IEEE P802.11
Wireless LANs

|  |
| --- |
| AKM for SAE |
| Date: 2021-06-29 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Po-Kai Huang | Intel |  |  |  |
| Ido Ouzieli |  |  |  |
| Ilan Peer |  |  |  |
| Johannes Berg |  |  |  |
|  |  |  |  |
|  |  |  |  |

Abstract

This document proposes two new AKMs to allow KCK and KEK bits under SAE that are higher than 128 bits when SHA-384 is used

Revisions:

* Rev 0: Initial version of the document.
* Rev 1: Editorial revision and revision based on the comment received offline.

Discussion:

In the current AKM suite selectors, AKM 8 (SAE) and AKM 9 (SAE under FT) are the only two AKMs for SAE, and the corresponding SHA algorithm for key derivation is based on Table 12-1—Hash algorithm based on length of prime. When GCMP-256 is used under SAE, SHA-384 will be negotiated, but the KCK bits and KEK bits are only 128 bits, which then does not match the 256 bits key length.

Obviously, we can not change Table 12-10 for AKM 8 and AKM 9 for backward compatible reason. As a result, we propose to have two new AKMs to address the issue.



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

* AKM suites

The AKM Suite Count field indicates the number of AKM suite selectors that are contained in the AKM Suite List field. The value 0 is reserved.

The AKM Suite List field contains a series of AKM suite selectors. In an IBSS only a single AKM suite selector is specified because IBSS STAs use the same AKM suite and because there is no mechanism to negotiate the AKMP in an IBSS (see 12.6.5 (RSNA policy selection in an IBSS)).

Each AKM suite selector specifies an AKMP. Table 9-151 (AKM suite selectors) gives the AKM suite selectors defined by this -standard. An AKM suite selector has the format shown in Figure 9-288 (Suite selector format).

|  |
| --- |
| * AKM suite selectors
 |
| OUI | Suite type | Meaning | Authentication algorithm numbers (see 9.4.1.1 (Authentication Algorithm Number field)) |
| Authentication type | Key management type | Key derivation type  |
| 00-0F-AC | 0 | 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) |
| 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) |
| 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 |
| 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 |
| 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) |
| 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) |
| 00-0F-AC | 7 | TDLS | TPK handshake | Defined in 12.7.1.6.2 (Key derivation function (KDF)) using SHA-256 | N/A |
| 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 Authentication0 (open) for PMKSA caching |
| 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 Association2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol) 0 (open) for FT Initial Mobility Domain Association over PMKSA caching |
| 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 |
| 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) |
| 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) |
| 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 |
| 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 Authentication0 (open) for IEEE Std 802.1X |
| 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 Authentication0 (open) for IEEE Std 802.1X |
| 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 FILS2 (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 |
| 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 FILS2 (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 |
| 00-0F-AC | 18 | 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 |
| 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) |
| 00-0F-AC  | <ANA-AKM-1> | 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 Authentication0 (open) for PMKSA caching |
| 00-0F-AC  | <ANA-AKM-2> | 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 Association2 (FT) for FT protocol reassociation as defined in 13.5 (FT protocol) 0 (open) for FT Initial Mobility Domain Association over PMKSA caching |
| 00-0F-AC |  <ANA-AKM-2>+1–255  | Reserved | Reserved | Reserved | Reserved |
| Other OUI or CID | Any | Vendor-specific | Vendor-specific | Vendor-specific | Vendor-specific |

NOTE 1—The selector value 00-0F-AC:1 specifies only that IEEE Std 802.1X-2010 is used as the authentication transport. IEEE Std 802.1X-2010 selects the authentication mechanism.

NOTE 2 - The selector value 00-0F-AC: 8 and 00-0F-AC:9 has PMK\_bits equal to 256, KCK\_bits equal to 128, and KEK\_bits equal to 128 (see 12.4.5.4, 12.7.1.3 and 12.7.3). The selector value 00-0F-AC: <ANA-AKM-1> and 00-0F-AC: <ANA-AKM-2> has PMK\_bits equal to 384, KCK\_bits equal to 192, and KEK\_bits equal to 256 (see 12.7.1.3 and 12.7.3).

The AKM suite selector value 00-0F-AC:8 (i.e., SAE authentication with SHA-256 is used when either a password or PSK is used with RSNA key management.

NOTE 2—Selector values 00-0F-AC:1 and 00-0F-AC:8 can simultaneously be enabled by an Authenticator.

The AKM suite selector value 00-0F-AC:2 (PSK) is used when an alternate form of PSK is used with RSNA key management.

NOTE 3—Selector values 00-0F-AC:1 and 00-0F-AC:2 can simultaneously be enabled by an Authenticator.

The AKM suite selector value 00-0F-AC:11 is used only with cipher suite selector values 00-0F-AC:8 (GCMP-128) and 00-0F-AC:11 (BIP-GMAC-128). The AKM suite selector value 00-0F-AC:12 is 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). The AKM suite selector value 00-0F-AC:13 is 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).

NOTE 4—The AKM suite selector value 00-0F-AC:11 is deprecated.

NOTE 5—The usage of selector values with authentication algorithms is defined in the Authentication algorithm numbers column of Table 9-151 (AKM suite selectors); see 9.4.1.1 (Authentication Algorithm Number field).

