IEEE P802.11
Wireless LANs

|  |
| --- |
| A Convention for naming the SHA family of Hash Functions |
| Date: 2014-10-29 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | Aruba Networks | 1322 Crossman ave, Sunnyvale, CA 94089United States of America | +1 408 227 4500 | dharkins at aruba networks dot com |
|  |  |  |  |  |

Abstract

This submission proposes a convention for naming of the SHA family of hash functions in both their native form and in HMAC constructs to resolve CIDs 3426, 3427, and 3429. The convention is the following where [] indicates conditional information and n is a truncation length:

 [HMAC-]SHA-<name>[-n] – for <name> being either 1, 256, or 384

This seems much better than the alternative being proposed in 11-14/1104rX which is:

 SHA-<name> —for <name> being 1, 256, or 384

 HMAC-SHA<name>[-n]—for <name> being either 256 or 384

 HMAC-SHA-1 when there is no truncation

 HMAC-SHA1-n when there is truncation

Or, “use the dash in native format but don’t use a dash in the HMAC construct unless it’s the HMAC construct of SHA-1 then use a dash if there’s no truncation but don’t use a dash if there is.”

|  |  |  |  |
| --- | --- | --- | --- |
| CID |  Comment |  Proposed Change | Resolution |
| 3426 | "HMAC-SHA-256" (6 instances) is confusing as 256 is not the output length | "HMAC-SHA256" | Reject: yes, 256 is the output length. And there is no confusion as the convention is:[HMAC-]SHA-<name>[-n] |
| 3247 | "HMAC-SHA1-128" | "Truncate128(HMAC SHA1-160" for consistency with other PMKIDs.  Also at 1935.35 | Counter: consistency is achieved by consistently following the convention:[HMAC-]SHA-<name>[-n] |
| 3429 | Is it SHA256 or is it SHA-256?  Ditto SHA(‑)384 | Pick one (or two, if the answers for the hash name on its own and when combined to form a HMAC (e.g. HMAC-SHA256) are different, to avoid confusion between the hash name and the output length) | Counter: better to pick one and stick to it. That one is:[HMAC-]SHA-<name>[-n] |

***Instruct the editor to modify in section 8.4.2.24.3 as indicated:***

**8.4.2.24.3 AKM suites**

 **Table 8-140—AKM suite selectors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00-0F-AC | 5 | Authentication negotiated over IEEE Std 802.1X or using PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA keymanagement) with SHA-256 Key Derivation | RSNA Key Management as defined in 8.5 or using PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA key management), with SHA-256 Key Derivation | Defined in 11.6.1.7.2 (Key derivation function (KDF)) |
| 00-0F-AC | 6 | PSK with SHA-256 Key Derivation | RSNA Key Management as defined in 11.6 (Keys and key distribution) using PSK with SHA-256 Key Derivation | Defined in 11.6.1.7.2 (Key derivation function (KDF)) |
| 00-0F-AC | 8 | SAE authentication with SHA-256 or using PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA key management) with SHA-256 key derivation | RSNA key management as defined in 11.6 (Keys and key distribution), PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA key management) with SHA-256 key derivation or authenticated mesh peering exchange as defined in 13.5 (Authenticated mesh peering exchange (AMPE)) | Defined in 11.6.1.7.2 (Key derivation function (KDF)) |
| 00-0F-AC | 9 | FT authentication over SAEwith SHA-256 | FT key management defined in 11.6.1.7 (FT key hierarchy) | Defined in 11.6.1.7.2 (Key derivation function (KDF)) |
| 00-0F-AC | 10 | APPeerKey Authenticationwith SHA-256 or usingPMKSA caching as defined in11.5.10.3 (Cached PMKSAsand RSNA key management)with SHA-256 KeyDerivation | RSNA key management as defined in 11.6 (Keys and key distribution) or using PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA key management) with SHA-256 Key Derivation | Defined in 11.6.1.7.2 (Key derivation function (KDF)) |
| 00-0F-AC | 11 | Authentication negotiatedover IEEE 802.1X or usingPMKSA caching as defined in11.5.10.3 (Cached PMKSAsand RSNA key management)using a Suite B compliantEAP method supporting EC ofGF(p=256) | RSNA key management as defined in 11.6 (Keys and key distribution) or using PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA key management) with HMAC-SHA-256 | Defined in 11.6.1.7.2 (Key derivation function (KDF)) using HMAC-SHA-256 |
| 00-0F-AC | 12 | Authentication negotiatedover IEEE 802.1X or usingPMKSA caching as defined in11.5.10.3 (Cached PMKSAsand RSNA key management)using a Suite B compliantEAP method supporting EC ofGP(p=384) | RSNA key management as defined in 11.6 (Keys and key distribution) or using PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA key management) with HMAC-SHA-384 | Defined in 11.6.1.7.2 (Key derivation function (KDF)) using HMAC-SHA-384 |
| 00-0F-AC | 13 | FT authentication negotiatedover IEEE 802.1X | FT key management as defined in 11.6.1.7 (FT key hierarchy) with HMAC-SHA-384 | Defined in 11.6.1.7.2 (Key derivation function (KDF)) using HMAC-SHA-384 |

The AKM suite selector value 00-0F-AC:8 (i.e., SAE authentication with SHA-256 or using PMKSA caching as defined in 11.5.10.3 (Cached PMKSAs and RSNA key management) with SHA-256 key derivation) is used when either a password or PSK is used with RSNA key management.

