IEEE P802.11
Wireless LANs

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |
| --- |
| LB270 KCK and KEK definition clarification |
| Date: 2023-1-24 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Po-Kai Huang | Intel  |  |  | po-kai.huang@intel.com |
| Ido Ouzieli | Intel |  |  |  |
| Ilan Peer | Intel |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

 |

Abstract

This submission proposes resolutions for the following comments from comment collection on P802.11-REVme D2.0:

3743, 3744, 3745, 3746

**Revision History:**

R0: Initial version.

R1: Revision based on offline feedback.

R2: Revision based on discussion during teleconference

# CID 3743, 3744, 3745, 3746

|  |  |  |
| --- | --- | --- |
| **CID****Clause****Page.Line** | **Comment** | **Proposed Change** |
| 3743 | Acronym "KCK" is defined as EAPOL-Key confirmation key in 3.4. This is not correct for many reference in the spec. For example, we have "TPK-KCK", but if we follow the definition, then it will be TDLS PeerKey EAPOL-Key confirmation key. However, TPK has nothing to do with EAPOL KCK. As another example, we have SAE KCK, but if we follow the definition, then it will be "simultaneous authentication of equals EAPOL-Key confirmation key" However, SAE has nothing to do withthe EAPOL KCK. | Change the description of KCK to simply "key confirmation key" Commenter is willing to submit contribution for the task. |
| 3745 | Acronym "KEK" is defined as EAPOL-Key encryption key in 3.4. This may not be correct for many reference in the spec. For example, we have KEK in FT, but FT does not use EAPOL to derive KEK. | Change the description of KEK to simply "key encryption key" Commenter is willing to submit contribution for the task. |
| 3744 | KCK is used extensively in 12. There are many different context for KCK like TPK-KCK or SAE-KCK or EAPOL-KCK. However, sometimes there is prefix for the KCK, and sometimes there is no prefix for the KCK, and it creates confusion on which KCK is referred to. | Go through all instances of KCK in the subclause and if there is no prefix for the KCK, adding apropriate prefix after confirming the context. Consider using PTK-KCK when the KCK is related to the context of PTK. Commenter is willing to submit contribution for the task. |
| 3746 | KEK is used extensively in 12. There are many different context for KEK like KEK in 4-way and KEK in FT. However, the spec seems to imply that KEK always mean EAPOL-KEK, which may be meaningless for FT. | Add prefix like PTK-KEK. Commenter is willing to submit contribution for the task. |

## Discussion:

KCK is defined officially as EAPOL key confirmation key as described in 3.4. Similarly, KEK is defined officially as EAPOL key encryption key. However, KCK and KEK is also used in other contexts like FT, where EAPOL key frame is not even used. Similar consideration for term like TPK-KCK, which makes the addition of EAPOL meaningless. Suggest to simply have KCK as key confirmation key and KEK as key encryption key.

**3.4 Acronyms and abbreviations**

|  |  |
| --- | --- |
| KCK  | EAPOL-Key confirmation key |
| KEK  | EAPOL-Key encryption key |

## Proposed Resolution: CID 3743 and 3745

**REVISED**

**Instruction to TGme Editor:**

Implement the proposed text updates for CID 3743 and 3745 in 11-23/0153r1

## Proposed Text Update: CID 3743 and CID 3745

*Instruction to TGme Editor: Update REVme D2.0 3.4 as shown below (track change on).*

**3.4 Acronyms and abbreviations**

|  |  |
| --- | --- |
| KCK  | key confirmation key |
| KEK  | key encryption key |

## Discussion:

It is clear that KEK and KCK can be used in different contexts, and the current spec use the following terminology.

Terminology 1: prefix and description

*EAPOLKey confirmation key (KCK) portion of the PTK.*

Terminology 2: no prefix and some description

*The KCK and KEK used for a group key handshake are the KCK and KEK derived by the 4-way handshake initiated by the same Authenticator that is initiating the group key handshake.*

*KCK portion of the PTK*

Terminology 3 with prefix:

*TPK-KCK, TPK-KEK, TPK-TK*

Suggest to simply unify the style by adding PTK prefix to KCK and KEK during 4-way handshake, FT, and other contexts that represent KCK and KEK derived under PTK. Fix the TPK session to always for prefix for KCK rather than sometimes TPK-KCK and sometimes just KCK. Do not have prefix for scenario like KCK\_bits, KEK\_bits, or TK\_bits, where TK\_bits is used in generic manner in table 12-8. Do not do further changes for KCK2 and KEK2 since there is no confusion.

