible IEEE 802.11 MPDU, expressed in octets.

And CCMP with a 256-bit key uses the following values for the CCM parameters:

* *M* = 16; indicating that the MIC is 16 octets.
* *L* = 2; indicating that the Length field is 2 octets, which is sufficient to hold the length of the largest possible IEEE 802.11 MPDU, expressed in octets.

CCM requires a fresh temporal key for every session. CCM also requires a unique nonce value for each frame protected by a given temporal key, and CCMP uses a 48-bit packet number (PN) for this purpose. Reuse of a PN with the same temporal key voids all security guarantees.

* CCMP MPDU format

***Change the length of the MIC field in figure 11-16 from “8 octets” to “variable”***

When used with a 128-bit key, CCMP processing expands the original MPDU size by 16 octets, 8 octets for the CCMP Header field and 8 octets for the MIC field. When used with a 256-bit key, CCMP processing expands the original MPDU size by 24 octets, 8 octets for the CCMP Header field and 16 octets for the MIC field. The CCMP Header field is constructed from the PN, ExtIV, and Key ID subfields. PN is a 48-bit PN represented as an array of 6 octets. PN5 is the most significant octet of the PN, and PN0 is the least significant. Note that CCMP does not use the WEP ICV.

* CCM originator processing

CCM is a generic authenticate-and-encrypt block cipher mode, and in this standard, CCM is used with the AES block cipher.

There are four inputs to CCM originator processing:

* *Key:* the temporal key (16 octets).
* *Nonce:* the nonce (13 octets) constructed as described in 11.4.3.3.4.
* *Frame body:* the frame body of the MPDU.
* *AAD:* the AAD (22–30 octets) constructed from the MPDU header as described in 11.4.3.3.3.

The CCM originator processing provides authentication and integrity of the frame body and the AAD as well as data confidentiality of the frame body. The output from the CCM originator processing consists of the encrypted data and encrypted MIC (see Figure 11-16).

* CCM recipient processing

CCM recipient processing uses the same parameters as CCM originator processing. A CCMP protected individually addressed robust management frame shall use the same TK as a Data MPDU.

There are four inputs to CCM recipient processing:

* *Key:* the temporal key (16 octets).
* *Nonce:* the nonce (13 octets) constructed as described in 11.4.3.3.4.
* *Encrypted frame body:* the encrypted frame body from the received MPDU. The encrypted frame body includes the MIC.
* *AAD:* the AAD (22–30 octets) that is the canonical MPDU header as described in 11.4.3.3.3.

***Instruct the editor to modify section 11.4.4 (and its subsections) as indicated:***

* Broadcast/Multicast Integrity Protocol (BIP)
* BIP overview

BIP provides data integrity and replay protection for group addressed robust management frames after successful establishment of an IGTKSA (see **11.5.1.1.9 (IGTKSA).**).

BIP provides data integrity and replay protection, using AES-128 in CMAC Mode, AES-128 in GMAC mode, and AES-256 in GMAC mode. NIST SP 800-38B defines the CMAC algorithm and NIST SP 800-38D defines the GMAC algorithm.BIP processing uses AES with a 128-bit integrity key and a CMAC TLen value of 128 (16 octets). The CMAC output is truncated to 64 bits:

MIC = L(CMAC Output, 0, 64)

Where L is defined in 11.6.1 (Key Hierarchy)**.**

BIP-GCMP-128 uses AES with a 128-bit integrity key and BIP-GCMP-256 uses AES with a 256-bit integrity key. The authentication tag for both BIP-GCMP-128 and BIP-GCMP-256 is not truncated and shall be 128 bits (16 octets).

BIP uses the IGTK to compute the MMPDU MIC. The authenticator shall distribute one new IGTK and IGTK PN (IPN) whenever it distributes a new GTK. The IGTK is identified by the MAC address of the transmitting STA plus an IGTK identifier that is encoded in the MME Key ID field.