***Instruct the editor to modify section 8.4.2.96 as indicated:***

**8.4.2.96 Emergency Alert Identifer element**

The Alert Identifier Hash (AIH) is an 8-octet field. It is a unique value used to indicate an instance of an

EAS message. The value of this field is the hash produced by the HMAC-SHA-1-64 hash algorithm

operating on the EAS message.

AIH =HMAC-SHA-1-64(“ES\_ALERT”, Emergency\_Alert\_Message)

***Instruct the editor to modify section 10.25.7 as indicated:***

**10.25.7 Interworking procedures: emergency alert system (EAS) support**

The Alert Identifier Hash in the Emergency Alert Identifier element shall be computed using HMAC-SHA-1-64 hash algorithm as shown in 8.4.2.96 (Emergency Alert Identifier element).

***Instruct the editor to modify section 11.3.2 as indicated:***

**11.3.2 Assumptions on SAE**

When used with AKMs 00-0F-AC:8 or 00-0F-AC:9 from Table 8-140 (AKM suite selectors), H is instantiated as HMAC-SHA-256:

H(salt, ikm) = HMAC-SHA-256(salt, ikm)

When used with AKMs 00-0F-AC:8 or 00-0F-AC:9 from Table 8-140 (AKM suite selectors), CN is instantiated as a function that takes a key, a counter, and a sequence of data. Each piece of data is converted to an octet string and concatenated together before being concatenated to the counter and passed, along with the key, to HMAC-SHA-256:

CN(key, counter,X, Y, Z, …) = HMAC-SHA-256(key, counter || D2OS(X) || D2OS(Y) || D2OS(Z) || …)

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

**11.6.1.2 PRF**

When the negotiated AKM is 00-0F-AC:5, 00-0F-AC:6, or 00-0F-AC:11, 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-SHA-256-128(K, A, B)

 PRF-192(K, A, B) = KDF-SHA-256-192(K, A, B)

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

 PRF-384(K, A, B) = KDF-SHA-256-384(K, A, B)

 PRF-512(K, A, B) = KDF-SHA-256-512(K, A, B)

 When the negotiated AKM is 00-0F-AC:12, 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-SHA-384-704(K, A, B)

When the negotiated AKM is 00-0F-AC:13, 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-384(K, A, B) = KDF-SHA-384-384(K, A, B)

PRF-512(K, A, B) = KDF-SHA-384-512(K, A, B)

PRF-704(K, A, B) = KDF-SHA-384-704(K, A, B)

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

**11.6.1.3 Pairwise key hierarchy**

A PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-1-128(PMK, "PMK Name" || AA || SPA)

Here, HMAC-SHA-1-128 is the first 128 bits of the HMAC-SHA-1 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:11, 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:12, 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 5—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-SHA-1-128 based derivation is used for the PMKID calculation.

***Instruct the editor to modify section 11.6.1.6 as indicated:***

**11.6.1.6 PeerKey key hierarchy**

A SMK identifier is defined as

SMKID = HMAC-SHA-1-128(SMK, "SMK Name" || PNonce || MAC\_P || INonce || MAC\_I)

Here, HMAC-SHA-1-128 is the first 128 bits of the HMAC-SHA-1 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 SMKID, and an SMK identifier is defined as

SMKID = Truncate-128(HMAC-SHA-256(SMK, "SMK Name" || PNonce || MAC\_P || INonce || MAC\_I))

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

**11.6.1.7.3 PMK-R0**

* If the AKM negotiated is 00-0F-AC:3, then Hash shall be SHA-256, 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 Std 802.1X authentication), i.e., XXKey = L(MSK, 256, 256). If the AKM negotiated is 00-0F-AC:4, then Hash shall be SHA-256, 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 SHA-256, 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:13, then Hash shall be SHA-384, 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).

The PMK-R0 is referenced and named as follows:

PMKR0Name = Truncate-128(SHA-256("FT-R0N" || PMK-R0Name-Salt))

where

* "FT-R0N" is 0x46 0x54 0x2D 0x52 0x30 0x4E.
* Truncate-128(-) returns the first 128 bits of its argument and securely destroys the remainder.
* SHA-256 is as defined in FIPS PUB 180-3-2008

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

**11.6.1.7.4 PMK-R1**

* If the AKM negotiated is 00-0F-AC:3, 00-0F-AC:4, or 00-0F-AC:9, then Hash shall be SHA-256, and Z shall be 256. If the AKM negotiated is 00-0F-AC:13, then Hash shall be SHA-384, and Z shall be 384.