## Proposed Resolution: CID 3744 and 3746

**REVISED**

**Instruction to TGme Editor:**

Implement the proposed text updates for CID 3744 and 3746 in 11-23/0153r1

## Proposed Text Update: CID 3744 and CID 3746

*Instruction to TGme Editor: Update REVme D2.0 12.7.8 as shown below (track change on).*

* **TDLS PeerKey (TPK) security protocol**
* **TPK handshake**

(…existing texts….)

(#2012)A TPK is a transient key. Each TPK has two component keys—TPK-KCK and TPK-TK, defined as follows:

The TPK key confirmation key (TPK-KCK) shall be computed as the first 128 bits (i.e., bits 0–127) of the TPK.

TPK-KCK = L(TPK, 0, 128)

The TPK-KCK is used to provide data origin authenticity in TDLS Setup Response and TDLS Setup Confirm frames.

The TPK temporal key (TPK-TK) shall be computed as the remaining bits (for CCMP-128, the second 128 bits, i.e., bits 128–255) of the TPK

TPK-TK = L(TPK, 128, Length – 128)

The TPK-TK is used to provide confidentiality for direct link(#1356) data.

The temporal key is configured into the STA by the SME through the use of the MLME-SETKEYS.request primitive.

*Instruction to TGme Editor: Update REVme D2.0 4.10.4 as shown below (track change on).*

* **IBSS functional model description**
* **General**

This subclause summarizes the system setup and operation of an RSNA in an IBSS. An IBSS RSNA is specified in 12.6.11 (RSNA authentication in an IBSS).

* **Key usage**

In an IBSS the individually addressed Data frames between two STAs are protected with a pairwise key. The key is part of the PTK, which is derived during a 4-way handshake. In an IBSS the 4-way handshake can follow IEEE 802.11 authentication of one STA to another. Such authentication might be used by the peer to cause deletion of the PTKSA and (#1405)block the Controlled Port, resetting any previous handshake.

In an IBSS group addressed Data frames are protected by a key, e.g., named B1, that is generated by the STA transmitting the group addressed frame. To allow other STAs to decrypt group addressed frames, B1 is sent to all of the other STAs in the IBSS. B1 is sent in an (#1836)EAPOL-Key PDU, encrypted under the PTK key encryption key (PTK-KEK), and protected from modification by the PTK key confirmation key (PTK-KCK).

In an IBSS the SME responds to Deauthentication frames from a STA by deleting the PTKSA associated with that STA.

* **Sample IBSS 4-way handshakes**

In this example (see Figure 4-33 (Sample 4-way handshakes in an IBSS)), there are three STAs: S1, S2, S3. The group addressed frames sent by S1 are protected by B1; similarly B2 for S2, and B3 for S3.

For STAs S2 and S3 to decrypt group addressed frames from S1, B1 is sent to S2 and S3. This is done using the 4-way handshake initially and using the group key handshake for GTK updates.

The 4-way handshake from S1 to S2 allows S1 to send group addressed frames to S2, but does not allow S2 to send group addressed frames to S1 because S2 has a different transmit GTK. Therefore, S2 needs to initiate a 4-way handshake to S1 to allow S1 to decrypt S2’s group addressed frames. Similarly, S2 also needs to initiate a 4-way handshake to S3 to enable S3 to receive group addressed frames from S2.

In a similar manner S3 needs to complete the 4-way handshake with S1 and S2 to deliver B3 to S1 and S2.

In this example, there are six 4-way handshakes. In general, *N* Supplicants require *N*(*N*–1) 4-way handshakes.

NOTE—In principle the PTK-KCK and PTK-KEK from a single 4-way handshake can be used for the group key handshake in both directions, but using two 4-way handshakes means the Authenticator key state machine does not need to be different between IBSS and ESS.

The group key handshake can be used to send the GTKs to the correct STAs. The 4-way handshake is used to derive the pairwise key and to send the initial GTK. Because in an IBSS there are two 4-way handshakes between any two Supplicants and Authenticators, the pairwise key used between any two STAs is from the 4-way handshake initiated by the STA Authenticator with the higher MAC address (see 12.7.1 (Key hierarchy) for the notion of address comparison).