* BIP transmission

When a STA transmits a protected group addressed robust management frame, it shall

* Select the IGTK currently active for transmission of frames to the intended group of recipients and construct the MME (see 8.4.2.57 (Management MIC element)) with the MIC field masked to 0 and the KeyID field set to the corresponding IGTK KeyID value. The transmitter shall insert a monotonically increasing non-negative integer into the MME IPN field. For BIP-GMAC-128 and BIP-GMAC-256, the initialization vector passed to GMAC shall be a concatentation of address 2 from the MAC header of the MPDU and the non-negative integer inserted into the MMP IPN field.
* Compute AAD as specified in 11.4.4.3 (BIP AAD construction).
* Compute an integrity value over the concatenation of (AAD || Management Frame Body including MME), and insert the output into the MME MIC field. For BIP, the integrity value is 64-bits and is computed using AES-128-CMAC; for BIP-GMAC-128, the integrity value is 128-bits and is computed using AES-128-GMAC; and, for BIP-GMAC-256, the integrity value is 128-bits and is computed using AES-256-GMAC.
* Compose the frame as the IEEE 802.11 header, management frame body, including MME, and FCS. The MME shall appear last in the frame body.
* Transmit the frame.
* BIP reception

When a STA with management frame protection negotiated receives a group addressed robust management frame protected by BIP, BIP-GMAC-128 or BIP-GMAC-256, it shall

* Identify the appropriate IGTK key and associated state based on the MME KeyID field. If no such IGTK exists, silently drop the frame.

b) Perform replay protection on the received frame. The receiver shall interpret the MME IPN field as a 48-bit unsigned integer. It shall compare this MME IPN integer value to the value of the receive replay counter for the IGTK identified by the MME Key ID field. If the integer value from the received MME IPN field is less than or equal to the replay counter value for this IGTK, the receiver shall discard the frame and increment the dot11RSNAStatsCMACReplays counter by 1.

* If the replay protection succeeds, compute AAD for this management frame, as specified in 11.4.4.3 (BIP AAD construction). For BIP-GMAC-128 and BIP-GMAC-256, an initialization vector for GMAC is constructed as the concatenation of address 2 from the MAC header of the MPDU and the 48-bit unsigned integer from the MME IPN field.
* Extract and save the received MIC value, and compute a verifier over the concatenation of (AAD || Management Frame Body || MME) with the MIC field masked to 0 in the MME. For BIP, the verifier is AES-128-CMAC; for BIP-GMAC-128, the verifier is AES-128-GMAC; and, for BIP-GMAC-256, the verifier is AES-256-GMAC. If the result does not match the received MIC value, then the receiver shall discard the frame and increment the dot11RSNAStatsCMACICVErrors counter by 1.
* Update the replay counter for the IGTK identified by the MME Key ID field with the integer value of the MME IPN field.

If management frame protection is negotiated, group addressed robust management frames that are received without BIP protection shall be discarded.

***Instruct the editor to modify section 11.4.5.1 as indicated:***

11.4.5.1 GCMP overview

Subclause 11.4.5. specifies the GCMP, which provides data confidentiality, authentication, integrity, and replay protection.

GCMP is based on the GCM of the AES encryption algorithm. GCM combines Galois/Counter Mode for data confidentiality and GMAC for authentication and integrity. GCM protects the integrity of both the MPDU Data field and selected portions of the MPDU header.

The AES algorithm is defined in FIPS PUB 197-2001. AES processing used within GCMP uses AES with a 128-bit key (GCMP-128) or a 256-bit key (GCMP-256).

