IEEE P802.11
Wireless LANs

|  |
| --- |
| 802.11PMKSA caching and MAC randomization  |
| Date: 2019-04-01 |
| Author(s): |
| Name | Company | Address | Phone | email |
| Thomas Derham | Broadcom |  |  | thomas.derham@broadcom.com |
| Nehru Bhandaru | Broadcom |  |  |  |
| Insun Song | Broadcom |  |  |  |

**Abstract**

This document provides a resolution for comment CID 2689. The document is based on REVmd D2.1.

R0: Initial draft

R1: Addressed review comments from preso at adhoc and subsequent offline comments. In the clarifications regarding PMKSA caching with FT Initial MD Assoc, changes include formalized definition of MPMK (already defined for FT-SAE) as the master PMK from which PMK-R0 is derived, which can be cached in PMKSA bound to AKM. Various clarifications and additions to fully define this behavior for all Auth/AKM cases.

R2 – Rebase on D2.2 (which contains some related approved changes), fix inconsistent text regarding inclusion of PMKID in M1 in non-PMKSA caching cases, fix inconsistent text in the case of using cached PMKSA from SAE with no match on AP side

R3 – Use new RSN Extension element for cap indication (based on 19/114r5)

**Discussion**

The recent 11aq amendment defines MAC address randomization behavior for a non-AP STA, whereby the STA might use a different randomized MAC address when sending probe requests and each time it associates with a network (ESS), in order to overcome privacy concerns resulting from persistent passive tracking of a STA each time it returns to a given location/network based on its MAC address identifier (which is sent in-the-clear in Address fields). The MAC address stays constant while the STA remains associated with the ESS (e.g. while roaming between APs in the ESS).

Given the increasing adoption of computationally-expensive authentication methods such as SAE, it is expected that PMKSA caching for long periods of time (hours, days or more) will become more prevalent.

Currently, 11aq text states that, if the STA uses PMKSA caching, it must return its MAC address to the MAC address that it used when establishing the PMKSA with the peer. In the general case, this is a reasonable rule because legacy peer (AP/Authenticator) implementations might be caching PMKs in a way that is indexed/linked to the STA’s/Supplicant’s MAC address.

However, it is suggested that such restriction is not required from a technical point of view – the peer (AP/Authenticator) can lookup a PMK based only on the PMKID identifier; the STA’s/Supplicant’s MAC address is not a constituent of the PMKSA definition. The fact that the PMK and/or PMKID might have been derived using the MAC address of the STA at the time does not preclude that cached PMK being used again when the STA’s MAC address has changed.

Therefore, this contribution proposes text for such capability, so that MAC randomization can be used in combination with PMKSA caching. This improves client privacy because the STA MAC address, which is sent in-the-clear in almost every frame sent to/from the STA, can be randomized each time the STA connects to the network.

It is noted that the PMKID itself, which is also sent in-the-clear but only in the initial association signaling, is static for the lifetime of the PMK and so could also be used as an identifier for tracking if that signaling is observed. Possible solutions for masking of such identifiers in (pre)association signaling can be handled in separate contributions.

The document also proposes new text and clarifications regarding PMKSA caching with FT Initial Mobility Domain Association (and some fixes in related areas). PMKSA caching with FT Initial Mobility Domain Association is ostensibly supported by the standard, e.g. references to PMKSA Caching in some FT AKM definitions, and this was also confirmed by REVmd motion related to CID 2211, however it is insufficiently specified in several aspects. The Master PMK (MPMK), currently defined for SAE as the “master” PMK from which the FT key hierarchy is defined, is formally defined as the basis for PMKSA caching with AKMs that use FT key management.

*

9.4.2.242 RSN Extension element (RSNXE)

***Instruct the editor to modify table as follows:***

|  |
| --- |
| Table 9-yyy Extended RSN Capabilities field  |
| Bit | Information | Notes |
| 0-3 | Field Length  | The length of the Extended RSN Capabilities field, in octets, minus 1, i.e., *n*-1. |
| 4 | Protected TWT Operations Support | The STA sets the Protected TWT Operations Support field to 1 when dot11ProtectedTWTOperationsImplemented is true, and sets it to 0 otherwise. See 10.48.1 (TWT overview). |
| 5 | PMKSA Caching with MAC Randomization | The STA sets the PMKSA Caching with MAC Randomization field to 1 when dot11PMKSACachingMACRandomizationActivated is true and sets it to 0 otherwise. |
| 6–(8×*n*-1) | Reserved |  |