*Instruction to TGme Editor: Update REVme D2.0 9.4.2.24.3 as shown below (track change on).*

* AKM suites

(…existing texts…)

NOTE 4—The selector values 00-0F-AC:8 and 00-0F-AC:9 have the length of the PMK in bits equal to 256, the length of the PTK-KCK in bits equal to 128, and the length of the PTK-KEK in bits equal to 128 (see 12.7.1.3 (Pairwise key hierarchy), and 12.7.3 (EAPOL-Key PDU construction and processing)). The selector values 00-0F-AC:24 and 00-0F-AC:25 have the length of the PMK, the length of the SAE-KCK and PTK-KCK, and the length of PTK-KEK depending on the hash algorithm specified in 12.4.2 (Assumptions on SAE) (see 12.7.1.3 (Pairwise key hierarchy) and 12.7.3 (EAPOL-Key PDU construction and processing)). (M21)(M67)

(…existing texts…)

*Instruction to TGme Editor: Update REVme D2.0 12.6.1.1.2 PMKSA as shown below (track change on).*

* **PMKSA**

(…existing texts…)

When the negotiated AKM uses PMKID derivation with PTK-KCK as a parameter as defined in 12.7.1.3 (Pairwise key hierarchy), the PMKID derived from the PTK-KCK during the initial 4-way handshake is not changed during the lifetime of this PMKSA.

(…existing texts…)

*Instruction to TGme Editor: Update REVme D2.0 12.6.15 RSNA key management in an IBSS as shown below (track change on).*

12.6.15 RSNA key management in an IBSS

(…existing texts….)

Each Authenticator uses the PTK-KCK and PTK-KEK negotiated by the exchange it initiates to distribute its own GTK and if management frame protection is enabled, its own IGTK.

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.6.22.2 Nontransparent multi-band RSNA as shown below (track change on).*

* **Nontransparent multi-band RSNA**

…(existing texts)….

If the Joint Multi-band RSNA subfield within the RSN Capabilities field is 1 for both the RSNA initiator and the RSNA responder and at least one of the STAs uses different MAC addresses for different bands/channels, the STAs shall use a single 4-way handshake to negotiate pairwise cipher suites and establish PTKSAs for both the current operating band/channel and the other supported band(s)/channel(s). As specified in 12.7.6 (4-way handshake), message 2 and message 3 of the 4-way handshake convey the RSNE and the Multi-band element(s). The RSNE in message 2 includes the selected pairwise cipher suite for the current operating band/channel, and the Multi-band element(s) in message 2 includes the selected pairwise cipher suite(s) for the other supported band(s)/channel(s). message 3 includes the RSNE and the Multi-band element(s) that the STA would send in a Beacon, DMG Beacon, Announce, Probe Response, or Information Response frame. message 3 may include a second RSNE and Multi-band element(s) that indicate the STA’s pairwise cipher suite assignments for the current operating band/channel and the other supported band(s)/channel(s). PTK-KCK and PTK-KEK associated with the current operating band/channel shall be used in the 4-way handshake.

*Instruction to TGme Editor: Update REVme D2.0 12.7.1.3 Pairwise key hierarchy as shown below (track change on).*

* **Pairwise key hierarchy**

Except when preauthentication or FILS authentication 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 12-30 (Pairwise key hierarchy(11ba)). When using AKM suite selector 00-0F-AC:12, 00-0F-AC:15, 00-0F-AC:20,(#590) 00-0F-AC:23(M20)(M67), the length of the PMK, PMK\_bits, shall be 384 bits. (M67)When using AKM suite selector 00-0F-AC:24 or 00-0F-AC:25, the length of the PMK, PMK\_bits, shall have the length of the digest generated by H() identified in 12.4.2 (Assumptions on SAE). When using AKM suite selectors for which the Authentication type column indicates FT authentication (see Table 9-188 (AKM suite selectors)), the FT key hierarchy is used to derive session specific keys from an MPMK as defined in 12.7.1.6 (FT key hierarchy). 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 PTK-KCK, PTK-KEK, (11ba)a temporal key, and a KDK if WUR frame protection is negotiated; otherwise the PTK is partitioned into PTK-KCK, PTK-KEK, and a temporal key. The temporal key is used by the MAC to protect individually addressed communication between the Authenticator’s and Supplicant’s respective STAs. If WUR frame protection is negotiated, the KDK is used to derive a WTK, which is used by the MAC of the WUR AP to protect and by the MAC of the WUR non-AP STA to validate individually addressed WUR Wake-up frames. PTKs are used between a single Supplicant and a single Authenticator.