The PMK-R1 is referenced and named as follows:

PMKR1Name = Truncate-128(SHA-256(“FT-R1N” || PMKR0Name || R1KH-ID || S1KHID))

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

**11.6.1.7.5 PTK**

If the AKM negotiated is 00-0F-AC:3, 00-0F-AC:4, or 00-0F-AC:9, then Hash shall be SHA-256. If the AKM negotiated is 00-0F-A:13, then Hash shall be SHA-384.

The PTK is referenced and named as follows:

PTKName = Truncate-128(SHA-256(PMKR1Name || “FT-PTKN” || SNonce || ANonce || BSSID || STA-ADDR))

***Instruct the editor to modify section 11.6.2 as indiated:***

**11.6.2 EAPOL-Key frames**

b) Key Information . This field is 2 octets and specifies characteristics of the key. See Figure 11-33 (Key Information bit layout).

The bit convention used is as in 11.2 of IEEE Std 802.1X-2010. The subfields of the Key Information field are as follows:

1) Key Descriptor Version (bits 0–2) shall be set to 0 on all transmitted EAPOL-Key frames except under the following circumstances:

ii) The value 2 shall be used for all EAPOL-Key frames to a STA when the negotiated AKM is 00-0F-AC:1 or 00-0F-AC:2 and either the pairwise or the group cipher is an enhanced data cryptographic encapsulation mechanism other than TKIP for Key Descriptor 2. This value indicates the following:

* + HMAC-SHA-1-128 is the EAPOL-Key MIC. HMAC is defined in IETF RFC 2104; and SHA-1, by FIPS PUB 180-3-2008. The output of the HMAC-SHA-1 shall be truncated to its 128 MSBs (octets 0–15 of the digest output by HMAC-SHA-1), i.e., the last four octets generated shall be discarded.

***Instruct the editor to modify section 11.6.3 as indicated:***

**11.6.3 EAPOL-Key frame construction and processing**

 **Table 11-8—Integrity and key-wrap algorithms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 00-0F-AC:1 | HMAC-SHA-1-128 | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:2 | HMAC-SHA-1-128 | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:11 | HMAC-SHA-256 | 128 | 16 | NIST AES Key Wrap | 128 |
| 00-0F-AC:12 | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 |
| 00-0F-AC:13 | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 |

***Instruct the editor to modify sections 11.6.6.2 through 11.6.6.5, 11.6.7.2 through 11.6.7.3, 11.6.8.2 through 11.6.8.6, 11.6.8.4, and 11.6.8.5.1 as indicated (they all have identical text to change):***

**11.6.6.2-5 4-Way Handshake Message 1-4**

**11.6.7.2-3 Group Key Handshake Message 1-2**

**11.6.8.2-6 SMK Handshake Message 1-5**

**11.6.8.4 STKSA rekeying**

**11.6.8.5.1 General**

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0

***Instruct the editor to modify section 11.6.9.2 as indicated:***

**11.6.9.2 TDLS Peer Key handshake**

The TPK shall be derived as follows:

TPK-Key-Input = SHA-256(min (SNonce, ANonce) || max (SNonce, ANonce))

TPK = KDF-N\_KEY(TPK-Key-Input, "TDLS PMK", min (MAC\_I, MAC\_R) || max (MAC\_I, MAC\_R) || BSSID)

***Instruct the editor to modify section 11.10.1 as indicated:***

**11.10.1 AP PeerKey overview**

When used with AKM 10 from Table 8-140 (AKM suite selectors) to indicate AP PeerKey, H shall be instantiated as HMAC-SHA-256:

H(salt, ikm) = HMAC-SHA-256(salt, ikm)

***Instruct the editor to modify section 11.10.2 as indicated:***

**11.10.2 AP PeerKey protocol**

The PMK shall be derived from the keyseed using the key derivation function (KDF) from 11.6.1.7.2 (Key derivation function (KDF)) using Equation (11-4) and the PMKID shall be derived according to Equation (11-5).

PMK = KDF-256(keyseed , “AP Peerkey Protocol”,

0x00 || Max(LOCAL-MAC, PEER-MAC) || Min(LOCAL-MAC, PEER-MAC) ) (11-4)

PMKID = Truncate-128(SHA-256(Q1 || Q2 ||

Max(LOCAL-MAC, PEER-MAC) ||

Min(LOCAL-MAC, PEER-MAC)) (11-5)

***Instruct the editor to modify section B.4.4.1 as indicated:***

**B.4.4.1 MAC protocol capabilities**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PC 34.1.11PC 34.1.12 | AKM: IEEE Std 802.1X authentication with SHA-256 PRFAKM PSK with SHA-256 PRF | 8.4.2.24 (RSNE), 11.6 (Keys and key distribution)8.4.2.24 (RSNE), 11.6 (Keys and key distribution) | PC34:0PC34:0 | Yes  No  N/A Yes  No  N/A  |

**References:**