IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Determine SAE PMK length | | | | |
| Date: 2022-03-01 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Po-Kai Huang | Intel |  |  |  |
| Ido Ouzieli |  |  |  |
| Ilan Peer |  |  |  |
| Johannes Berg |  |  |  |
| Jouni Malinen | Qualcomm |  |  |  |
| Thomas Derham | Broadcom |  |  |  |
| Dan Harkins | HPE |  |  |  |

Abstract

This document proposes a way to differentiate when to use 256 bit PMK and when to use 384 bit PMK for SAE.

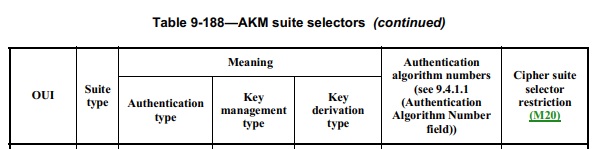
Revisions:

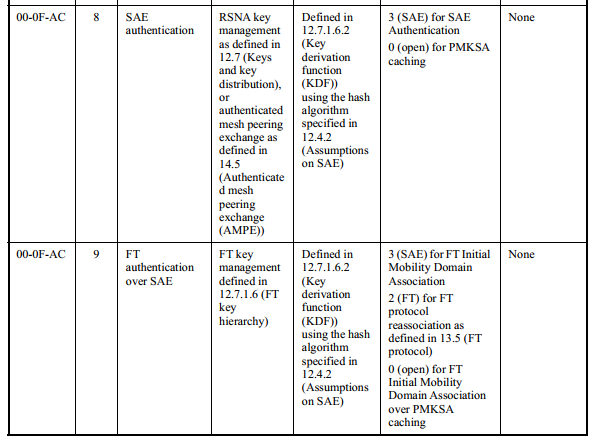
* Rev 0: Initial version of the document.
* Rev 1: Revision based on the comments from Mark Rison and Thomas
* Rev 2: Revision of discussion and texts based on further feedback
* Rev 3: Revision to use an additional element to indicate AKM to avoid confusion of RSNE verification. Have the second SAE message indicate the AKM as well to conform. Revise the PMK bit length to be based on the hash algorithm of SAE based on Dan’s comment. Revise EAPOL-KCK key length and EAPOL-KEK length to be based on hash algorithm of SAE as well, which follows similar design of OWE.
* Rev 4: Editorial revision based on the discussion during teleconference. Revise 12.7.3 (EAPOL-Key frame construction and processing) based on offline discussion. Change in 12.7.3 are marked with green.
* Rev 5: Add cosigners
* Rev 6: Remove the texts related to group 20 and add cosigner

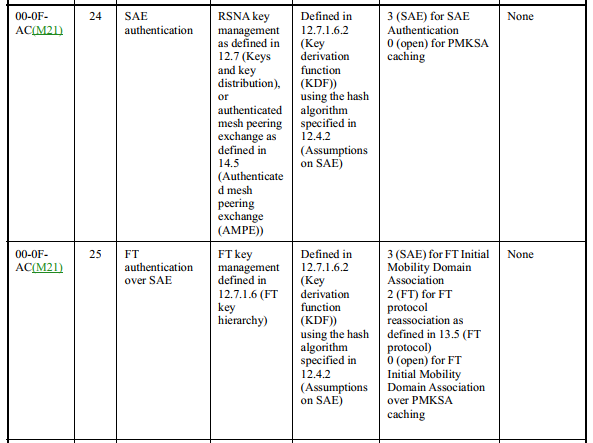
Discussion:

There is a potential issue on how to determine the PMK length for SAE. The reason for the issue is that the current spec ties the length of PMK to the AKM that is used as shown below, but during SAE exchange there is no way to know the AKM since AKM is not carried during SAE exchange because RSNE is not carried during the SAE exchange.

*If used with (M21)AKMs 00-0F-AC:8 or 00-0F-AC:9 and the looping method of PWE generation  
(see 12.4.4.2.2 (Generation of the password element with ECC groups by looping) and 12.4.4.3.2 (Generation of the password element with FFC groups by looping)), both the KCK and PMK shall be 256 bits in length. If used with AKMs 00-0F-AC:8 or 00-0F-AC:9 and the hash-to-element method(#344) of PWE generation (see 12.4.4.2.3 (Hash-to-element(#331) generation of the password element with ECC groups) and 12.4.4.3.3 (Direct generation of the password element with FFC groups)), the KCK shall be the length of the digest generated by H() and the PMK shall be 256 bits in length (M21)(see 12.7.1.3 (Pairwise key hierarchy)). When AKM 00-0F-AC:24 or 00-0F-AC:25 is negotiated, the hash-to-element method(#344) of PWE generation (see 12.4.4.2.3 (Hash-to-element(#331) generation of the password element with ECC groups) and 12.4.4.3.3 (Direct generation of the password element with FFC groups)) shall be used, and the KCK shall be the length of the digest generated by H() and the PMK shall be 384 bits in length (see 12.7.1.3 (Pairwise key hierarchy)).*



