IEEE P802.11
Wireless LANs

|  |
| --- |
| Changes to KDF, PRF, PKEX  |
| Date: 2016-09-07 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE | 1322 Crossman avenue, Sunnyvale, California, 94089, United States of America | +1 408 555 1212 | First dot last at hpe dot com |
|  |  |  |  |  |

Abstract

This submission addresses some soon-to-be comments in the next round of TGai balloting.

Discussion

D10.1 modifies 12.7.1.7.2 in the base specification to indicate that KDF used for FILS AKMs, not PRF. But the text in the base standard has changed so that text needs to move to 12.7.1.2.

In looking at that change it was determined that the definition of what bits to generate with KDF was wrong. There are multiple PRFs defined in 12.7.1.2 depending on the number of bits needed to be produced. The number of bits required is calculated as KCK\_bits + KEK\_bits + TK\_bits. In addition, two of the FILS AKMs generate an additional FILS-FT key. But there were no PRFs defined to support all of these key sizes so additional PRF-XXX()s need to be defined.

Also, since FILS uses an AEAD there are really no KCK bits since authentication of unencrypted information is provided by the KEK in an AEAD so table 12-8 needed to be fixed. FILS use of an integrity check for FILS handshaking mentions a KCK but this key is not used the same way the key by the name of KCK is used in the base standard so FILS should change that name.

Also, there was an editorial mistake in table 9-133.

***Instruct the editor to modify table 9-133 in section 9.4.2.25.3 as indicated:***

**9.4.2.25.3 AKM suites**

 **Table 9-133—AKM suite selectors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00-0F-AC |  16 |  FT authentication over FILS with SHA-256 and AES-SIV-256 | FT authentication defined in 12.7.1.7.2 (Key derivation function (KDF)) | Defined in 12.7.1.7.2 (Key derivation function (KDF)) using SHA-256 |

***Instruct the editor to modify section 12.7.1.2 as indicated:***

**12.7.1.2 PRF**

When the negotiated AKM is 00-0F-AC:5, 00-0F-AC:6, or 00-0F-AC:11, the KDF specified in 12.7.1.7.2 (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 12.7.1.7.2 (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 12.7.1.7.2 (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)

When the negotiated AKM is 00-0F-AC:14 or 00-0F-AC:16, the KDF specified in 12.7.1.7.2 (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-256-384(K, A, B)

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

PRF-640(K, A, B) = KDF-SHA-256-640(K, A, B)

 PRF-768(K, A, B) = KDF-SHA-256-768(K, A, B)

 PRF-896(K, A, B) = KDF-SHA-256-896(K, A, B)

 PRF-1024(K, A, B) = KDF-SHA-256-1024(K, A, B)

When the negotiated AKM is 00-0F-AC:15 or 00-0F-AC:17, the KDF specified in 12.7.1.7.2 (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-640(K, A, B) = KDF-SHA-384-640(K, A, B)

PRF-768(K, A, B) = KDF-SHA-384-768(K, A, B)

PRF-1024(K, A, B) = KDF-SHA-384-1024(K, A, B)

 PRF-1152(K, A, B) = KDF-SHA-384-1152(K, A, B)

 PRF-1408(K, A, B) = KDF-SHA-384-1408(K, A, B)

 PRF-1536(K, A, B) = KDF-SHA-384-1536(K, A, B)

***Instruct the editor to delete the changes made to 12.7.1.7.2 in the TGai draft***

***Instruct the editor to modify table 12-8 in section 12.7.3 as indicated:***

 **Table 12-8—Integrity and Key Wrap Algorithms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  AKM |  Integrity  Algorithm  |  KCK bits |  Size of MIC |  Key-wrap  algorithm |  KEK bits |
| 00-0F-AC:14 |  AES-SIV-256 |  0 |  0 |  AES-SIV-256 |  256 |
| 00-0F-AC:15 |  AES-SIV-512 |  0 |  0 |  AES-SIV-512 |  512 |
| 00-0F-AC:16 |  AES-SIV-256 |  0 |  0 |  AES-SIV-256 |  256 |
| 00-0F-AC:17 |  AES-SIV-512 |  0 |  0 |  AES-SIV-512 |  512 |