***Instruct the editor to modify section 11.6.1.2 as indicated:***

* PRF

A PRF is used in a number of places in this standard. Depending on its use, it may need to output 128 bits, 192 bits, 256 bits, 384 bits, 512 bits, or 704 bits. This subclause defines sixfunctions:

* PRF-128, which outputs 128 bits
* PRF-192, which outputs 192 bits
* PRF-256, which outputs 256 bits
* PRF-384, which outputs 384 bits
* PRF-512, which outputs 512 bits
* PRF-704, which outputs 704 bits

In the following, *K* is a key; *A* is a unique label for each different purpose of the PRF; *B* is a variable-length string; *Y* is a single octet containing 0; *X* is a single octet containing the loop parameter *i*; and || denotes concatenation:

H-SHA-1(*K*, *A*, *B*, *X*)  HMAC-SHA-1(*K*, *A* || *Y* || *B* || *X*)

PRF(*K*, *A*, *B*, *Len*)

**for** *i*  0 **to** (*Len*+159)/160 **do**

*R*  *R* || H-SHA-1(*K*, *A*, *B*, *i)*

**return** L(*R*, 0, Len)

PRF-128(*K*, *A*, *B*) = PRF(*K*, *A*, *B*, 128)

PRF-192(*K*, *A*, *B*) = PRF(*K*, *A*, *B*, 192)

PRF-256(*K*, *A*, *B*) = PRF(*K*, *A*, *B*, 256)

PRF-384(*K*, *A*, *B*) = PRF(*K*, *A*, *B*, 384)

PRF-512(*K*, *A*, *B*) = PRF(*K*, *A*, *B*, 512)

When the negotiated AKM is 00-0F-AC:5, 00-0F-AC:6, 00-0F-AC:<ANA-5>, the KDF specified in 11.6.1.7.2 (Key derivation function (KDF)) shall be used instead of the PRF construction defined here. In this case, A is used as the KDF label and B as the KDF Context and the PRF functions are defined as follows:

PRF-128(K, A, B) = KDF-SHA256-128(K, A, B)

PRF-192(K, A, B) = KDF-SHA256-192(K, A, B)

PRF-256(K, A, B) = KDF-SHA256-256(K, A, B)

PRF-384(K, A, B) = KDF-SHA256-384(K, A, B)

PRF-512(K, A, B) = KDF-SHA256-512(K, A, B)

When the negotiated AKM is 00-0F-AC:<ANA-6> the KDF specified in 11.6.1.7.2 (Key derivation function (KDF)) shall be used instead of the PRF construction defined here. In this case, A is used as the KDF label and B as the KDF Context and the PRF function is defined as follows:

PRF-704(K, A, B) = KDF-SHA384-704(K, A, B)

When the negotiated AKM is 00-0F-AC:<ANA-7> the KDF specified in 11.6.1.72 (Key derivation function (KDF)) shall be used instead of the PRF construction defined here. In this case, A is used as the KDF label and B as the KDF Context and the PRF functions are defined as follows:

PRF-384(K, A, B) = KDF-SHA384-384(K, A, B)

PRF-512(K, A, B) = KDF-SHA384-512(K, A, B)

PRF-704(K, A, B) = KDF-SHA384-704(K, A, B)

***Instruct the editor to modify section 11.6.1.3 as indicated:***

* Pairwise key hierarchy

Except when preauthentication is used, the pairwise key hierarchy utilizes PRF-384, PRF-512 or PRF-704 to derive session-specific keys from a PMK, as depicted in Figure 11-24 (Pairwise key hierarchy). When using AKM suite selector 00-0F-AC:<ANA-6>, the length of the PMK, PMK\_bits, shall be 384 bits. With all other AKM suite selectors, the length of the PMK, PMK\_bits, shall be 256 bits. The pairwise key hierarchy takes a PMK and generates a PTK. The PTK is partitioned into KCK, KEK, and temporal keys, which are used by the MAC to protect individually addressed communication between the Authenticator’s and Supplicant’s respective STAs. PTKs are used between a single Supplicant and a single Authenticator.

***Instruct the editor to modify figure 11-4 to replace “L(PTK, 0, 128) (KCK)” with “L(PTK, 0, KCK\_bits) (KCK)”, “L(PTK, 128, 128) (KEK)” with “L(PTK, KCK\_bits, KEK\_bits) (KEK)” and “L(PTK,256,TK\_bits) (TK)” with “L(PTK, KCK\_bits+KEK\_bits, TK\_bits) (TK)”***

When not using a PSK, the PMK is derived from the MSK. The PMK shall be computed as the first PMK\_bitsbits (bits 0–PMK\_bits) of the MSK: PMK  L(MSK, 0, PMK\_bits). The PTK shall not be used longer than the PMK lifetime as determined by the minimum of the PMK lifetime indicated by the AS, e.g., Session-Timeout + dot1xAuthTxPeriod or from dot11RSNAConfigPMK--Lifetime. When RADIUS is used and the Session-Timeout attribute is not in the RADIUS Accept message, and if the key lifetime is not otherwise specified, then the PMK lifetime is -infinite.