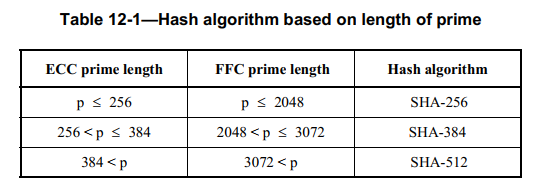




It maybe argued that we can use the Hash algorithm used by SAE to determine the AKM that is used, which then determines the PMK length, since SHA-256 with group 19, where an ECC group defined over a 256-bit prime order field, is mandated to be implemented as shown below and is commonly used for AKMs 00-0F-AC:8 or 00-0F-AC:9

*For the purpose of interoperability, a STA shall implement support for group 19, an ECC group defined over a 256-bit prime order field.*

However, it is also ture that the spec does not limit SHA-384 or SHA-512 to be used for AKMs 00-0F-AC:8 or 00-0F-AC:9 although there is no specific advantage to use longer SHA to increase the security strength. Therefore, using the hash algorithm still may not determine the AKM to be used without considering backward compatibility.



To resolve the problem, the easiest way seems to include AKM in the very first message of SAE so that there will be no ambiguity. The second message can then indicate the same AKM.



To make sure that we have legacy compatiability, we can also include AKM 00-0F-AC:24 or 00-0F-AC:25 in the first message of SAE only when AKM 00-0F-AC:24 or 00-0F-AC:25 is supported by the AP during discovery and intended to be used by the client.

If AKM is not included in the first message of the SAE, then we can follow the existing procedure of using AKMs 00-0F-AC:8 or 00-0F-AC:9 before AKMs 00-0F-AC:24 and 00-0F-AC:25 are introduced.

We use a new element to indicate the AKM rather than RSNE to avoid confusion of RSNE verification.

For future extension without the need to introduce further AKM, we let the PMK/EAPOL-KCK/EAPOL-KEK length depends on the hash algorithm length used by the SAE for the new AKM. This follows similar design principle of OWE.

Clarification on PRF usage and for PTK based on AKM suite selctor table is also done.

Texts are provided below based on the above discussion and received offline feedback.

**Propose:**

**9.4.2 Elements**

**9.4.2.1 General**

***TGme editor:******Insert a new row to*** [***Table 9-128 (Element IDs(#1009)(#1121))***](#bookmark88)***:***

**Table 9-128—Element IDs(#1009)(#1121)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Element ID** | **Element ID Extension** | **Extensible** | **Fragmentable** |
| **AKM Suite Selector (see 9.4.2.XXX (AKM Suite Selector element))** | 255 | <ANA> | Yes | No |

*TGme editor: Add a new subclause in 9.4.2 Elements as follows:*

**9.4.2.XXX AKM Suite Selector element**

The AKM Suite Selector element is used to indicate the intended AKM in SAE.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Element ID | Length | Element ID Extension | AKM Suite Selector |
| Octets: | 1 | 1 | 1 | 4 |

Figure 9-xxx – AKM Suite Selector element format

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1 (General).

The AKM Suite Selector field has the format shown in Figure 9-349—(Suite selector format), where the suite type is defined in Table 9-188 (AKM suite selectors).

*TGme editor: Modify 12.4.5.3 Construction of an SAE Commit message as follows: (track change on)*

**12.4.5.3 Construction of an SAE Commit message**

The scalar and element in an SAE Commit message shall be produced using ***PWE*** and secrets generated in 12.4.5.2 (PWE and secret generation), as follows:

*commit*-*scalar* = (*rand* + *mask*) mod *r****COMMIT-ELEMENT*** = inverse-op(scalar-op(*mask*, ***PWE***))

This message shall be transmitted to the peer as described in 12.4.7 (Framing of SAE). The temporary secret *mask* may be deleted at this point.

To derive keys for use with AKM 00-0F-AC:24 or AKM 00-0F-AC:25, an AKM Suite Selector element indicating 00-0F-AC:24 or 00-0F-AC:25 shall be included in an SAE Commit message transmitted to the peer.