***Instruct the editor to modify section 12.12.2.5.1 as indicated:***

**12.12.2.5.1 General**

When not using PMKSA caching, a PMK is created according to 12.12.2.5.2 (PMKSA key derivation with FILS authentication). 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 12.7.1.7.2 (Key derivation function (KDF)) to derive the following keys from the PMK: an integrity check key (ICK), a key encryption key (KEK), and a temporal key (TK). PTKSA key establishment shall immediately be followed by key confirmation per 12.12.2.6 (Key confirmation with FILS authentication).

***Instruct the editor to modify section 12.12.2.5.3 as indicated:***

**12.12.2.5.3 PTKSA key derivation with FILS authentication**

For PTKSA key generation, the inputs to the PRF are the PMK of the PMKSA, a constant label, and a concatenation of the STA’s MAC address, the AP’s BSSID, the STA’s nonce, and the AP’s nonce. When the AKM negotiated is 00-0F-AC:14 or 00-0F-AC:16, the length of KEK shall be 256 bits, and the length of the ICK shall be 256 bits. When the AKM negotiated is 00-0F-AC:15 or 00-0F-AC:17, the length of the KEK shall be 512 bits, and the length of ICK shall be 384 bits. When the AKM negotiated is 00-0F-AC:16, FILS-FT is 256 bits; when AKM negotiated if 00-0F-AC:17, FILS-FT is 384 bits; otherwise, FILS-FT is not derived. The total amount of bits extracted from the KDF shall therefore be 512+TK bits, 896+TK bits, or 1280+TK bits depending on the AKM negotiated, where TK\_bits are determined from Table 12-4 (Cipher suite key lengths):

FILS-Key-Data = PRF-X(PMK, “FILS PTK Derivation”, SPA || AA || SNonce || ANonce)

ICK = L(FILS-Key-Data, 0, ICK\_bits)

KEK = L(FILS-Key-Data, ICK\_bits, KEK\_bits)

TK = L(FILS-Key-Data, ICK\_bits + KEK\_bits, TK\_bits)

When doing FT initial mobility domain association using FILS authentication,

FILS-FT = L(FILS-Key-Data, ICK\_bits + KEK\_bits + TK\_bits, FILS-FT\_bits)

where:

* ICK\_bits is the length of ICK in bits.
* KEK\_bits is the length of KEK in bits.
* FILS-FT\_bits is the length of FILS-FT in bits when doing FT initial mobility domain association using FILS authentication.
* X is 512+TK\_bits, 768+TK bits, 896+TK bits, or 1280+TK bits from Table 12-4 (Cipher suite key lengths) depending on the AKM negotiated.
* MK 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 the AA is the AP’s BSSID.
* SNonce is the STA’s nonce and ANonce

***Instruct the editor to modify section 12.12.2.6.2 as indicated:***

**12.12.2.6.2 (Re)Association Request for FILS key confirmation**

The STA constructs a (Re)Association Request frame for FILS authentication per 9.3.3.6 (Association Request frame format) and 9.3.3.8 (Reassociation Request frame format). Hash functions are used to generate the FILS Key Confirmation element and the specific hash function depends on the AKM negotiated (9.4.2.25.3 (AKM suites)).

For FILS shared key authentication, the KeyAuth field of the FILS Key Confirmation element is constructed by using the HMAC mode of the negotiated hash function with a key of ICK on a concatenation of the STA’s nonce, the AP’s nonce, the STA’s MAC address, the AP’s BSSID, and conditionally the STA’s public Diffie-Hellman value and the AP’s public Diffie-Hellman value, in that order:

Key-Auth = HMAC-Hash(ICK, SNonce || ANonce || STA-MAC || AP-BSSID [ || gSTA || gAP ])

where:

* Hash is the hash function specific to the negotiated AKM.
* SNonce is the STA's nonce, ANonce is the AP’s nonce.
* STA-MAC is the MAC address of the STA and AP-BSSID is the BSSID of the AP.
* gSTA is the STA’s Diffie-Hellman public value and gAP is the AP’s Diffie-Hellman public value.
* The brackets indicate the inclusion of the Diffie-Hellman public values when doing PFS with FILS shared key authentication; there are no Diffie-Hellman public values to include otherwise.