* AKM suites

***Instruct the editor to modify as follows:***

|  |
| --- |
| * AKM suite selectors
 |
| OUI | Suite type | Meaning |  |
| Authentication type | Key management type | Key derivation type  | Authentication algorithm numbers (see 9.4.1.1 (Authentication Algorithm Number field))(M85) |
| 00-0F-AC | 0 | Reserved | Reserved | Reserved | Reserved |
| 00-0F-AC | 1 | Authentication negotiated over IEEE Std 802.1X or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | RSNA key management as defined in 12.7 (Keys and key distribution) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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), using PSK | Defined in 12.7.1.2 (PRF) | 0 (open) |
| 00-0F-AC  | 3 | FT authentication negotiated over IEEE Std 802.1X or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | FT key management as defined in 12.7.1.6 (FT key hierarchy) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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 or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | RSNA key management as defined in 12.7 (Keys and key distribution) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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) using PSK | 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 with SHA-256 or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | RSNA key management as defined in 12.7 (Keys and key distribution), PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) 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 SHA-256 | 3 (SAE) for SAE Authentication0 (open) for PMKSA caching |
| 00-0F-AC | 9 | FT authentication over SAE or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | FT key management defined in 12.7.1.6 (FT key hierarchy) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | Defined in 12.7.1.6.2 (Key derivation function (KDF)) using SHA-256 | 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 |
| 00-0F-AC | 10 | APPeerKey Authentication with SHA-256 or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | RSNA key management as defined in 12.7 (Keys and key distribution) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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 or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) using a Suite B compliant EAP method supporting SHA-256 | RSNA key management as defined in 12.7 (Keys and key distribution) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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 or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) using a Suite B compliant EAP method supporting SHA-384 | RSNA key management as defined in 12.7 (Keys and key distribution) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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 or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management)  | FT key management as defined in 12.7.1.6 (FT key hierarchy) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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(11ai) | 14 | Key management over FILS using SHA-256 and AES-SIV-256, PMKSA caching, or authentication negotiated over IEEE Std 802.1X(#114) | FILS key management defined in 12.12.2.5 (Key establishment with FILS authentication) | Defined in 12.12.2.5 (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(11ai) | 15 | Key management over FILS using SHA-384 and AES-SIV-512, PMKSA caching, or authentication negotiated over IEEE Std 802.1X(#114) | FILS key management defined in 12.12.2.5 (Key establishment with FILS authentication) | Defined in 12.12.2.5 (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(11ai) | 16 | FT authentication over FILS with SHA-256 and AES-SIV-256, PMKSA caching, or authentication negotiated over IEEE Std 802.1X(#114) | FT ~~authentication defined in 12.7.1.6.2 (Key derivation function (KDF))~~ key management as defined in 12.7.1.6 (FT key hierarchy) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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 |
| 00-0F-AC(11ai) | 17 | FT authentication over FILS with SHA-384 and AES-SIV-512, PMKSA caching, or authentication negotiated over IEEE Std 802.1X(#114) | FT ~~authentication defined in 12.7.1.6.2 (Key derivation function (KDF))~~ key management as defined in 12.7.1.6 (FT key hierarchy) or using PMKSA caching as defined in 12.6.10.3 (Cached PMKSAs and RSNA key management) | 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 |
| 00-0F-AC(#170) | 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(#171) | 20 | PSK | RSNA key management(Ed) as defined in 12.7 (Keys and key distribution) using PSK | Defined in 12.7.1.6.2 (Key derivation function (KDF)) using SHA-384. | 0 (open) |
| 00-0F-AC | (11ai)18, (#171)21–255  | Reserved | Reserved | Reserved | Reserved |
| Other OUI or CID | Any | Vendor-specific | Vendor-specific | Vendor-specific | Vendor-specific |

* PMKID

***Instruct the editor to modify as follows:***

~~The PMKID Count and List fields are used only in the RSNE in the (Re)Association Request frame to an AP and in FT authentication sequence frames.~~

The PMKID Count field indicates the number of PMKIDs that are contained in the PMKID List field. The PMKID List field contains a series (possibly empty) of PMKIDs.

When one or more PMKIDs are included in a (Re)Association Request frame or FILS Authentication frame to an AP, they identify PMKSAs that the STA believes to be valid for the destination AP. When a PMKID is included in a FILS Authentication frame to a STA, it identifies a PMKSA that the AP has selected.

~~The~~ A PMKID in the PMKID List field can refer to

* The PMKID of a~~A~~ cached PMKSA that has been obtained through preauthentication with the target AP
* The PMKID of a~~A~~ cached PMKSA from an EAP, FILS or SAE authentication
* The PMKID of a~~A~~ PMKSA derived from a PSK for the target AP
* The PMKR0Name of a~~A~~ PMK-R0 security association derived as part of an FT initial mobility domain association
* The PMKR1Name of a~~A~~ PMK-R1 security association derived as part of an FT initial mobility domain association or as part of a fast BSS transition.