If an SAE Commit message that includes an AKM Suite Selector element has been received, the AKM indicated n the AKM Suite Selector element is supported, and a SAE Commit message is constructed, then the SAE Commit message shall include an AKM Suite Selector element that indicates the same AKM.

* Processing of a peer’s SAE Commit message

*TGme editor: Add paragraphs after the second paragraph in 12.4.5.4 Processing of a peer’s SAE Commit message as follows: (track change on)*

If the peer’s SAE Commit message contains a password identifier, the value of that identifier shall be used in  
construction of the password element (PWE) for this exchange. If a password identifier is present in the peer’s  
SAE Commit message and there is no password with the given identifier a STA shall fail authentication.

If the peer’s SAE Commit message contains a Rejected Groups element, the list of rejected groups shall be  
checked to ensure that all of the groups in the list are groups that would be rejected. If any groups in the list  
would not be rejected then processing of the SAE Commit message terminates and the STA shall reject the  
peer’s authentication. While the rejected groups are appended to the Rejected Groups element as they are  
rejected (see 12.4.7.4 (Encoding and decoding of SAE Commit messages)) there is no inherent order to the  
groups in the list. The order in which they are sent and received shall be retained when deriving keys.

If the state of the SAE finite state machine is *Committed* (see 12.4.8.2.2 Protocol instance states) and the SAE Commit message that has been sent by the SAE finite state machine to transition into *Committed* state includes an AKM Suite Selector element, the authentication shall fail if either of the following conditions is true:

* the peer’s SAE Commit message does not contain an AKM Suite Selector element
* the peer’s SAE Commit message contains an AKM Suite Selector element and the AKM Suite Selector element does not indicate the same AKM

*TGme editor: Modify the following pagraph as follows (track change on):*

The entropy of *k* shall then be extracted using H to produce *keyseed*. The key derivation function from 12.7.1.6.2 (Key derivation function (KDF)) shall then be used with the hash algorithm identified for H() (see 12.4.2 (Assumptions on SAE)) to derive a key confirmation key, KCK, and a pairwise master key, PMK, from *keyseed*.

The intended AKM for the purpose of PMK and KCK size determination (see below) is determined as follows:

* If an AKM Suite Selector element is not included in the SAE Commit message from the peer and the state of the SAE finite state machine is *Nothing* (see 12.4.8.2.2 Protocol instance states), then 00-0F-AC:8 or 00-0F-AC:9 shall be the intended AKM.
* If the state of the SAE finite state machine is *Committed* (see 12.4.8.2.2 Protocol instance states) and the SAE Commit message that has been sent by the SAE finite state machine to transition into *Committed* state does not include an AKM Suite Selector element, then 00-0F-AC:8 or 00-0F-AC:9 shall be the intended AKM.
* If an AKM Suite Selector element that indicates AKM 00-0F-AC:24 or AKM 00-0F-AC:25 is included in the SAE Commit message from the peer and the state of the SAE finite state machine is *Nothing* (see 12.4.8.2.2 Protocol instance states), then the indicated AKM shall be the intended AKM
* If the state of the SAE finite state machine is *Committed* (see 12.4.8.2.2 Protocol instance states) and the SAE Commit message that has been sent by the SAE finite state machine to transition into *Committed* state includes an AKM Suite Selector element that indicates AKM 00-0F-AC:24 or AKM 00-0F-AC:25, then the indicated AKM shall be the intended AKM

If the intended AKM is (M21) 00-0F-AC:8 or 00-0F-AC:9 and the looping method of PWE generation (see 12.4.4.2.2 (Generation of the password element with ECC groups by looping) and 12.4.4.3.2 (Generation of the password element with FFC groups by looping)), both the KCK and PMK shall be 256 bits in length. If the intended AKM is 00-0F-AC:8 or 00-0F-AC:9 and the hash-to-element method(#344) of PWE generation (see 12.4.4.2.3 (Hash-to-element(#331) generation of the password element with ECC groups) and 12.4.4.3.3 (Direct generation of the password element with FFC groups)), the KCK shall have the length of the digest generated by H() and the PMK shall be 256 bits in length (M21)(see 12.7.1.3 (Pairwise key hierarchy)). If the intended AKM is 00-0F-AC:24 or 00-0F-AC:25, the hash-to-element method(#344) of PWE generation (see 12.4.4.2.3 (Hash-to-element(#331) generation of the password element with ECC groups) and 12.4.4.3.3 (Direct generation of the password element with FFC groups)) shall be used, the KCK and the PMK shall have the length of the digest generated by H(). Use of other AKMs with the hash-to-element method(#344) will require definition of the length of the PMK. If both SAE Commit messages indicated a status code of SAE\_HASH\_TO\_ELEMENT, a salt consisting of the  
concatenation of the rejected groups from each peer’s Rejected Groups element shall be passed to the KDF; those of the peer with the highest MAC address go first (if only one sent a Rejected Groups element then the salt will consist of that list). If neither peer sent a Rejected Groups element or the status code was not SAE\_HASH\_TO\_ELEMENT, the salt shall consist of a series of octets of the value zero whose length equals the length of the digest of the hash function used to instantiate H().