When using IEEE 802.1X authentication, the PMK is derived from the MSK. The PMK shall be computed as the first PMK\_bits bits (bits 0 to PMK\_bits–1) of the MSK: PMK = L(MSK, 0, PMK\_bits). When using SAE or FILS authentication, the PMK is derived per 12.4.5.4 (Processing of a peer’s SAE Commit message) or 12.11.2.5.2 (PMKSA key derivation with FILS authentication), respectively.

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 [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 [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 when using TKIP. 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.

The following apply when not using FILS authentication:

* 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-Length(PMK, “Pairwise key expansion”, Min(AA,SPA) || Max(AA,SPA) || Min(ANonce,SNonce) || Max(ANonce,SNonce))

where (11ba)Length = KCK\_bits + KEK\_bits + TK\_bits + KDK\_bits, if WUR frame protection is being negotiated; otherwise, Length = KCK\_bits + KEK\_bits + TK\_bits. The values of KCK\_bits and KEK\_bits are AKM suite dependent and are listed in Table 12-11 (Integrity and key wrap algorithms). The value of TK\_bits is cipher-suite dependent and is defined in Table 12-8 (Cipher suite key lengths(#1083)). (11ba)The value of KDK\_bits is equal to the value of PMK\_bits. 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 nonces are encoded as specified in 9.2.2 (Conventions).

* (11ba)If WUR frame protection is being negotiated, the KDK shall be computed as the next KDK\_bits bits of the PTK:

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

Otherwise, the KDK is not derived.

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

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

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

The PTK-KCK is used by IEEE Std 802.1X-2010 to provided data origin authenticity in the 4-way handshake and group key handshake messages.

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

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

The PTK-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 bits of the PTK:

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

(…existing texts…)

*Instruction to TGme Editor: Update REVme D2.0 12.7.1.6.5 PTK as shown below (track change on).*

* **PTK**

(…existing texts….)

(11ba)Except when WUR frame protection is negotiated, each PTK has five component keys, PTK-KCK, PTK-KEK, a temporal key, KCK2, and KEK2 derived as follows:

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

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

The PTK-KCK is used to provide data origin authenticity in EAPOL-Key frames, as defined in 12.7.2 (EAPOL-Key frames), and in the FT authentication sequence, as defined in 13.8 (FT authentication sequence).

(11ba)When WUR frame protection is negotiated, each PTK has six component keys, PTK-KCK, PTK-KEK, a temporal key, KCK2, KEK2, and a KDK derived as follows:

(11ba)The PTK-KCK, PTK-KEK, temporal key, KCK2, and KEK2 shall be computed in the same way as when WUR frame protection is not negotiated.

(…existing texts….)

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

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

The PTK-KEK is used to provide data confidentiality for certain fields (KeyData) in EAPOL-Key frames, as defined in 12.7.2 (EAPOL-Key frames), and in the FT authentication sequence, as defined in 13.8 (FT authentication sequence).

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.2 EAPOL-Key frames as shown below (track change on).*

* **EAPOL-Key frames**

(…existing texts….)

* **EAPOL-Key IV**. This field contains the IV used with the PTK-KEK. It shall contain 0 when an IV is not required. It should be initialized by taking the current value of the global key counter (see 12.7.10 (RSNA Authenticator key management state machine)) and then incrementing the counter. Note that only the lower 16 octets of the counter value are used.

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.4 EAPOL-Key PDU notation* *as shown below (track change on).*

**EAPOL-Key PDU notation(#1836)**

(…existing texts….)

MIC is the integrity check, which is generated using the PTK-KCK. This is the Key MIC field. (#1831)When using an AEAD cipher, (#1825)this parameter is ignored, and no Key MIC field is included in the EAPOL-Key PDU(#216).

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.6.1 General as shown below (track change on).*

* **General**

(…existing texts….)

* The MIC is computed over the body of the (#1836)EAPOL-Key PDU (with the Key MIC field first zeroed before the computation) using the PTK-KCK defined in 12.7.1.3 (Pairwise key hierarchy) for PTK generation.

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.6.3 4-way handshake message 2 as shown below (track change on).*

* **4-way handshake message 2**

(…existing texts….)