If authentication is deemed a failure, ICK, KEK, TK and the PTKSA shall be irretrievably deleted and the AP shall return an Authentication frame with a status code set to 112 (Authentication rejected due to FILS authentication failure). If PMKSA caching was not being employed for this failed authentication attempt, the PMKSA shall also be deleted. If PMKSA caching was being used, the cached PMKSA may not be deleted.

***Instruct the editor to modify section 12.12.2.6.3 as indicated:***

**12.12.2.6.3 (Re)Association Response for FILS key confirmation**

The AP constructs a (Re)Association Response frame for FILS authentication per 9.3.3.7 (Association Response frame format) and 9.3.3.9 (Reassociation Response frame format). As with the (Re)Association Request frame, hash functions are used to generate the FILS Key Confirmation element and the specific hash function depends on the AKM negotiated (see 9.4.2.25.3 (AKM suites)).

The AP constructs a Key Delivery element indicating the current GTK and Key RSC, the current IGTK and IPN if management frame protection is enabled. The GTK is carried in a GTK KDE with Tx subfield equal to 0. The IGTK and IPN are carried in an IGTK KDE. The AP puts this element into the (Re)Association Response frame.

For FILS shared key authentication, the KeyAuth field of the FILS Key Confirmation element is constructed by using the HMAC mode of the negotiated hash function with a key of ICK on a concatenation of the AP’s nonce, the STA’s nonce, the AP’s BSSID, the STA’s MAC address, and conditionally the AP’s public Diffie-Hellman value and the STA’s public Diffie-Hellman value, in that order:

Key-Auth = HMAC-Hash(ICK, ANonce || SNonce || AP-BSSID || STA-MAC [ || gAP || gSTA ])

where:

* Hash is the hash function specific to the negotiated AKM.
* ANonce is the AP’s nonce and SNonce is the STA’s nonce.
* AP-BSSID is the BSSID of the AP and STA-MAC is the MAC address of the STA.
* gAP is the AP’s Diffie-Hellman public value and gSTA is the STA’s Diffie-Hellman public value.
* The brackets indicate the inclusion of the Diffie-Hellman public values when doing PFS with FILS shared key authentication; there are no Diffie-Hellman public values to include otherwise.

If authentication is deemed a failure, the ICK, KEK, PMK, and TK shall be irretrievably deleted and the

STA shall abandon the exchange. Otherwise authentication succeeds and the STA and AP shall irretrievably

delete the nonpersistent secret keying material that is created by executing the key establishment with FILS

shared key authentication scheme (12.12.2.3 (Key establishment with FILS shared key authentication)) or

the key establishment with FILS public key authentication scheme (12.12.2.4 (Key establishment with FILS

public key authentication)). The KEK and PMK shall be used for subsequent key management as

specified in 12.6 (RSNA security association management). If the lifetime of the rMSK is known, the STA

and AP shall set the lifetime of the PMKSA to the lifetime of the rMSK. Otherwise, the STA and AP shall

set the lifetime of the PMKSA to the value dot11RSNAConfigPMKLifetime.

Discussion

PKEX’s security considerations rely on public keys not being sent multiple times through PKEX. This makes its utility in FILS somewhat less than ideal. There are better ways to solve this problem—parlaying a simple shared key/code/word/phrase into a trusted public key—that do not have such usage constraints so let’s remove PKEX from FILS and try a different route in 11md.

***Instruct the editor to remove clause 12.7.12 in its entirety***

**References:**