*TGme editor: Modify 9.3.3.11 Authentication frame format as follows (track change on):*

* **Authentication frame format**

The frame body of an Authentication frame contains the information shown in Table 9-68 (Authentication frame body). FT authentication is used when FT support is advertised by the AP and dot11FastBSSTransitionActivated is true in the STA. SAE authentication is used when dot11MeshActiveAuthenticationProtocol is sae (1). FILS authentication is used if support for FILS authentication is advertised by the AP and dot11FILSActivated is true in the STA.

|  |  |  |
| --- | --- | --- |
| * **Authentication frame body** | | |
| **Order** | **Information** | **Notes** |
| 1 | Authentication algorithm number |  |
| 2 | Authentication transaction sequence number |  |
| 3 | Status code | The status code information is reserved in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 4 | Finite Cyclic Group | An unsigned integer indicating a finite cyclic group as described in 9.4.1.42 (Finite Cyclic Group field). This is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 5 | Anti-Clogging Token | A random bit string used for anti-clogging purposes as described in 12.4.6 (Anti-clogging tokens). This is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 6 | Send-Confirm | A binary encoding of an integer used for anti-replay purposes as described in 12.4.7.5 (Encoding and decoding of SAE Confirm messages). This is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 7 | Scalar | An unsigned integer encoded as described in 12.4.7.4 (Encoding and decoding of SAE Commit messages). This is present only in cer-tain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 8 | FFE(#312) | An element in a finite field encoded as described in 12.4.7.4 (Encoding and decoding of SAE Commit messages). This is present only in cer-tain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 9 | Confirm | An unsigned integer encoded as described in 12.4.7.5 (Encoding and decoding of SAE Confirm messages). This is present only in cer-tain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 10 | Challenge text | A Challenge Text element is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 11 | RSN | An RSNE is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 12 | Mobility Domain | An MDE is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 13 | Fast BSS Transition | An FTE is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 14 | Timeout Interval (reassociation deadline) | A TIE containing the reassociation deadline interval is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 15 | RIC | A resource information container, containing a variable number of elements, is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 16 | Multi-band | The Multi-band element is optionally present if dot11MultibandImplemented is true. |
| 17 | Neighbor Report | One or more Neighbor Report elements are present only in cer-tain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 18 | FILS Nonce | The FILS Nonce element is present in FILS Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 19 | FILS Session | The FILS Session element is present in FILS Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 20 | FILS Wrapped Data | The FILS Wrapped Data element is present in FILS Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 21 | Association Delay Info | The Association Delay Info element is present in FILS Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 22 | Password Identifier | The Password Identifier element is optionally present in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 23 | Rejected Groups | The Rejected Groups element is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 24 | Anti-Clogging Token Container | The Anti-Clogging Token Container element is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| 25 | AKM Suite Selector | The AKM Suite Selectorelement is present only in certain Authentication frames as defined in Table 9-69 (Presence of fields and elements in Authentication frames). |
| Last | Vendor Specific | One or more Vendor Specific elements are optionally present. These elements follow all other elements. |