Key MIC = Not present when using an AEAD cipher; otherwise, MIC(PTK-KCK, EAPOL)(#1833)

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.6.4 4-way handshake message 3 as shown below (track change on).*

* **4-way handshake message 3**

(…existing texts….)

Key MIC = Not present when using an AEAD cipher; or otherwise, MIC(PTK-KCK, EAPOL)(#1827)(#1833)

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.6.5 4-way handshake message 4 as shown below (track change on).*

* **4-way handshake message 4**

(…existing texts….)

Key MIC = Not present when using an AEAD cipher; or otherwise, MIC(PTK-KCK, EAPOL)(#1833)

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.6.6 4-way handshake implementation considerations as shown below (track change on).*

* **4-way handshake implementation considerations**

(…existing texts….)

An implementation should save the PTK-KCK and PTK-KEK beyond the 4-way handshake, as they are needed for group key handshakes, and recovery from TKIP MIC failures.

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.7.1 General as shown below (track change on).*

* **General**

(…existing texts….)

The following apply:

* (#1406)RSC denotes the last TSC or PN sent using the GTK.
* GTK[N] denotes the GTK (#1453)with its key identifier as encapsulated using the KDE defined in 12.7.2 (EAPOL-Key frames) using the PTK-KEK defined in 12.7.1.3 (Pairwise key hierarchy) and associated IV.
* IGTK[M], when present, denotes the IGTK (#1453)with its key identifier as encapsulated using the KDE defined in 12.7.2 (EAPOL-Key frames) using the PTK-KEK defined in 12.7.1.3 (Pairwise key hierarchy) and associated IV.
* BIGTK[Q], when present, denotes the BIGTK with its key identifier as encapsulated using the KDE as defined in 12.7.2 (EAPOL-Key frames) using the PTK-KEK defined in 12.7.1.3 (Pairwise key hierarchy) and associated IV.
* (11ba)WIGTK[R], when present, denotes the WIGTK with its key identifier as encapsulated using the KDE as defined in 12.7.2 (EAPOL-Key frames) using the PTK-KEK defined in 12.7.1.3 (Pairwise key hierarchy) and associated IV.
* The MIC is computed over the body of the (#1836)EAPOL-Key PDU (with the MIC field zeroed for the computation) using the PTK-KCK defined in 12.7.1.3 (Pairwise key hierarchy).

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.7.2 Group key handshake message 1 as shown below (track change on).*

* **Group key handshake message 1**

(…existing texts….)

Key MIC = Not present when using an AEAD cipher; otherwise, MIC(PTK-KCK, EAPOL)

(…existing texts….)

* Verifies that the MIC is valid, i.e., it uses the PTK-KCK that is part of the PTK to verify that there is no data integrity error, or that the AEAD decryption steps succeed.

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.7.7.3 Group key handshake message 2 as shown below (track change on).*

* **Group key handshake message 2**

(…existing texts….)

Key MIC = Not present when using an AEAD cipher; otherwise, MIC(PTK-KCK, EAPOL)

(…existing texts….)

* Verifies that the MIC is valid, i.e., it uses the PTK-KCK that is part of the PTK to verify that there is no data integrity error, or that the AEAD decryption steps succeed.

*Instruction to TGme Editor: Update REVme D2.0 13.8.4 FT authentication sequence: contents of third message as shown below (track change on).*

* **FT authentication sequence: contents of third message**

(…existing texts….)

If present, the FTE shall be set as follows:

* ANonce, SNonce, R0KH-ID, and R1KH-ID shall be set to the values contained in the second message of this sequence.
* The Element Count subfield of the MIC Control field shall be set to the number of elements protected in this frame (variable).
* The RSNXE Used subfield of the MIC Control field shall be set to 1 if the FTO set to 1 any subfield, except the Field Length subfield, of the Extended RSN Capabilities field in the RSNXE; otherwise this subfield shall be set to 0.
* When the negotiated AKM is 00-0F-AC:3, 00-0F-AC:4, or 00-0F-AC:9, the MIC shall be calculated using the PTK-KCK and the AES-128-CMAC algorithm. The output of the AES-128-CMAC shall be 128 bits.
* When the negotiated AKM is 00-0F-AC:13, the MIC shall be calculated using the PTK-KCK and the HMAC-SHA-384 algorithm. The output of the HMAC-SHA-384 shall be truncated to 192 bits.
* When the negotiated AKM is 00-0F-AC:16, the MIC shall be calculated using the KCK2 and the AES-128-CMAC algorithm. The output of the AES-128-CMAC shall be 128 bits.
* When the negotiated AKM is 00-0F-AC:17, the MIC shall be calculated using the KCK2 and the HMAC-SHA-384 algorithm. The output of the HMAC-SHA-384 shall be truncated to 192 bits.
* When the negotiated AKM is 00-0F-AC:25(Ed1), the MIC shall be calculated using the PTK-KCK and the HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 algorithm when the length of the PTK-KCK in bits is 128/192/256. The output of the HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 shall be truncated to 128/192/256 bits.(M21)(M67)