NOTE 1—If the protocol between the Authenticator (or AP) and AS is RADIUS, then the MS-MPPE-Recv-Key attribute (-vendor-id = 17; see Section 2.4.3 in IETF RFC 2548-1999 [B30]) is available to be used to transport the first 32 octets of the MSK to the AP, and the MS-MPPE-Send-Key attribute (vendor-id = 16; see Section 2.4.2 in IETF RFC 2548-1999 [B30]) is available to be used to transport the remaining 32 octets of the MSK.

NOTE 2—When reauthenticating and changing the pairwise key, a race condition might occur. If a frame is received while MLME-SETKEYS.request primitive is being processed, the received frame might be decrypted with one key and the MIC checked with a different key. Two possible options to avoid this race condition are as follows: the frame might be checked against the old MIC key, and the received frames might be queued while the keys are changed.

NOTE 3—If the AKMP is RSNA-PSK, then a 256-bit PSK might be configured into the STA and AP or a pass-phrase might be configured into the Supplicant or Authenticator. The method used to configure the PSK is outside this standard, but one method is via user interaction. If a pass-phrase is configured, then a 256-bit key is derived and used as the PSK. In any RSNA-PSK method, the PSK is used directly as the PMK. Implementations might support different PSKs for each pair of communicating STAs.

Here, the following assumptions apply:

* SNonce is a random or pseudorandom value contributed by the Supplicant; its value is taken when a PTK is instantiated and is sent to the PTK Authenticator.
* ANonce is a random or pseudorandom value contributed by the Authenticator.
* The PTK shall be derived from the PMK by

PTK  PRF-X(PMK, “Pairwise key expansion”, Min(AA,SPA) || Max(AA,SPA) || Min(ANonce,SNonce) || Max(ANonce,SNonce))

where X = KCK\_bits + KEK\_bits+ TK\_bits. The values of KCK\_bits and KEK bits are AKM suite dependent and are listed in Table 11-9 (Integrity and key wrap algorithms). The value of TK\_bits is cipher-suite dependent and is defined in Table 11-4 (Cipher suite key lengths). The Min and Max operations for IEEE 802 addresses are with the address converted to a positive integer treating the first transmitted octet as the most significant octet of the integer. The Min and Max operations for nonces are with the nonces treated as positive integers converted as specified in 8.2.2 (Conventions).

NOTE—The Authenticator and Supplicant normally derive a PTK only once per association. A Supplicant or an Authenticator use the 4-Way Handshake to derive a new PTK. Both the Authenticator and Supplicant create a new nonce value for each 4-Way Handshake instance.

* The KCK shall be computed as the first KCK\_bits bits (bits 0–KCK\_bits) of the PTK:

KCK  L(PTK, 0, KCK\_bits)

The KCK is used by IEEE Std 802.1X-2004 to provided data origin authenticity in the 4-Way Handshake and Group Key Handshake messages.

* The KEK shall be computed as the next KEK\_bits bits of the PTK:

KEK  L(PTK, KCK\_bits, KEK\_bits)

The KEK is used by the EAPOL-Key frames to provide data confidentiality in the 4-Way Handshake and Group Key Handshake messages.

* The temporal key (TK) shall be computed as the next TK\_bits of the PTK:

TK  L(PTK, KCK\_bits + KEK\_bits, TK\_bits)

The EAPOL-Key state machines (see 11.6.10 (RSNA Supplicant key management state machine) and 11.6.11 (RSNA Authenticator key management state machine)) use the MLME-SETKEYS.request primitive to configure the temporal key into the STA. The STA uses the temporal key with the pairwise cipher suite; interpretation of this value is cipher-suite-specific.