|  |  |  |  |
| --- | --- | --- | --- |
| * **Presence of fields and elements in Authentication frames** | | | |
| **Authentication algorithm** | **Authentication transaction sequence number** | **Status code** | **Presence of fields and elements  from order 4 onward** |
| Open System | 1 | Reserved | Not present |
| Open System | 2 | Not REJECTED\_WITH\_SUGGESTED\_BSS\_TRANSITION | Not present |
| Open System | 2 | REJECTED\_WITH\_SUGGESTED\_BSS\_TRANSITION | One or more Neighbor Report element(s) is present |
| Shared Key | 1 | Reserved | Not present |
| Shared Key | 2 | Any | The Challenge Text element is present |
| Shared Key | 3 | Reserved | The Challenge Text element is present |
| Shared Key | 4 | Any | Not present |
| FT | 1 | Reserved | The Mobility Domain element is present.  The Fast BSS Transition element and RSNEs are present if dot11RSNAActivated is true. |
| FT | 2 | Not REJECTED\_WITH\_SUGGESTED\_BSS\_TRANSITION | The Mobility Domain element is present if the Status Code field is 0.  The Fast BSS Transition element and RSNEs are present if the Status Code field is 0 and dot11RSNAActivated is true. |
| FT | 2 | REJECTED\_WITH\_SUGGESTED\_BSS\_TRANSITION | One or more Neighbor Report element(s) is present |
| FT | 3 | Reserved | The Mobility Domain element is present.  The Fast BSS Transition element and RSNEs are present if dot11RSNAActivated is true.  The RIC element is optionally present. |
| FT | 4 | Any | The Mobility Domain element is present if the Status Code field is 0.  The Fast BSS Transition element and RSNEs are present if dot11RSNAActivated is true.  The RIC element is optionally present if the Status Code field is 0.  The TIE (reassociation deadline) is present if a RIC element is present. |
| SAE | 1 | Any | The Scalar field is present if the Status Code field is zero or 126.  The FFE field is present if the Status Code field is zero or 126.  When the hunting-and-pecking method is used to drive the PWE, the Anti-Clogging Token field is present if the Status Code field is ANTI\_CLOGGING\_TOKEN\_REQUIRED or if the Authentication frame is in response to a previous rejection with the Status Code field equal to ANTI\_CLOGGING\_TOKEN\_REQUIRED.  The Finite Cyclic Group field is present if the Status Code field is zero, ANTI\_CLOGGING\_TOKEN\_REQUIRED, 77 or 126.  The Password Identifier element is optionally present if the Status Code field is zero, 123 or 126.  (#288)If the Status Code field is 126, the Rejected Groups element is conditionally present as described in 12.4.7.4 (Encoding and decoding of SAE Commit messages); otherwise the Rejected Groups element is not present. When the hash-to-element method is used to derive the PWE, the Anti-Clogging Token Container element is present if the Status Code field is ANTI\_CLOGGING\_TOKEN\_REQUIRED or if the Authentication frame is in response to a previous rejection with the Status Code field equal to ANTI\_CLOGGING\_TOKEN\_REQUIRED.  The AKM Suite Selector element is present if 00-0F-AC:24 or 00-0F-AC:25 is the intended AKM (see 12.4.5.3 (Construction of an SAE Commit message) and 12.4.5.4 (Processing of a peer’s SAE Commit message)); otherwise, it is not present. |
| SAE | 2 | Not REJECTED\_WITH\_SUGGESTED\_BSS\_TRANSITION | The Send-Confirm field is present.  The Confirm field is present. |
| SAE | 2 | REJECTED\_WITH\_SUGGESTED\_BSS\_TRANSITION | One or more Neighbor Report element(s) are present |
| FILS Shared Key authentication without PFS | 1 | Reserved | The RSNE is present.  The MDE is present if the FILS authentication is used for FT initial mobility domain association.  The FILS Nonce element is present.  The FILS Session element is present.  The FILS Wrapped Data element is present. |
| FILS Shared Key authentication without PFS | 2 | Status | The RSNE is present.  The MDE and the FTE are present if the Status Code field is 0 and FILS authentication is used for FT initial mobility domain association.  The FILS Nonce element is present if the Status Code field is 0.  The FILS Session element is present if the Status Code field is 0.  The FILS Wrapped Data element is present if the Status Code field is 0.  The Association Delay Info element is present if the Status Code field is 0 and the AP expects that the (Re)Association Response frame will be transmitted more than 1 TU after the (Re)Association Request frame. |
| FILS Shared Key authentication with PFS | 1 | Reserved | The Finite Cyclic Group field is present.  The FFE field is present.  The RSNE is present.  The MDE is present if the FILS authentication is used for FT initial mobility domain association.  The FILS Nonce element is present.  The FILS Session element is present.  The FILS Wrapped Data element is present. |
| FILS Shared Key authentication with PFS | 2 | Status | The Finite Cyclic Group field(#313) is present if the Status Code field is 0.  The FFE field is present if the Status Code field is 0.  The RSNE is present.  The MDE and the FTE are present if the Status Code field is 0 and FILS authentication is used for FT initial mobility domain association.  The FILS Nonce element is present if the Status Code field is 0.  The FILS Session element is present if the Status Code field is 0.  The FILS Wrapped Data element is present if the Status Code field is 0.  The Association Delay Info element is present if the Status Code field is 0 and the AP expects that the (Re)Association Response frame will be transmitted more than 1 TU after the (Re)Association Request frame. |
| FILS Public Key authentication | 1 | Reserved | The Finite Cyclic Group field is present.  The FFE field is present.  The RSNE is present.  The MDE is present if the FILS authentication is used for FT initial mobility domain association.  The FILS Nonce element is present.  The FILS Session element is present. |
| FILS Public Key authentication | 2 | Status | The Finite Cyclic Group field(#313) is present if the Status Code field is 0.  The FFE field is present if the Status Code field is 0.  The RSNE is present.  The MDE and the FTE are present if the Status Code field is 0 and FILS authentication is used for FT initial mobility domain association.  The FILS Nonce element is present if the Status Code field is 0.  The FILS Session element is present if the Status Code field is 0.  The Association Delay Info element is present if the Status Code field is 0 and the AP expects that the (Re)Association Response frame will be transmitted more than 1 TU after the (Re)Association Request  frame. |