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 13.8.5 FT authentication sequence: contents of fourth message as shown below (track change on).*

* **FT authentication sequence: contents of fourth message**

(…existing texts….)

If present, the FTE shall be set as follows:

* ANonce, SNonce, R0KH-ID, and R1KH-ID shall be set to the values contained in the second message of this sequence.
* The Element Count subfield of the MIC Control field shall be set to the number of elements protected in this frame (variable).
* The RSNXE Used subfield of the MIC Control field shall be set to 1 if the target AP includes an RSNXE in its Beacon and Probe Response frames; otherwise this subfield shall be set to 0.
* If dot11RSNAOperatingChannelValidationActivated is true and Supplicant indicates OCVC capability, the Authenticator shall include FT OCI subelement in FTE.
* When this message of the authentication sequence appears in a Reassociation Response frame, the Optional Parameter(s) field in the FTE may include the GTK, IGTK, BIGTK, and WIGTK(11ba) subelements. If a GTK, an IGTK, a BIGTK, or WIGTK(11ba) are included, the Key field of the subelement shall be wrapped using PTK-KEK or KEK2 and the appropriate key wrap algorithm, as specified in Table 12-11 (Integrity and key wrap algorithms) and 12.7.2 (EAPOL-Key frames). The padding consists of appending a single octet 0xdd followed by zero or more 0x00 octets. When processing a received message, the receiver shall ignore this trailing padding. Addition of padding does not change the value of the Key Length field. Note that the length of the encrypted Key field can be determined from the length of the GTK, IGTK(M21), BIGTK, or WIGTK subelement.
* When the negotiated AKM is 00-0F-AC:3, 00-0F-AC:4, or 00-0F-AC:9, the MIC shall be calculated using the PTK-KCK and the AES-128-CMAC algorithm. The output of the AES-128-CMAC algorithm shall be 128 bits.
* When the negotiated AKM is 00-0F-AC:13, the MIC shall be calculated using the PTK-KCK and the HMAC-SHA-384 algorithm. The output of the HMAC-SHA-384 shall be truncated to 192 bits.
* When the negotiated AKM is 00-0F-AC:16, the MIC shall be calculated using the KCK2 and the AES-128-CMAC algorithm. The output of the AES-128-CMAC shall be 128 bits.
* When the negotiated AKM is 00-0F-AC:17, the MIC shall be calculated using the KCK2 and the HMAC-SHA-384 algorithm. The output of the HMAC-SHA-384 shall be truncated to 192 bits.
* When the negotiated AKM is 00-0F-AC:25(Ed1), the MIC shall be calculated using the PTK-KCK and the HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 algorithm when the length of the PTK-KCK in bits is 128/192/256. The output of the HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 shall be truncated to 128/192/256 bits.(M21)(M67)

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 Figure 13-18—S1KH state machine, including portions of the SME (part 1) as shown below (track change on)*



**

*Instruction to TGme Editor: Update REVme D2.0 12.11.2.5.1 General as shown below (track change on).*

**12.11.2.5.1 General**

(…existing texts….)

PTKSA creation uses the KDF from 12.7.1.6.2 (Key derivation function (KDF)) to derive the following keys from the PMK: an integrity check key (ICK), a PTK key encryption key (PTK-KEK), and a temporal key (TK).