A PMK identifier is defined as

PMKID = HMAC-SHA1-128(PMK, "PMK Name" || AA || SPA)

Here, HMAC-SHA1-128 is the first 128 bits of the HMAC-SHA1 of its argument list.

When the negotiated AKM is 00-0F-AC:5 or 00-0F-AC:6, HMAC-SHA-256 is used to calculate the PMKID, and the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-256(PMK, "PMK Name" || AA || SPA))

When the negotiated AKM is 00-0F-AC:<ANA-5>, HMAC-SHA-256 is used to calculate the PMKID, and the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-256(KCK, "PMK Name" || AA || SPA))

When the negotiated AKM is 00-0F-AC:<ANA-6>, HMAC-SHA-384 is used to calculate the PMKID, and the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-384(KCK, "PMK Name" || AA || SPA))

NOTE—When the PMKID is calculated for the PMKSA as part of RSN preauthentication, the AKM has not yet been negotiated. In this case, the HMAC-SHA1-128 based derivation is used for the PMKID calculation..

***Instruct the editor to modify section 11.6.1.7.2 as indicated:***

* Key derivation function (KDF)

The KDF for the FT key hierarchy, and for AKMs 00-0F-AC:<ANA-5> and 00-0F-AC:<ANA-6>, is a variant of the pseudorandom function (PRF) defined in 11.6.1.2 (PRF) and is defined as follows:

**Output**  **KDF-Hash-Length (K, label, Context) where**

Input: *K*, a key derivation key whose length equals the block size of the hash function

*Hash*, a cryptographically strong hash function

*label*, a string identifying the purpose of the keys derived using this KDF

*Context*, a bit string that provides context to identify the derived key

*Length*, the length of the derived key in bits

Output: a *Length*-bit derived key

*result*  ""

*iterations*  (*Length*+255)/256

**do** *i* = 1 **to** *iterations*

*result*  *result* || HMAC-Hash(*K*, *i* || *label* || *Context* || *Length*)

**od**

**return** first *Length* bits of *result,* and securely delete all unused bits

In this algorithm, *i* and *Length* are encoded as 16-bit unsigned integers, represented using the bit ordering conventions of 8.2.2 (Conventions). *K*, *label*, and *Context* are bit strings and are represented using the ordering conventions of 8.2.2 (Conventions).

***Instruct the editor to modify 11.6.1.7.3 as indicated:***

**11.6.1.7.3 PMK-R0**

The first-level key in the FT key hierarchy, PMK-R0, is derived using the KDF defined in 11.6.1.7.2. The

PMK-R0 is the first level 256-bit keying material used to derive the next level keys (PMK-R1s):

R0-Key-Data = KDF-Hash-Z(XXKey, "FT-R0", SSIDlength || SSID || MDID || R0KHlength ||

R0KH-ID || S0KH-ID)

PMK-R0 = L(R0-Key-Data, 0, L)

PMK-R0Name-Salt = L(R0-Key-Data, L, 128)

where

* KDF-Hash-Z is the KDF as defined in 11.6.1.7.2 used to generate a key of length 384 bits.
* L(-) is defined in 11.6.1.
* If the AKM negotiated is 00-0F-AC:3, then Hash shall be SHA256, Z shall be 384, L shall be 256, and XXKey shall be the second 256 bits of the MSK (which is derived from the IEEE 802.1X authentication), i.e., XXKey = L(MSK, 256, 256). If the AKM negotiated is 00-0F-AC:4, then Hash shall be SHA256, Z shall be 384, L shall be 256 and XXKey shall be the PSK. If the AKM negotiated is 00-0F-AC:9, then Hash shall be SHA256, Z shall be 384, L shall be 256 and XXKey shall be the MPMK generated as the result of SAE authentication. If the AKM negotiated is 00-0F-AC-<ANA-7> then Hash shall be SHA384, Z shall be 512, L shall be 384, and XXKey shall be the first 384 bits of the MSK (which is derived from the IEEE 802.1X authentication), i.e., XXKey = L(MSK, 0, 384).