*TGme editor: Change Clause 9.4.2.24.3* *as follows (track change on):*

* AKM suites

(…existing texts…)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| * **AKM suite selectors** | | | | | | |
| **OUI** | **Suite type** | **Meaning** | | | **Authentication algorithm numbers  (see 9.4.1.1 (Authentication Algorithm Number field))** | **Cipher suite selector restriction (M20)** |
| **Authentication  type** | **Key management  type** | **Key derivation type** |
| 00-0F-AC | 0 | Reserved | Reserved | Reserved | Reserved | Reserved |
| 00-0F-AC | 1 | Authentication negotiated over  IEEE Std 802.1X | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.2 (PRF) | 0 (open) | None |
| 00-0F-AC | 2 | PSK | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.2 (PRF) | 0 (open) | None |
| 00-0F-AC | 3 | FT authentication negotiated over  IEEE Std 802.1X | FT key management as defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | 2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association over  IEEE Std 802.1X or PMKSA caching | None |
| 00-0F-AC | 4 | FT authentication using PSK | FT key management as defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | 2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association using PSK | None |
| 00-0F-AC | 5 | Authentication negotiated over  IEEE Std 802.1X | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | 0 (open) | None |
| 00-0F-AC | 6 | PSK | RSNA Key Management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | 0 (open) | None |
| 00-0F-AC | 7 | TDLS | TPK handshake | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | N/A | None |
| 00-0F-AC | 8 | SAE authentication | RSNA key management as defined in 12.7 (Keys and key distribution), or authenticated mesh peering exchange as defined in 14.5 (Authenticated mesh peering exchange (AMPE)) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using the hash algorithm specified in 12.4.2 (Assumptions on SAE) | 3 (SAE) for SAE Authentication  0 (open) for PMKSA caching | None |
| 00-0F-AC | 9 | FT authentication over SAE | FT key management defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using the hash algorithm specified in 12.4.2 (Assumptions on SAE) | 3 (SAE) for FT Initial Mobility Domain Association  2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association over PMKSA caching | None |
| 00-0F-AC | 10 | APPeerKey Authentication with SHA-256 | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | N/A | None |
| 00-0F-AC | 11 | Authentication negotiated over  IEEE Std 802.1X using a Suite B compliant EAP method supporting SHA-256 | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | 0 (open) | Used only with cipher suite selector values 00-0F-AC:8 (GCMP-128) and 00-0F-AC:11 (BIP-GMAC-128) |
| 00-0F-AC | 12 | Authentication negotiated over  IEEE Std 802.1X using a CNSA Suite compliant EAP method | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-384 | 0 (open) | Used only with cipher suite selector values 00-0F-AC:9 (GCMP-256), 00-0F-AC:10 (CCMP-256), 00-0F-AC:13 (BIP-CMAC-256), and 00-0F-AC:12 (BIP-GMAC-256) |
| 00-0F-AC | 13 | FT authentication negotiated over  IEEE Std 802.1X | FT key management as defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-384 | 2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association over  IEEE Std 802.1X or PMKSA caching | Used only with cipher suite selector values 00-0F-AC:9 (GCMP-256), 00-0F-AC:10 (CCMP-256), 00-0F-AC:13 (BIP-CMAC-256), and 00-0F-AC:12 (BIP-GMAC-256) |
| 00-0F-AC | 14 | Key management over FILS using SHA-256 and  AES-SIV-256, or authentication negotiated over  IEEE Std 802.1X | FILS key management defined in 12.11.2.5 (Key establishment with FILS authentication) | Defined in 12.11.2.5 (Key establishment with FILS authentication)  using SHA-256 | 4, 5 or 6 (FILS) for FILS Authentication  0 (open) for  IEEE Std 802.1X | None |
| 00-0F-AC | 15 | Key management over FILS using SHA-384 and  AES-SIV-512, or authentication negotiated over  IEEE Std 802.1X | FILS key management defined in 12.11.2.5 (Key establishment with FILS authentication) | Defined in 12.11.2.5 (Key establishment with FILS authentication)  using SHA-384 | 4, 5 or 6 (FILS) for FILS Authentication  0 (open) for  IEEE Std 802.1X | None |
| 00-0F-AC | 16 | FT authentication over FILS with  SHA-256 and  AES-SIV-256 or authentication negotiated over  IEEE Std 802.1X | FT key management as defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-256 | 4, 5 or 6 (FILS) for FT Initial Mobility Domain Association over FILS  2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association over  IEEE Std 802.1X or PMKSA caching | None |
| 00-0F-AC | 17 | FT authentication over FILS with  SHA-384 and  AES-SIV-512, or authentication negotiated over  IEEE Std 802.1X | FT key management as defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-384 | 4, 5 or 6 (FILS) for FT Initial Mobility Domain Association over FILS  2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association over  IEEE Std 802.1X or PMKSA caching | None |
| 00-0F-AC | 18 | Reserved | Reserved | Reserved | Reserved | Reserved |
| 00-0F-AC | 19 | FT authentication using PSK | FT key management as defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-384 | 2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association using PSK | None |
| 00-0F-AC | 20 | PSK | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-384 | 0 (open) | None |
| 00-0F-AC(M20) | 22 | FT authentication negotiated over  IEEE Std 802.1X | FT key management as defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using SHA-384 | 2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association over  IEEE Std 802.1X or PMKSA caching | None |
| 00-0F-AC(M20) | 23 | Authentication negotiated over  IEEE Std 802.1X | RSNA key management as defined in 12.7 (Keys and key distribution) | Defined in 12.7.1.6.2 (Key derivation function (KDF)) using SHA-384 | 0 (open) | None |
| 00-0F-AC(M21) | 24 | SAE authentication | RSNA key management as defined in 12.7 (Keys and key distribution), or authenticated mesh peering exchange as defined in 14.5 (Authenticated mesh peering exchange (AMPE)) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using the hash algorithm specified in 12.4.2 (Assumptions on SAE) | 3 (SAE) for SAE Authentication  0 (open) for PMKSA caching | None |
| 00-0F-AC(M21) | 25 | FT authentication over SAE | FT key management defined in 12.7.1.6 (FT key hierarchy) | Defined in 12.7.1.6.2 (Key derivation function (KDF))  using the hash algorithm specified in 12.4.2 (Assumptions on SAE) | 3 (SAE) for FT Initial Mobility Domain Association  2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol)  0 (open) for FT Initial Mobility Domain Association over PMKSA caching | None |
| 00-0F-AC | 21, 26–255(M21) | Reserved | Reserved | Reserved | Reserved | Reserved |
| Other OUI or CID | Any | Vendor-specific | Vendor-specific | Vendor-specific | Vendor-specific | Vendor-specific |

