IEEE P802.11
Wireless LANs

|  |
| --- |
| Resolution of CIDs for section 11.11.2.3 |
| Date: 2014-07-10 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | Aruba Networks | 1322 Crossman ave, Sunnyvale, CA | +1 408 227 4500 | dharkins at aruba networks dot com |
|  |  |  |  |  |

Abstract

This submission proposes resolution to comments related to the key derivation section, 11.11.2.3. Namely: CIDs 4082, 4083, 4292, 4329, 4330, 4331, 4893, 4950, 5075, and 5076.

***Instruct editor to modify section 2 as indicated:***

**2. Normative references**

IETF RFC 5869 – HMAC-based Extract-and-Expand Key Derivation Function (HKDF)

***Instruct editor to modify section 11.11.2.3 as indicated:***

**11.11.2.3 Key derivation with FILS authentication**

When not using PMKSA caching, a PMK is created using the Extract function of RFC 5869. When using PMKSA caching, a new PMKSA is not created. Instead, the PMKSA used for PMKSA caching remains and continues to be identified by the appropriate PMKID. Regardless of whether PMKSA caching is used or not, a PTKSA shall be generated with each FILS authentication exchange.

PTKSA creation uses the KDF from 11.6.1.7.2 (Key derivation function (KDF)) to derive the following keys from the PMK: a key confirmation key (KCK), a key encryption key (KEK), and a temporal key (TK).

When the AKM used is 00-0F-AC:<ANA-1> the hash algorithm used for the PMKSA and PTKSA creationshall be SHA256 and when the AKM used is 00-0F-AC:<ANA-2> the hash algorithm used for PMKSA and PTKSA creationshall be SHA384.

**11.11.2.3.1 PMK key derivation with FILS authentication**

The Extract function used to derive the PMK takes the two nonces as salt and the secret(s) from FILS Key establishment as input keying material. A PMKID used to identify the PMKSA is generated using the hash algorithm from the negotiated AKM on input data specific to the FILS Key Establishment step. The length of the PMK shall be either 256 bits or 384 bits depending on the negotiated AKM, and the length of the PMKID shall be 128 bits:

PMK = HMAC-Hash(SNonce || ANonce, IKM)

 PMKID = Truncate-128(Hash(Context))

Where:

* < SNonce is the STA nonce and ANonce is the AP nonce
* IKM is one of:
	+ The rMSK is FILS Shared Key authentication was performed without PFS
	+ A concatenation of the rMSK and the shared secret, ss, in that order if FILS Shared Key authentication was performed with PFS
	+ The shared secret, ss, if FILS Public Key authentication was performed
* Context is either:
	+ The EAP Init/Re-Auth packet when FILS Shared Key authentication is being performed without PFS; or,
	+ A concatenation of the STA’s Diffie-Hellman public value and the AP’s Diffie-Hellman public value, in that order, in all other cases

Upon completion of PMK and PMKID generation the shared secret, ss, and rMSK, if applicable, shall be irretrievably destroyed.

**11.11.2.3.2 PTK key derivation with FILS authentication**

For PTKSA key generation, the inputs to the KDF are thethe PMK of the PMKSA, a constant label, and a concatentation of the STA MAC address, the AP’s BSSID, the STA’s nonce, and the AP’s nonce. When the AKM used is 00-0F-AC:<ANA-1>, the length of KEK shall be 128 bits, and the length of the KCK 256 bits. When the AKM used is 00-0F-AC:<ANA-2> the length of the KEK shall be 256 bits, and the length of KCK shall be 384 bits, The total amount of bits extracted from the KDF shall therefore be 384+TK or 640+TK bits depending on the AKM used, where TK\_bits is determined from Table 11-4 (Cipher suite key lengths).

KCK || KEK || TK = KDF-X(PMK, “FILS PTKSA Derivation”, SPA || AA || SNonce || ANonce)

Where:

— X is 384+TK\_bits or 640+TK bits from Table 11-4 (Cipher suite key lengths) depending on the AKM used

* PMK is the PMK from the PMKSA, either created from an initial FILS connection or from a cached PMKSA, when PMKSA caching is used.
* SPA is the STA’s MAC address and AA is the AP’s BSSID.
* SNonce is the STA’s nonce and ANonce is the AP’s nonce.

If the negotiated AKM is 00-0F-AC-<ANA-1> or 00-0F-AC-<ANA-2>, FILS requires an additional element: a 13 octet AEAD counter to be part of the newly created PTKSA. The STA shall set the AEAD counter to 13 octets of zero and the AP shall set the first octet to the value 128 and the remaining octets to zero (i.e. the first bit of the AEAD counter is 1 and the rest of the bits in the counter are 0). To allow for proper processing, each side shall include the AEAD counter of the other as a peer's AEAD counter (see 11.11.2.5 (AEAD cipher mode)). AEAD counters are processed per 11.11.2.5 (AEAD cipiher mode for FILS).

**References:**