See 12.7.1.3 (Pairwise key hierarchy) and 12.7.1.6.3 (PMK-R0) for the construction of the PMKID, 13.8 (FT authentication sequence) for the population of PMKID List for fast BSS transitions, 12.6.10.3 (Cached PMKSAs and RSNA key management) for the population of PMKID List when using PMKSA caching, 13.4 (FT initial mobility domain association) for the population of PMKID List for FT initial mobility domain association, 12.12.2 (FILS authentication protocol) for the population of PMKID List with FILS authentication, and 12.7.1.6 (FT key hierarchy) for the construction of PMKR0Name and PMKR1Name.

NOTE—A STA need not insert a PMKID in the PMKID List field if the STA will not be using that PMKSA.

* Requirements for support of MAC privacy enhancements(11aq)

***Instruct the editor to modify as follows:***

The non-AP STA connecting to an infrastructure BSS shall retain a single MAC address for the duration of its connection across an ESS. A PMKSA created as part of an RSNA will contain the MAC address of the AP used to create the PMKSA. Implementations of APs where dot11PMKSACachingMACRandomizationActivated is false or undefined might also bind a cached PMKSA to the non-AP STA’s MAC address. Therefore, a non-AP STA that supports PMKSA caching shall, if necessary, change its MAC address back to that value when attempting a subsequent association to the ESS using PMKSA caching, unless dot11PMKSACachingMACRandomizationActivated is true and the AP indicates support for PMKSA Caching with MAC Randomization in its RSN Extension element, in which case the non-AP STA may use a different MAC address.

* PMKSA

*Instruct the editor to modify as follows:*

When the PMKSA is the result of a successful IEEE 802.1X authentication, it is derived from the EAP authentication and authorization parameters provided by the AS. When the PMKSA is the result of a successful SAE authentication, it is generated as a result of the successful completion of the SAE exchange.

(M84)The PMKSA is created by the Supplicant’s SME when the EAP authentication or FILS authentication(11ai) completes successfully or the PSK is configured. The PMKSA is created by the Authenticator’s SME when the PMK is created from the keying information transferred from the AS in an(11ai) IEEE 802.1X authentication exchange, when the FILS authentication completes successfully(11ai), when the SAE exchange successfully completes, or when the PSK is configured. ~~When the negotiated AKM uses PMKID derivation with KCK as a parameter as defined in 12.7.1.3 (Pairwise key hierarchy), t~~The PMKID derived ~~from the KCK~~ during the initial 4-way handshake or FILS authentication exchange is not changed during the lifetime of this PMKSA.(M84)

NOTE – The PMKID does not change during the lifetime of the PMKSA even if the negotiated AKM uses PMKID derivation with one or more parameters that change during that lifetime (e.g. KCK, SPA). The keys derived when creating a PTKSA from a PMKSA use the current parameters at the time the PTKSA is created.

A PMKSA association is bidirectional. In other words, both parties use the information in the security association for both sending and receiving. The PMKSA is used to create the PTKSA. PMKSAs have a certain lifetime. The PMKSA consists of the following:

* PMKID, as defined in 12.7.1.3 (Pairwise key hierarchy) or 12.7.1.6.3 (PMK-R0). The PMKID identifies the security association.
* Authenticator’s MAC address; or, if the PMKSA was established by SAE authentication in an infrastructure BSS, the AP’s MAC address; or, if the PMKSA was established by SAE authentication in a PBSS or IBSS, the peer’s MAC address. For multi-band RSNA, the MAC address is associated with the operating band in use when the PMKSA is established.
* PMK; or, if the PMKSA was established with an AKM suite type for which the Authentication type column indicates FT authentication (see Table 9-151 (AKM suite selectors)), MPMK (see 12.7.1.6.3 (PMK-R0)).
* Lifetime, as defined in 12.7.1.3 (Pairwise key hierarchy) or 12.7.1.6 (FT key hierarchy).
* AKMP.
* All authorization parameters specified by the AS or local configuration. This might include parameters such as the STA’s authorized SSID.
* Cache Identifier, if advertised by the AP in FILS Indication element(11ai).
* Security association in an ESS

***Instruct the editor to modify as follows:***

When FT is not enabled, a STA roaming within an ESS establishes a new PMKSA by one of the (11ai)five

schemes:

— In the case of (re)association followed by IEEE 802.1X or PSK authentication, the STA repeats the same actions as for an initial contact association, but its Supplicant also deletes the PTKSA when it roams from the old AP. The Supplicant also deletes the PTKSA when it disassociates/deauthenticates from all BSSIDs in the ESS.