***Instruct the editor to modify 11.6.1.7.4 as indicated:***

**11.6.1.7.4 PMK-R1**

The second-level key in the FT key hierarchy, PMK-R1, is a 256-bit or 384-bit key used to derive the PTK. The PMKR1is derived using the KDF defined in 11.6.1.7.2:

PMK-R1 = KDF-Hash-Z(PMK-R0, "FT-R1", R1KH-ID || S1KH-ID)

where

* KDF-Hash-Z is the KDF as defined in 11.6.1.7.2 .
* If the AKM negotiated is 00-0F-AC:3, 00-0F-AC:4, or 00-0F-AC:9, then Hash shall be SHA256 and Z shall be 256. If the AKM negotiated is 00-0F-AC:<ANA-7> then Hash shall be SHA384 and Z shall be 384.
* PMK-R0 is the first level key in the FT key hierarchy.
* "FT-R1" is 0x46 0x54 0x2D 0x52 0x31.
* R1KH-ID is a MAC address of the holder of the PMK-R1 in the Authenticator of the AP.
* S1KH-ID is the SPA.

***Instruct the editor to modify 11.6.1.7.5 as indicated:***

**11.6.1.7.5 PTK**

The third-level key in the FT key hierarchy is the PTK. This key is mutually derived by the S1KH and the R1KH used by the target AP, with the key length being a function of the negotiated cipher suite as defined by Table 11-4 in 11.6.2.

Using the KDF defined in 11.6.1.7.2, the PTK derivation is as follows:

PTK = KDF-Hash-PTKLen(PMK-R1, "FT-PTK", SNonce || ANonce || BSSID || STA-ADDR)

where

* KDF-Hash-PTKLen is the KDF as defined in 11.6.1.7.2 used to generate a PTK of length PTKLen.
* If the AKM negotiated is 00-0F-AC:3, 00-0F-AC:4, or 00-0F-AC:9, then Hash shall be SHA256. If the AKM negotiated is 00-0F-A:<ANA-7>, then Hash shall be SHA384.
* PMK-R1 is the key that is shared between the S1KH and the R1KH.
* "FT-PTK" is 0x46 0x54 0x2D 0x50 0x54 0x4B.
* SNonce is a 256-bit random bit string contributed by the S1KH.
* ANonce is a 256-bit random bit string contributed by the R1KH.
* STA-ADDR is the non-AP STA’s MAC address.
* BSSID is the BSSID of the target AP.
* PTKlen is the total number of bits to derive, i.e., number of bits of the PTK. The length is dependent on the negotiated cipher suites and AKM suites as defined by Table 11-4 in 11.6.2 and Table 11-9 in 11.6.3.

Each PTK has three component keys, KCK, KEK, and a temporal key, derived as follows:

The KCK shall be computed as the first KCK\_bits bits (bits 0–KCK\_bits-1) of the PTK:

KCK = L(PTK, 0, KCK\_bits)

where L(-) is defined in 11.6.1.

The KCK is used to provide data origin authenticity in EAPOL-Key messages, as defined in 11.6.2, and in the FT authentication sequence, as defined in 12.8.

The KEK shall be computed as the next KEK\_bits bits of the PTK:

KEK = L(PTK, KCK\_bits, KEK\_bits)

The KEK is used to provide data confidentiality for certain fields (KeyData) in EAPOL-Key messages, as defined in 11.6.2, and in the FT authentication sequence, as defined in 12.8.

The temporal key (TK) shall be computed as the next TK\_bits bits (see table 11-4) of the PTK:

TK = L(PTK, KCK\_bits+KEK\_bits, TK\_bits)

For vendor-specific cipher suites, the length of the temporal key (and the value of PTKLen) depend on the vendor-specific algorithm.