(…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 KCK in bits equal to 128, and the length of the KEK in bits equal to 128 (see 12.4.5.4 (Processing of a peer’s SAE Commit message), 12.7.1.3 (Pairwise key hierarchy), and 12.7.3 (EAPOL-Key frame 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 KCK, and the length of the KEK depending on the 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 frame construction and processing)). (M21)

(…existing texts…)

* **PRF**

*TGme editor: Add the following pagraph at the end of 12.7.1.2 as follows:*

When the negotiated AKM is 00-0F-AC:8 or 00-0F-AC:9 or 00-0F-AC:24 or 00-0F-AC:25, the KDF specified in 12.7.1.6.2 (Key derivation function (KDF)) shall be used instead of the PRF construction defined here (see Key derivation type in Table 9-188 (AKM suite selectors)). In this case, A is used as the KDF label and B as the KDF context, and

PRF-Length(K, A, B) = KDF-Hash-Length(K, A, B), where Hash is the hash algorithm specified in 12.4.2 (Assumptions on SAE)

* Pairwise key hierarchy

*TGme editor: Modify the following pagraph as follows (track change on):*

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), the length of the PMK, PMK\_bits, shall be 384 bits. 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 KCK, KEK, (11ba)a temporal key, and a KDK if WUR frame protection is negotiated; otherwise the PTK is partitioned into KCK, 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.