— In the case of SAE authentication followed by (re)association, the STA repeats the same actions as for initial contact association, but the non-AP STA also deletes the PTKSA when it roams from the old AP. Note that a STA can take advantage of the fact that it can perform SAE authentication to multiple APs while maintaining a single association with one AP, and then use any of the PMKSAs created during authentication to effect a fast BSS transition.

* A STA (AP) can cache PMKSAs for APs (STAs) in the ESS to which it has previously performed a full IEEE 802.1X authentication or SAE authentication. If a STA wishes to roam to an AP for which it has cached one or more PMKSAs, it can include one or more PMKIDs in the RSNE of its (Re)Association Request frame. An AP that has retained the PMK for one or more of the PMKIDs can proceed with the 4-way handshake (see 12.6.10.3 (Cached PMKSAs and RSNA key management). ~~The AP shall include the PMKID of the selected PMKSA in message 1 of the 4-way handshake. If none of the PMKIDs of the cached -PMKSAs matches any of the supplied PMKIDs, or if the AKM of the cached PMKSA differs from that offered in the (Re)Association Request, then the Authenticator, in the case of Open System authentication, shall perform another IEEE 802.1X authentication and, in the case of SAE authentication, shall transmit a Deauthentication frame to the STA. Similarly, i~~If the STA fails to send a PMKID when the negotiated AKM uses IEEE 802.1X authentication, the STA and AP need to perform a full IEEE 802.1X authentication.
* A STA already associated with the ESS can request its IEEE 802.1X Supplicant to authenticate with a new AP before associating to that new AP. The normal operation of the DS via the old AP provides the communication between the STA and the new AP. The SME delays reassociation with the new AP until IEEE 802.1X authentication completes via the DS. If IEEE 802.1X authentication completes successfully, then PMKSAs shared between the new AP and the STA are cached, thereby enabling the possible usage of reassociation without requiring a subsequent full IEEE 802.1X authentication procedure.
* In the case of FILS, the STA may repeat the same actions as an initial contact and authentication~~,~~. The STA may also use a cached PMKSA to authenticate (see 12.6.10.3 (Cached PMKSAs and RSNA key management) and 12.12.2 (FILS authentication protocol)). A STA already associated with the ESS can initiate FILS authentication to multiple other APs while associated.
* Cached PMKSAs and RSNA key management

***Instruct the editor to modify as follows:***

~~In a non-FT environment, a~~ A STA might cache PMKSAs it establishes as a result of previous authentication. The PMKSA shall not be changed while cached. The PMK(11ai) in the PMKSA is used with the 4-way handshake or FILS authentication(11ai) to establish fresh PTKs.

If a STA in an infrastructure BSS has determined it has a valid PMKSA with an AP to which it is about to (re)associate, it performs Open System authentication or FILS Authentication to the AP, When Open System authentication is used, ~~and then~~ it includes the PMKID for the PMKSA in the RSNE in the (Re)Association Request. When FILS Authentication is used, it includes the PMKID for the PMKSA in the FILS Authentication frame (see 12.12.2 (FILS authentication protocol)).. When the PMKSA was not created using pre-authentication, the AKM indicated in the RSNE by the STA in the (Re)Association Request shall be identical to the AKM used to establish the cached PMKSA in the first place.

A cached PMKSA established using an AKM suite type for which the Authentication type column indicates FT authentication (see Table 9-151 (AKM suite selectors)) may be used in a subsequent FT Initial Mobility Domain Association using an identical AKM suite. A cached PMKSA is not used in an FT authentication sequence.

If a cached PMKSA is used in FT Initial Mobility Domain Association, the cached MPMK is used to derive the PMK-R0 of a new FT key hierarchy (see 12.7.1.6 (FT key hierarchy)). The PMKID indicated by the STA in the (Re)Association Request frame and message 1 of the FT 4-way handshake (when FILS authentication is not used) or FILS Authentication frame (when FILS authentication is used) is the PMKID of the cached PMKSA as defined in 12.7.1.6.3 (PMK-R0) (i.e. not the PMKR0Name or PMKR1Name of the FT key hierarchy that was derived when the PMKSA was originally established). The PMKR1Name indicated in RSNE in messages 2 and 3 of the FT 4-way handshake (when FILS authentication is not used) or in (Re)Association Request and Response frames (when FILS authentication is used) is the PMKR1Name of the newly derived FT key hierarchy (see 13.4 (FT initial mobility domain association).

