IEEE P802.11
Wireless LANs

|  |
| --- |
| SA Ballot on D5.0 – CID 22342 |
| Date: 2023-05-10 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Thomas Derham |  |  |  | thomas.derham@broadcom.com |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CID** | **Clause** | **Page** | **Comment** | **Proposed Change** | **Resolution** |
| 22342 | 12 | 446 | While various subclauses (e.g. 11.3.5.2) indicate FILS Authentication is defined between MLDs, usage of STA-MAC and BSSID in FILS key derivations has not been updated for ML case. | In all locations where STA-MAC and/or BSSID are used in FILS authentication in baseline (e.g. multiple locations for hash and signature derivation in 12.11.2.6.2), add text that specifies how it is performed for ML case - presumably using MLD addresses for consistency with other security modes such as SAE. | Revised – agree with the commenter. Apply the changes in this document. |

**Discussion**

For consistency with all other cases where authentication is performed between MLDs, the MLD addresses should be used in FILS authentication key derivations too.

In addition, since FILS authentication is currently described as being performed between an AP and STA, various text also needs updating to allow it to be used between MLDs, including procedural descriptions, requirements for delivery of per-link group keys, and so on. It seems reasonable to reuse the approach in 11be subclause 13 (FT) where Originator (non-AP STA or non-AP MLD) and Responder (AP or AP MLD) roles are defined and used in the context of a FILS authentication exchange. The original text below (12.11) is copied from REVme D3.0.

**Proposed text**

Tgbe editor: Add the following subclause 12.11 as follows:

* Authentication for FILS
* General

FILS authentication is an RSNA authentication protocol. The FILS authentication protocol authenticates STAs or MLDs to each other, using either a shared key or a public key.

When FILS authentication is performed between STAs, the non-AP STA is known as the FILS Originator (FILSO) and the AP is known as the FILS Responder (FILSR).

When FILS authentication is performed between MLDs, the non-AP MLD is known as the FILS Originator (FILSO) and the AP MLD is known as the FILS Responder (FILSR).

When FILS Shared Key authentication is used, the authentication exchange can optionally be performed with PFS. When FILS Public Key authentication is used, PFS is always used. When the FILS authentication protocol is performed with PFS, the FILSO and FILSR derive ephemeral public and private keys with respect to a particular set of domain parameters that define a finite cyclic group and then exchange public keys. The result of the FILS authentication includes a PTKSA.

The security of FILS authentication depends on the following assumptions:

* When FILS Shared Key authentication is used, the FILSO and FILSR share either a valid rRK as defined in IETF RFC 6696 with a TTP that is capable of being used with EAP-RP, or a PMK cached from a previous authenticated connection.
* When FILS Public Key authentication is used, the FILSO and FILSR have a means to trust the public key of the FILSR and FILSO, respectively.
* When PFS is used, a finite cyclic group is negotiated where solving the discrete logarithm problem is computationally infeasible.
* When PFS is used, both the FILSO and FILSR have in common at least one finite cyclic group from the dot11RSNAConfigDLCGroupTable.
* FILS authentication protocol
* General

The FILSO and FILSR perform key establishment using Authentication frames and perform key confirmation using (Re)Association Request and (Re)Association Response frames.

After exchanging Authentication frames, the FILSO and FILSR derive a shared and secret key that will be used to derive a set of secret keys (as defined in 12.11.2.5.2 (PMKSA key derivation with FILS authentication)) that are authenticated after exchanging (Re)Association Request and (Re)Association Response frames.

When a shared key is used for FILS authentication, and if the FILSO shares a valid rRK with the TTP, then EAP-RP as defined in IETF RFC 5295 and IETF RFC 6696 shall be used.

To prevent key reinstallation attacks, the non-AP STA shall maintain a copy of the most recent GTK, most recent IGTK(11ba), most recent BIGTK, and most recent WIGTK installed as part of the FILS authentication protocol as if they were installed as a result of receipt of EAPOL-Key frames (see 12.7.7.4 (Group key handshake implementation considerations)) and shall refuse to update a GTK, IGTK(11ba), BIGTK, or WIGTK when the key to be set matches any one of these keys (see 6.5.14 (SetKeys)).

* Discovery of a FILS AP

An AP indicates that it is capable of performing FILS authentication by indicating support for a FILS AKM in an RSNE(#1517) and including it, and the FILS Indication element, in Beacon and Probe Response frames. All APs affiliated with an AP MLD shall advertise the same FILS Indication element, and at least one FILS AKMs shall be common across all those APs.

An AP may indicate that it is capable of performing FILS authentication by indicating support for a FILS AKM in the FD RSN Information subfield in a FILS Discovery frame.