***Instruct the editor to modify table 11-4 in section 11.6.2 as indicated:***

|  |  |  |
| --- | --- | --- |
| * Table 11-4 Cipher suite key lengths | | |
| Cipher suite | Key length  (octets) | TK\_bits  (bits) |
| WEP-40 | 5 | 40 |
| WEP-104 | 13 | 104 |
| TKIP | 32 | 256 |
| CCMP | 16 | 128 |
| BIP | 16 | 128 |
| GCMP-128 | 16 | 128 |
| GCMP-256 | 32 | 256 |
| CCMP-256 | 32 | 256 |
| BIP-GMAC-128 | 16 | 128 |
| BIP-GMAC-256 | 32 | 256 |

***Instruct the editor to modify table 11-9 in section 11.6.3 as indicated:***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | * Table 11-9 Integrity and key-wrap algorithms | | | | |  |
| AKM | | Integrity algorithm | KCK\_bits | Size of MIC | Key-wrap algorithm | KEK\_bits |
| Deprecated | | HMAC-MD5 | 128 | 16 | ARC4 | 128 |
| 00-0F-AC:1 | | HMAC-SHA1-128 | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:2 | | HMAC-SHA1-128 | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:3 | | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:4 | | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:5 | | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:6 | | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:<ANA-5> | | HMAC-SHA256 | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:<ANA-6> | | HMAC-SHA384 | 192 | 24 | NIST AES Key Wrap | 256 |
| 00-0F-AC:<ANA-7> | | HMAC-SHA384 | 192 | 24 | NIST AES Key Wrap | 256 |

***Instruct the editor to modify section 11.7.7 as indicated:***

* Mapping IGTK to BIP keys

See 11.6.1.5 (Integrity group key hierarchy) for the definition of the IGTK key. A STA shall use bits 0–127 of the IGTK as the AES-128-CMAC key, bits 0-127 of the IGTK as the AES-128-GMAC key, and bits 0-255 of the IGTK as the AES-256-GMAC key.

***Instruct the editor to append the following to M.6.4***

=== CCMP 256 Test Vector ====

TK: c9 7c 1f 67 ce 37 11 85 51 4a 8a 19 f2 bd d5 2f 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

PN: b5 03 97 76 e7 0c

802.11 Header: 08 48 c3 2c 0f d2 e1 28 a5 7c 50 30 f1 84 44 08 ab ae a5 b8 fc ba 80 33

CCMP-256 AAD: 08 40 0f d2 e1 28 a5 7c 50 30 f1 84 44 08 ab ae a5 b8 fc ba 00 00

CCMP-256 Nonce: 00 50 30 f1 84 44 08 b5 03 97 76 e7 0c

Plaintext Data: f8 ba 1a 55 d0 2f 85 ae 96 7b b6 2f b6 cd a8 eb 7e 78 a0 50

CCMP-256 encrypted: f3 d0 a2 fe 9a 3d bf 23 42 a6 43 e4 32 46 e8 0c 3c 04 d0 19

Encrypted MPDU (without FCS) : 08 48 c3 2c 0f d2 e1 28 a5 7c 50 30 f1 84 44 08 ab ae a5 b8 fc ba 80 33 0c e7 00 20 76 97 03 b5 f3 d0 a2 fe 9a 3d bf 23 42 a6 43 e4 32 46 e8 0c 3c 04 d0 19 d5 eb ec a5 c0 6d 10 5d 68 9f b3 db 81 e6 cc ff

FCS: 4e 69 88 a3

***Instruct the editor to append the following to M.9.1***

==== BIP-GMAC-128 with broadcast Deauthentication frame ====

Unprotected broadcast Deauthentication frame (without FCS):

c0 00 00 00 ff ff ff ff ff ff 02 00 00 00 00 00 02 00 00 00 00 00 09 00 02 00

FC=c0 00

DUR=00 00

DA=ff ff ff ff ff ff

SA=02 00 00 00 00 00

BSSID=02 00 00 00 00 00

SEQ=09 00

Reason Code: 00 20

IGTK: 4e a9 54 3e 09 cf 2b 1e ca 66 ff c5 8b de cb cf

IPN: 04 00 00 00 00 00

BIP-GMAC AAD (FC | A1 | A2 | A3): c0 00 ff ff ff ff ff ff 02 00 00 00 00 00 02 00 00 00 00 00