Upon receipt of a (Re)Association Request frame following Open System authentication with one or more PMKIDs, an AP checks whether its Authenticator has cached a PMKSA for the PMKIDs and whether the AKM in the cached PMKSA matches the AKM in the (Re)Association Request; and if so, it shall assert possession of that PMKSA by beginning the 4-way handshake after association has completed and shall include the PMKID in message 1 (see 12.7.6.2 4-way handshake message 1). If the Authenticator does not have a PMKSA for the PMKIDs in the (Re)Association Request or the AKM does not match, its behavior depends on how the PMKSA was established. If SAE authentication was used to establish the PMKSA, then the AP shall reject (re)association by sending a (Re)Association Response frame with status code STATUS\_INVALID\_PMKID. Note that this allows the non-AP STA to fall back to full SAE authentication to establish another PMKSA. If IEEE 802.1X authentication was used to establish the PMKSA, the AP begins a full IEEE 802.1X authentication after association has completed.

Upon receipt of a FILS Authentication frame with one or more PMKIDs, an AP checks whether its Authenticator has cached a PMKSA for the PMKIDs and whether the AKM in the cached PMKSA matches the AKM in the FILS Authentication frame, and whether the PMK is still valid; and if so, it shall assert possession of that PMK by including the PMKID in the FILS Authentication frame sent in response (see 12.12.2.3.4 AP construction of Authentication frame). If the Authenticator does not have a PMK for the PMKIDs in the FILS Authentication frame or the AKM does not match, the AP may either reply with EAP-Finish/Re-auth to continue FILS Shared Key authentication option if the non-AP STA included sufficient information for that, or the AP rejects the authentication(11ai).

If both sides assert possession of a cached PMKSA, but the 4-way handshake or FILS authentication(11ai) fails, both sides may delete the cached PMKSA for the selected PMKID.

If the lifetime of a cached PMKSA expires, the STA shall delete the expired PMKSA.

If a STA roams to an AP with which it is preauthenticating and the STA does not have a PMKSA for that AP, the STA needs to initiate a full IEEE 802.1X EAP authentication.

If an AP with dot11PMKSACachingMACRandomizationActivated true has cached a PMKSA, it shall assert possession of the PMKSA on reception of a (Re)Association Request frame or FILS Authentication frame from a non-AP STA indicating the matching PMKID and AKM as described above, irrespective of the MAC address used by that non-AP STA.

NOTE – An AP with dot11PMKSACachingMACRandomizationActivated false or undefined might not assert possession of a cached PMKSA if the non-AP STA that indicates the PMKID is using a different MAC address compared to when the PMKSA was established, or the AP might never cache PMKSAs at all.

* RSNA rekeying

***Instruct the editor to modify as follows:***

When a PTKSA is deleted, a non-AP and non-PCP STA may reassociate with the same AP or PCP and/or establish a new RSNA with the AP or PCP. If the non-AP and non-PCP STA has cached one or more PMKSAs, it may skip the PMKSA establishment and proceed with the creation of a new PTKSA by using 4-way handshake, FT 4-way handshake or FILS authentication(11ai) using the procedures defined in 12.6.10.3 (Cached PMKSAs and RSNA key management). When a GTKSA is deleted, an originating STA may create a new GTKSA by using 4-way handshake or group key handshake.(#59)

* Keys and key distribution
* Key hierarchy
* General

***Instruct the editor to modify as follows:***

RSNA defines the following key hierarchies:

* Pairwise key hierarchy, to protect individually addressed traffic
* GTK, a hierarchy consisting of a single key to protect group addressed traffic

NOTE—Pairwise key support with enhanced data cryptographic encapsulation mechanisms allows a receiving STA to detect MAC address spoofing and data forgery. The RSNA architecture binds the transmit and receive addresses to the pairwise key. If an attacker creates an MPDU with the spoofed TA, then the decapsulation procedure at the receiver generates an error. GTKs do not have this property.

* Integrity GTK (IGTK), a hierarchy consisting of a single key to provide integrity protection for group addressed robust Management frames

 d) BIGTK, a hierarchy consisting of a single key to provide for integrity protection for Beacon frames

e) FT key hierarchy, to protect individually addressed traffic in an FT environment

* Pairwise key hierarchy

***Instruct the editor to modify as follows:***

Except when preauthentication or FILS authentication(11ai) 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, 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. 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.

<snip>

When ~~not~~ using ~~a PSK~~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 12.12.2.5.2 (PMKSA establishment with FILS authentication), respectively.

<snip>

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 Length = KCK\_bits + KEK\_bits + TK\_bits. The values of KCK\_bits and KEK\_bits are AKM suite dependent and are listed in Table 12-8 (Integrity and key-wrap algorithms(#102)(#1188)). The value of TK\_bits is cipher-suite dependent and is defined in Table 12-5 (Cipher suite key lengths). 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).