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.11.2.5.3 PTKSA Key derivation with FILS authentication as shown below (track change on).*

* **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 BSS’s BSSID, the STA’s nonce, and the AP’s nonce. When the negotiated AKM is 00-0F-AC:14 or 00-0F-AC:16, the length of PTK-KEK shall be 256 bits, and the length of the ICK shall be 256 bits. When the negotiated AKM is 00-0F-AC:15 or 00-0F-AC:17, the length of the PTK-KEK shall be 512 bits, and the length of ICK shall be 384 bits. When the negotiated AKM is 00-0F-AC:16, FILS-FT is 256 bits; when the negotiated AKM is 00-0F-AC:17, FILS-FT is 384 bits; otherwise, FILS-FT is not derived(11ba); when WUR frame protection is negotiated, the length of KDK is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)); otherwise, the KDK is not derived. The total amount of bits extracted from the KDF shall therefore be (11ba)640+TK bits, 1124+TK bits, or 1408+TK bits depending on the negotiated AKM when WUR frame protection is negotiated, otherwise, shall be 512+TK bits, 896+TK bits, or 1280+TK bits depending on the negotiated AKM, where TK\_bits are determined from Table 12-8 (Cipher suite key lengths(#1083)):

PTK(#1778) = PRF-X(PMK, “FILS PTK Derivation”, SPA || AA || SNonce || ANonce [ || DHss ])

ICK = L(PTK(#1778), 0, ICK\_bits)

PTK-KEK = L(PTK(#1778), ICK\_bits, KEK\_bits)

TK = L(PTK(#1778), ICK\_bits + KEK\_bits, TK\_bits)

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 12.11.2.6.1 General as shown below (track change on).*

**General**

Key confirmation for FILS authentication is a (Re)Association Request frame followed by a (Re)Association Response frame. Components of the (Re)Association Request and (Re)Association Response frames shall be protected using PTK-KEK.

*Instruction to TGme Editor: Update REVme D2.0 12.11.2.6.2 (Re)Association Request for FILS key confirmation* *as shown below (track change on).*

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

(…existing texts….)

The (Re)Association Request frame shall be encrypted using the AEAD algorithm as defined in 12.11.2.7 (AEAD cipher mode for FILS) with the PTK-KEK as the key. The AAD used with the AEAD algorithm for the Association Request frame consists of the following data passed as separate components in the following order:

* STA’s MAC address
* AP’s BSS’s BSSID
* STA’s nonce
* AP’s nonce
* The contents of the (Re)Association Request frame from the Capability Information field (inclusive) to the (#482)FILS Session element (inclusive)

(…existing texts….)

The AP decrypts and verifies the received (Re)Association Request frame with the AEAD algorithm as defined in 12.11.2.7 (AEAD cipher mode for FILS) with the PTK-KEK as the key. The AAD is reconstructed as defined above and is passed, along with the cipher text of the received frame, to the AEAD decryption operation.

(…existing texts….)

If authentication is deemed a failure, ICK, PTK-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 employed, the reason for failure might be an impersonation attack. Therefore, when FILS with PMKSA caching fails, the AP may decide to retain the cached PMKSA.

*Instruction to TGme Editor: Update REVme D2.0 12.11.2.6.3 (Re)Association Response for FILS key confirmation* *as shown below (track change on).*

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

(…existing texts….)

The (Re)Association Response frame shall be encrypted using the AEAD algorithm as defined in 12.11.2.7 (AEAD cipher mode for FILS) with the PTK-KEK as the key. The AAD used with the AEAD algorithm for the (Re)Association Response frame consists of the following data passed as separate components in the following order:

* AP’s BSS’s BSSID
* STA’s MAC address
* AP’s nonce
* STA’s nonce
* The contents of the (Re)Association Response frame from the Capability Information field (inclusive) to the FILS Session element (inclusive)

(…existing texts….)

The STA decrypts and verifies the received (Re)Association Response frame with the AEAD algorithm as defined in 12.11.2.5 (Key establishment with FILS authentication) with the PTK-KEK as the key. The AAD is reconstructed as defined in this subclause above and is passed with the cipher text of the received frame to the AEAD decryption operation.

(…existing texts….)

If authentication is deemed a failure, the ICK, PTK-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 (12.11.2.3 (Key establishment with FILS Shared Key authentication)) or the key establishment with FILS Public Key authentication scheme (12.11.2.4 (Key establishment with FILS Public Key authentication)). The PTK-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.

(…existing texts….)

*Instruction to TGme Editor: Update REVme D2.0 13.4.4 FT initial mobility domain association over FILS in an RSN as shown below (track change on).*

* **FT initial mobility domain association over FILS in an RSN**

(…existing texts….)

When FILS authentication is used to establish the FT key hierarchy, TK and PTK-KEK for the initial mobility domain association are derived as part of the FILS authentication as defined in 12.11.2.5.3 (PTKSA Key derivation with FILS authentication).

(…existing texts….)