A PMKSA established using a given AKM selector value may be cached and used in a subsequent (re)association as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management).

* Pairwise key hierarchy

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

When the negotiated AKM is 00-0F-AC:8 or 00-0F-AC: <ANA-AKM-1>, the PMK identifier is derived as defined in 12.4.5.4.

*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). When using AKM suite selector 00-0F-AC:12 or 00-0F-AC:15 or 00-0F-AC: <ANA-AKM-1> or or 00-0F-AC: <ANA-AKM-2>, the length of the PMK, PMK\_bits, shall be 384 bits. When using AKM suite selectors for which the Authentication type column indicates FT authentication (see Table 9-151 (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, and a temporal key, which is used by the MAC
to protect individually addressed communication between the Authenticator’s and Supplicant’s respective
STAs. PTKs are used between a single Supplicant and a single Authenticator.

* Processing of a peer’s SAE Commit message

*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*. If used with 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 direct hashing technique of PWE generation (see 12.4.4.2.3 (Hash-to-curve 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 (see 12.7.1.3 (Pairwise key hierarchy)). If used with AKMs 00-0F-AC: <ANA-AKM-1> or 00-0F-AC: <ANA-AKM-2> and the direct hashing technique of PWE generation (see 12.4.4.2.3 (Hash-to-curve 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 384 bits in length (see 12.7.1.3 (Pairwise key hierarchy)).Use of other AKMs with the direct hashing technique 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 the following pagraph as follows (track change on):*

*keyseed* = H(*salt*, *k*)

*context* = (*commit-scalar* + *peer-commit-scalar*) mod *r*

*Length* = *Q* + PMK\_bits

*kck\_and\_pmk* = KDF-Hash-*Length*(*keyseed*, “SAE KCK and PMK”, *context*)

*KCK* = L(*kck\_and\_pmk*, 0, *Q*)

*PMK* = L(*kck\_and\_pmk*, *Q*, PMK\_bits)

where

*salt* is either a series of 0 octets or a list of rejected groups (see 12.4.7.4 (Encoding and decoding of SAE Commit messages))

KDF-Hash-*Length* is the key derivation function defined in 12.7.1.6.2 (Key derivation function (KDF)) using the hash algorithm defined for H()

*Q* is the length of the digest of the H(), the hash function used

*context* is treated as an integer and converted into an octet string of length *m* such that 28m > *r* according to 12.4.7.2.2 (Integer to octet string conversion)

PMK\_bits is the length of PMK defined in 12.7.1.3 (Pairwise key hierarchy)

The PMK identifier is defined as follows:

PMKID = L(*context*, 0, 128)

**12.7.1.6.3 PMK-R0**

*TGme editor: Add the following pagraph after “If the negotiated AKM is 00-0F-AC:19, then Q = 384 and…” as follows (track change on):*

If the negotiated AKM is 00-0F-AC: <ANA-AKM-2>, then Q = 384 and
— MPMK = PMK generated as the result of SAE authentication per 12.4.5.4
— PMKID is derived as defined in 12.4.5.4

* 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: <ANA-AKM-1> | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |
| 00-0F-AC: <ANA-AKM-2> | HMAC-SHA-384 | 192 | 24 | NIST AES Key Wrap | 256 | 0 | 0 |

**13.2.2 Authenticator key holders**

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

The R0KH derives the PMK-R0 for use in the mobility domain utilizing the MSK (when the negotiated
AKM is 00-0F-AC:3 or 00-0F-AC:13), the PSK (when the negotiated AKM is 00-0F-AC:4) or the PMK
(when the negotiated AKM is 00-0F-AC:9 or 00-0F-AC: <ANA-AKM-2>), or the FILS-FT (when the negotiated

AKM is 00-0F-AC:16 or 00-0F-AC:17). The R0KH shall be responsible for deriving a PMK-R1 for each R1KH within the mobility domain.

**13.2.3 Supplicant key holders**

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

The S0KH derives the PMK-R0 for use in the mobility domain utilizing the MSK (when the negotiated
AKM is 00-0F-AC:3 or 00-0F-AC:13), the PSK (when the negotiated AKM is 00-0F-AC:4) or the PMK
(when the negotiated AKM is 00-0F-AC:9 or 00-0F-AC: <ANA-AKM-2>), or the FILS-FT (when the negotiated AKM is 00-0F-AC:16 or 00-0F-AC:17)

**13.4.2 FT initial mobility domain association in an RSN**

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

On successful (re)association, the S0KH on the STA and the R0KH on the AP then proceed with an
IEEE 802.1X authentication using EAPOL PDUs carried in IEEE 802.11 Data frames if SAE authentication
was not performed (i.e., if the suite type is not 00-0F-AC:9 or 00-0F-AC: <ANA-AKM-2>). The S0KH shall use the value of R0KH-ID as the endpoint identifier of the NAS Client (NAS-Identifier if RADIUS is used) in the exchange as defined in IETF RFC 3748.

**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: <ANA-AKM-2>, 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.
* If dot11RSNAOperatingChannelValidationActivated is true and Authenticator indicates OCVC
capability, the supplicant 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 and BIGTK subelements. If a
GTK, an IGTK or a BIGTK 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 12-10 and 12.7.2. 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 or BIGTK 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: <ANA-AKM-2>, 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.
* 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.

**References:**