NOTE 4(11ai)—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(11ai) to derive a new PTK. Both the Authenticator and Supplicant create a new nonce value for each 4-way handshake or FILS authentication(11ai) instance.

When using FILS authentication, the PTK is derived as defined in 12.12.2.5.3 (PKTSA key derivation with FILS authentication).

<snip>

When the negotiated AKM is 00-0F-AC:5~~,~~ or 00-0F-AC:6, ~~00-0F-AC:14, or 00-0F-AC:16~~, the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-256(PMK, "PMK Name" || AA || SPA))

When the negotiated AKM is 00-0F-AC:11, the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-256(KCK, "PMK Name" || AA || SPA))

When the negotiated AKM is 00-0F-AC:12, and the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-384(KCK, "PMK Name" || AA || SPA))

When the negotiated AKM is 00-0F-AC:14:

* When IEEE 802.1X authentication is used, the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-256(PMK, "PMK Name" || AA || SPA))

* When FILS authentication is used, the PMK identifier is derived as defined in 12.12.2.5 (Key establishment with FILS authentication).

When the negotiated AKM is 00-0F-AC:15:

* When IEEE 802.1X authentication is used, the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-384(PMK, "PMK Name" || AA || SPA))

* When FILS authentication is used, the PMK identifier is derived as defined in 12.12.2.5 (Key establishment with FILS authentication).

When the negotiated AKM is ~~00-0F-AC:13, 00-0F-AC:15,~~ ~~00-0F-AC:17 or~~ 00-0F-AC:20, ~~and~~ the PMK identifier is defined as

PMKID = Truncate-128(HMAC-SHA-384(PMK, "PMK Name" || AA || SPA))

When the negotiated AKM is 00-0F-AC:8, the PMK identifier is derived as defined in 12.4.5.4 (Processing of a peer’s SAE Commit message).

When the negotiated AKM is is a suite type for which the Authentication type column indicates FT authentication (see Table 9-151 (AKM suite selectors)) ~~00-0F-AC:3, 00-0F-AC:4, 00-0F-AC:9, or 00-0F-AC:19~~, the PMKID (used for PMKSA caching in FT Initial Mobility Domain Association, see 12.6.10.3 (Cached PMKSAs and RSNA key management)) and ~~the~~ PMKR0Name ~~is~~ are derived as defined in 12.7.1.6.3 (PMK-R0) and PMKR1Name is derived as defined in 12.7.1.6.4 (PMK-R1).

Otherwise, the PMK identifier is defined as

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

In all these cases, “PMK Name” is treated as an ASCII string.

When the PMKID is calculated for the PMKSA as part of preauthentication, the AKM has not yet been negotiated. In this case, the HMAC-SHA-1 based derivation is used for the PMKID calculation.

* FT key hierarchy
* Overview

***Instruct the editor to modify as follows:***

This subclause describes the FT key hierarchy and its supporting architecture. The FT key hierarchy is designed to allow a STA to make fast BSS transitions between APs without the need to perform an SAE or IEEE 802.1X authentication at every AP within the mobility domain.

The FT key hierarchy can be used with SAE, IEEE 802.1X authentication, (11ai)PSK authentication, or FILS authentication(11ai).

A three-level key hierarchy provides key separation between the key holders. The FT key hierarchy for the Authenticator is shown in Figure 12-32 (FT key hierarchy at an Authenticator(11ai)(#114)). An identical key hierarchy exists for the Supplicant, and identical functions are performed by the corresponding S0KH and S1KH.

The FT key hierarchy shown in Figure 12-32 (FT key hierarchy at an Authenticator(11ai)(#114)) consists of three levels whose keys are derived using the key derivation function (KDF) described in 12.7.1.6.2 (Key derivation function (KDF)) as follows:

* PMK-R0 – the first-level key of the FT key hierarchy. This key is derived as a function of the master session key (MSK), PMK or PSK. It is stored by the PMK-R0 key holders, R0KH and S0KH.
* PMK-R1 – the second-level key of the FT key hierarchy. This key is mutually derived by the S0KH and R0KH.
* PTK – the third-level key of the FT key hierarchy that defines the IEEE 802.11 and IEEE 802.1X protection keys. The PTK is mutually derived by the PMK-R1 key holders, R1KH and S1KH.