Management Frame Body: 02 00

BIP-GMAC IV: 02 00 00 00 00 00 00 00 00 00 00 04

BIP-GMAC MMIE MIC: 3e d8 62 fb 0f 33 38 dd 33 86 c8 97 e2 ed 05 3d

Protected MPDU (without FCS):

c0 00 00 00 ff ff ff ff ff ff 02 00 00 00 00 00 02 00 00 00 00 00 09 00 02 00 4c 18 04 00 04 00 00 00 00 00 3e d8 62 fb 0f 33 38 dd 33 86 c8 97 e2 ed 05 3d

==== BIP-GMAC-256 with broadcast Deauthentication frame ====

Unprotected broadcast Deauthentication frame (without FCS):

c0 00 00 00 ff ff ff ff ff ff 02 00 00 00 00 00 02 00 00 00 00 00 09 00 02 00

FC=c0 00

DUR=00 00

DA=ff ff ff ff ff ff

SA=02 00 00 00 00 00

BSSID=02 00 00 00 00 00

SEQ=09 00

Reason Code: 00 20

IGTK: 4e a9 54 3e 09 cf 2b 1e ca 66 ff c5 8b de cb cf 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

IPN: 04 00 00 00 00 00

BIP-GMAC AAD (FC | A1 | A2 | A3): c0 00 ff ff ff ff ff ff 02 00 00 00 00 00 02 00 00 00 00 00

Management Frame Body: 02 00

BIP-GMAC IV: 02 00 00 00 00 00 00 00 00 00 00 04

BIP-GMAC MMIE MIC: 23 be 59 dc c7 02 2e e3 83 62 7e bb 10 17 dd fc

Protected MPDU (without FCS):

c0 00 00 00 ff ff ff ff ff ff 02 00 00 00 00 00 02 00 00 00 00 00 09 00 02 00 4c 18 04 00 04 00 00 00 00 00 23 be 59 dc c7 02 2e e3 83 62 7e bb 10 17 dd fc

***Instruct the editor to append the following to M.11.1***

==== GCMP-256 test mpdu #3 =====

TK= c9 7c 1f 67 ce 37 11 85 51 4a 8a 19 f2 bd d5 2f 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

PN= 00 89 5f 5f 2b 08

802.11 Header= 88 48 0b 00 0f d2 e1 28 a5 7c 50 30 f1 84 44 08 50 30 f1 84 44 08 80 33 03 00

GCMP AAD= 88 40 0f d2 e1 28 a5 7c 50 30 f1 84 44 08 50 30 f1 84 44 08 00 00 03 00

GCMP Nonce= 50 30 f1 84 44 08 00 89 5f 5f 2b 08

Plaintext Data= 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13

14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27

GCMP MIC= 11 43 16 85 90 95 47 3d 5b 1b d5 96 b3 de a3 bf

Encrypted Data= 65 83 43 c8 b1 44 47 d9 21 1d ef d4 6a d8 9c 71 0c 6f c3 33

33 23 6e 39 97 b9 17 6a 5a 8b e7 79 b2 12 66 55 5e 70 ad 79

Encrypted MPDU (without FCS)=

88 48 0b 00 0f d2 e1 28 a5 7c 50 30 f1 84 44 08 50 30 f1 84 44 08 80 33 03 00 08 2b 00 20 5f 5f 89 00 65 83 43 c8 b1 44 47 d9 21 1d ef d4 6a d8 9c 71 0c 6f c3 33 33 23 6e 39 97 b9 17 6a 5a 8b e7 79 b2 12 66 55 5e 70 ad 79 11 43 16 85 90 95 47 3d 5b 1b d5 96 b3 de a3 bf

FCS= cf 16 ed 59

**References:**

* Special Publication 800-56A, National Institute of Standards and Technology
* Suite B Implementer’s Guide to NIST SP 800-56A, National Security Agency (USA)