**12.7.1.6.3 PMK-R0**

*TGme editor: Modify the following pagraph as follows (track change on):*

If the negotiated AKM is 00-0F-AC:25, then Q is equal to the length of the digest generated by H() identified in 12.4.2 (Assumptions on SAE)and

* MPMK = PMK generated as the result of SAE authentication per 12.4.5.4 (Processing of a peer’s SAE Commit message)
* PMKID is derived as defined in 12.4.5.4 (Processing of a peer’s SAE Commit message)

12.7.3 EAPOL-Key frame construction and processing

***TGme editor: change table 12-10 as follows***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| * **Integrity and key wrap algorithms** | | | | | | | |
| **AKM** | **Integrity algorithm** | **KCK\_bits** | **Size of MIC** | **Key wrap algorithm** | **KEK\_bits** | **KCK2\_bits** | **KEK2\_bits** |
| Deprecated | HMAC-MD5 | 128 | 16 | ARC4 | 128 | 0 | 0 |
| 00-0F-AC:1 | HMAC-SHA-1-128 | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:2 | HMAC-SHA-1-128 | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:3 | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:4 | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:5 | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:6 | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:8 | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:9 | AES-128-CMAC | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:11 | HMAC-SHA-256 | 128 | 16 | NIST AES Key Wrap | 128 | 0 | 0 |
| 00-0F-AC:12 | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |
| 00-0F-AC:13 | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |
| 00-0F-AC:14 | AES-SIV-256 | 0 | 0 | AES-SIV-256 | 256 | 0 | 0 |
| 00-0F-AC:15 | AES-SIV-512 | 0 | 0 | AES-SIV-512 | 512 | 0 | 0 |
| 00-0F-AC:16 | AES-SIV-256/ AES-128-CMAC | 0 | 0/16 | AES-SIV-256/ NIST AES Key Wrap | 256 | 128 | 128 |
| 00-0F-AC:17 | AES-SIV-512/ HMAC-SHA-384 | 0 | 0/24 | AES-SIV-512/ NIST AES Key Wrap | 512 | 192 | 256 |
| 00-0F-AC:19(M20) | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |
| 00-0F-AC:20(M20) | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |
| 00-0F-AC:22(M20) | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |
| 00-0F-AC:23M20) | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |
| 00-0F-AC:24(M21) | HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 (see Table 12-XX (Hash identified in SAE and integrity algorithm)) | 128/192/256 | 16/24/32 | NIST AES Key Wrap | 128/256/256 | 0 | 0 |
| 00-0F-AC:25(M21) | HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 (see Table 12-XX (Hash identified in SAE and integrity algorithm)) | 128/192/256 | 16/24/32 | NIST AES Key Wrap | 128/256/256 | 0 | 0 |

***TGme editor: Add the following table after table 12-11:***

**Table 12-XX – Hash identified in SAE and integrity algorithm**

|  |  |
| --- | --- |
| **Hash identified in 12.4.2 (Assumptions on SAE)** | **Integrity algorithm** |
| SHA-256 | HMAC-SHA-256 |
| SHA-384 | HMAC-SHA-384 |
| SHA-512 | HMAC-SHA-512 |

***TGme editor: change 00-0F-AC:<ANA-AKM-4> in clause 13 to 00-0F-AC:25***

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

*TGme editor: Modify the third pagraph as follows (track change on):*

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 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 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, the MIC shall be calculated using the KCK and the HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 algorithm when the length of the 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)
* If dot11RSNAOperatingChannelValidationActivated is true and Authenticator indicates OCVC capability, the Supplicant(#1380) shall include FT OCI subelement in FTE.
* The MIC shall be calculated on the concatenation of the following data, in the order given here:
* FTO’s MAC address (6 octets)
* Target AP’s MAC address (6 octets)
* Transaction sequence number (1 octet), which shall be set to the value 5 if this is a Reassociation Request frame and, otherwise, set to the value 3
* RSNE
* MDE
* FTE, with the MIC field of the FTE set to 0
* Contents of the RIC-Request (if present)
* RSNXE (if present)
* All other fields shall be set to 0.

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

*TGme editor: Modify the fourth pagraph as follows (track change on):*

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 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 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 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, the MIC shall be calculated using the KCK and the HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512 algorithm when the length of the 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)
* The MIC shall be calculated on the concatenation of the following data, in the order given here:
* FTO’s MAC address (6 octets)
* Target AP’s MAC address (6 octets)
* Transaction sequence number (1 octet), which shall be set to the value 6 if this is a Reassociation Response frame or, otherwise, set to the value 4
* RSNE
* MDE
* FTE, with the MIC field of the FTE set to 0
* Contents of the RIC-Response (if present)
* RSNXE (if present)
* All other fields shall be set to 0.