As shown in Figure 12-32 (FT key hierarchy at an Authenticator), ~~the R0KH computes the PMK-R0~~ the Master PMK (MPMK) is obtained from the PMK ~~key~~ obtained from SAE authentication ~~(for the purposes of FT this key is identified as the Master PMK, or MPMK)~~, from the PSK, or from the MSK resulting (per IETF RFC 3748 [B44]) from a successful IEEE 802.1X authentication between the AS and the Supplicant, or from the PMK (see 12.12.2.5.2 (PMKSA key derivation with FILS authentication)) resulting from a successful FILS authentication. The R0KH computes the PMK-R0 from the MPMK.

NOTE -- The MPMK is a constituent of a PMKSA that may be cached (see 12.6.1.1.2 (PMKSA) and 12.6.10.3 (Cached PMKSAs and RSNA key management)).

Upon a successful authentication, the R0KH shall delete any prior PMK-R0 security association for this mobility domain pertaining to this S0KH. The R0KH shall also delete all PMK-R1 security associations derived from that prior PMK-R0 security association. The PMK-R1s are generated by the R0KH and are assumed to be delivered from the R0KH to the R1KHs within the same mobility domain. The PMK-R1s are used for PTK generation. Upon receiving a new PMK-R1 for an S0KH, an R1KH deletes the prior PMK-R1 security association and PTKSAs derived from the prior PMK-R1.

***Instruct the editor to replace Figure 12-32 with the following:***



***Instruct the editor to modify as follows:***

It is assumed by this standard that the PSK is specific to a single S0KH and a single R0KH.

The lifetime of the MPMK, PMK-R0, PMK-R1, and PTK are bound to the lifetime of the ~~M~~PMK, PSK, or MSK from which it was derived. For example, the AS may communicate the MSK lifetime with the MSK. If such an attribute is provided, the lifetime of the MPMK ~~PMK-R0~~ shall be not more than the lifetime of the MSK. The lifetime of the PTK ~~and~~ , PMK-R1 and PMK-R0 is the same as that of the MPMK~~PMK-R0~~. When the key lifetime expires, each key holder shall delete its respective MPMK, PMK-R0, PMK-R1 or PTKSA.

***Instruct the editor to move Section 12.7.1.6.2 (Key derivation function (KDF)) to between 12.7.1.2 and 12.7.1.3, since the KDF is used in :***

***Instruct the editor to modify as follows:***

* PMK-R0

The first-level key in the FT key hierarchy, PMK-R0, is derived using the KDF defined in 12.7.1.6.2 (Key derivation function (KDF)). The PMK-R0 is the first level keying material used to derive the next level keys (PMK-R1s).

The PMK-R0 is derived from the MPMK, which is obtained when a PMKSA is established using a negotiated AKM for which the Authentication type column indicates FT authentication (see Table 9-151 (AKM suite selectors)):

If the negotiated AKM is 00-0F-AC:3, then Q = 256 and:

* MPMK = L(MSK, 256, 256), i.e. the second 256 bits of the MSK (which is derived from the IEEE 802.1X authentication)
* PMKID = Truncate-128(HMAC-SHA-256(MPMK, "PMK Name" || AA || SPA))

If the negotiated AKM is 00-0F-AC:4, then Q = 256 and:

* MPMK = PSK
* PMKID = Truncate-128(HMAC-SHA-256(MPMK, "PMK Name" || AA || SPA))

If the negotiated AKM is 00-0F-AC:9, then Q = 256 and:

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

If the negotiated AKM is 00-0F-AC:13, then Q = 384 and:

* MPMK = L(MSK, 0, 384), i.e. the first 384 bits of the MSK (which is derived from the IEEE 802.1X authentication)
* PMKID = Truncate-128(HMAC-SHA-384(MPMK, "PMK Name" || AA || SPA))

If the negotiated AKM is 00-0F-AC:16, then Q = 256 and:

* When FILS authentication is used:
	+ MPMK = PMK generated as the result of FILS authentication per 12.12.2.5.2 (PMKSA key derivation with FILS authentication)
	+ PMKID is as defined in 12.12.2.5.2 (PMKSA key derivation with FILS authentication)
* When IEEE 802.1X authentication is used, then:
	+ MPMK = L(MSK, 256, 256), i.e. the second 256 bits of the MSK (which is derived from the IEEE 802.1X authentication)
	+ PMKID = Truncate-128(HMAC-SHA-256(MPMK, "PMK Name" || AA || SPA))

If the negotiated AKM is 00-0F-AC:17, then Q = 384 and:

* When FILS authentication is used:
	+ MPMK = the PMK generated as the result of FILS authentication per 12.12.2.5.2 (PMKSA key derivation with FILS authentication)
	+ PMKID is as defined in 12.12.2.5.2 (PMKSA key derivation with FILS authentication)
* When IEEE 802.1X authentication is used, then:
	+ MPMK = L(MSK, 0, 384), i.e. the first 384 bits of the MSK (which is derived from the IEEE 802.1X authentication)
	+ PMKID = Truncate-128(HMAC-SHA-384(MPMK, "PMK Name" || AA || SPA))

If the negotiated AKM is 00-0F-AC:19, then Q = 384 and:

* MPMK = PSK
* PMKID = Truncate-128(HMAC-SHA-384(MPMK, "PMK Name" || AA || SPA))

In all these cases, “PMK Name” is treated as an ASCII string.

The PMK-R0 is then derived as follows:

R0-Key-Data = KDF-Hash-Length(XXKey, "FT-R0", SSIDlength || SSID || MDID || R0KHlength || R0KH-ID || S0KH-ID)

PMK-R0 = L(R0-Key-Data, 0, Q)

PMK-R0Name-Salt = L(R0-Key-Data, Q, 128)

Length = Q + 128

where

* KDF-Hash-Length is the (#246)key derivation function as defined in 12.7.1.6.2 (Key derivation function (KDF)) using the hash algorithm identified by the AKM suite selector (see Table 9-151 (AKM suite selectors)).
* ~~If the negotiated AKM(#1389) is 00-0F-AC:3, then Q shall be 256, and XXKey shall be the second 256 bits of the MSK (which is derived from the IEEE 802.1X authentication), i.e., XXKey = L(MSK, 256, 256). If the negotiated AKM(#1389) is 00-0F-AC:4, then Q shall be 256, and XXKey shall be the PSK. If the negotiated AKM(#1389) is 00-0F-AC:9, then Q shall be 256, and XXKey shall be the MPMK generated as the result of SAE authentication. If the negotiated AKM(#1389) is 00-0F-AC:13, then Q shall be 384, and XXKey shall be the first 384 bits of the MSK (which is derived from the IEEE 802.1X authentication), i.e., XXKey = L(MSK, 0, 384). If the negotiated AKM(#1389) is 00-0F-AC:16, then Q shall be 256, and XXKey shall be the FILS-FT described in 12.12.2.5.3 (PTKSA Key derivation with FILS authentication).~~ If the negotiated AKM is 00-0F-AC:16 or 00-0F-AC:17 and FILS authentication is used, then ~~Q shall be 384, and~~ XXKey shall be the FILS-FT derived from MPMK as described in 12.12.2.5.3 (PTKSA Key derivation with FILS authentication)(11ai). Otherwise, XXKey is equal to MPMK.
* EAPOL-Key frames

***Instruct the editor to modify as follows:***

The following EAPOL-Key frames are used to implement the three different exchanges:

* **4-way handshake message 1** is an EAPOL-Key frame with the Key Type subfield equal to 1. Use of the ~~The~~ Key Data field to indicate ~~shall contain~~ a PMKID when a cached ~~for the~~ PMKSA ~~that~~ is being used in this key derivation is defined in 12.6.10.3 (Cached PMKSAs and RSNA key management). When a cached PMKSA is not being used, inclusion of the PMKID (if derived) is optional. The Key Data field ~~and~~ need not be encrypted.
* PTKSA Key derivation with FILS authentication

***Instruct the editor to modify as follows:***

When doing FT initial mobility domain association using FILS authentication,

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

where

ICK\_bits is the length of ICK in bits

KEK\_bits is the length of KEK in bits

FILS-FT\_bits is the length of FILS-FT in bits when doing FT initial mobility domain association using FILS authentication

X is 512+TK\_bits, 768+TK bits, 896+TK bits, or 1280+TK bits from Table 12-5 (Cipher suite key lengths) depending on the negotiated AKM(#1389)

PMK is the PMK from the PMKSA, either created from an initial FILS connection or from a cached PMKSA, when PMKSA caching is used; when doing FT initial mobility domain association using FILS authentication, it is equal to the MPMK (see 12.7.1.6.3 (PMK-R0))

***Instruct the editor to modify C.3 as follows:***

**C.3 MIB detail**

Dot11StationConfigEntry::=

 SEQUENCE {

......

dot11PMKSACachingMACRandomizationActivated TruthValue }

dot11PMKSACachingMACRandomizationActivated OBJECT-TYPE

 SYNTAX TruthValue

 MAX-ACCESS read-write

 STATUS current

 DESCRIPTION

"This is a control variable.

It is written by the MAC or an external management entity.

Changes take effect as soon as practical in the implementation.

This variable indicates whether support for PMKSA Caching with MAC randomization is supported by the STA.

.

 ::= { dot11StationConfigEntry <ANA>}

DEFVAL { false }