An AP indicates support for FILS Shared Key authentication without PFS by setting the FILS Shared Key authentication without PFS supported bit to 1 in the FILS Information field of the FILS Indication element. An AP indicates support for FILS Shared Key authentication with PFS by setting the FILS Shared Key authentication with PFS Supported bit to 1 in the FILS Information field of the FILS Indication element. An AP may advertise between zero and seven realms using the Realm Identifier subfield(s) of the FILS Indication element that is part of Beacon, Probe Response, and FILS Discovery frames. If the STA or non-AP MLD believes it shares a valid rRK as defined in IETF RFC 6696 with the AP or AP MLD through, e.g., a hashed domain name that matches an AP-advertised realm, a HESSID, or other ANQP information, the STA or non-AP MLD may begin FILS Shared Key authentication with the AP or AP MLD using EAP-RP. Domain name hashing is specified in 11.45.4 (FILS authentication and higher layer setup capability indications). If a STA or non-AP MLD discovers a FILS AP or AP MLD and the STA or non-AP MLD believes it shares a PMKSA with the AP or AP MLD, it may begin the FILS authentication protocol with the AP or AP MLD using PMKSA caching.

An AP indicates support for FILS Public Key authentication by setting the FILS Public Key authentication Supported bit to 1 in the FILS Information field of the FILS Indication element. An AP may advertise up to seven public key indicators in the FILS Indication element that is part of Beacon, Probe Response, and FILS Discovery frames. If the STA or non-AP MLD discovers that it trusts the issuer of an AP’s X.509v3 certificate, or that it trusts its uncertified public key identified by matching its hash, the STA or non-AP MLD may begin the FILS authentication protocol to the AP or AP MLD and perform mutual authentication using trusted public keys.

* Key establishment with FILS Shared Key authentication
* Overview

This subclause defines the procedure for establishing a shared key between a FILSO and FILSR using FILS Shared Key authentication that uses shared symmetric keys between the FILSO and the TTP.

A STA or non-AP MLD may initiate FILS Shared Key authentication either with a FILS AP or AP MLD that is connected to a TTP that shares a valid key, called an rRK, as defined in IETF RFC 6696 with the STA or non-AP MLD, or with a FILS AP or AP MLD with whom it shares a cached PMKSA. If neither of these cases applies, a full EAP exchange may be performed via IEEE 802.1X authentication to establish rRK as defined in IETF RFC 6696 or another form of FILS authentication may be used to establish a shared PMKSA. When performing a full EAP exchange using RSNA to establish rRK, the Authentication algorithm number 0 (Open System) is used.

EAP-RP signaling as defined in IETF RFC 5295 and IETF RFC 6696 is used to validate the mutual possession of rRK between the FILSO and the TTP. EAP-RP signaling is encapsulated using a FILS Wrapped Data element in the Authentication frame. The FILSR unwraps the encapsulated EAP-RP packet received from the FILSO in the FILS Wrapped Data element and forwards the EAP-RP packet to the TTP using a transport that is out of scope of this standard. When the FILSR receives an EAP-RP packet from the TTP, the FILSR forwards the packet to the FILSO by encapsulating the EAP-RP packet in the FILS Wrapped Data element of the Authentication frame.

The message sequence is depicted in Figure 12-55 (FILS Shared Key authentication(#3596)).

The following subclauses are organized per each step as shown in Figure 12-55 (FILS Shared Key authentication(#3596)).

* FILSO construction of Authentication frame

If the STA or non-AP MLD (i.e. the FILSO) chooses to initiate FILS Shared Key authentication, it shall first choose a random 16-octet nonce and then determine whether to attempt PMKSA caching. If PMKSA caching is attempted, it shall generate a list of PMKSA identifiers. If the FILSO attempts to initiate EAP-RP, it shall construct an EAP-Initiate/Reauth packet per IETF RFC 6696, with the following clarifications:

* Regarding EAP-RP Flags:
* The B flag shall be set to 0, indicating that this is not an EAP-RP bootstrap message.
* The L flag shall be set to 1, indicating that the TTP with whom the FILSO shares the rRK is to provide the lifetimes of rRK and rMSK in (#330)the EAP-Finish/Reauth packet.
* EAP Identifier is set to 0.
* The Cryptosuite field shall not be set to 1.

If PFS is desired, the FILSO selects a finite cyclic group from the dot11RSNAConfigDLCGroupTable, which comprises identifying number from a repository maintained by IANA as “Group Description” attributes for IETF RFC 2409 (IKE) [B14][B29]. The FILSO then generates an ephemeral private key, and performs the group’s scalar-op (see 12.4.4.1 (General)) with its random ephemeral private key and the generator from the selected finite cyclic group to compute an ephemeral public key.

The FILSO then constructs an Authentication frame with the Authentication algorithm number set to 4 (FILS Shared Key authentication without PFS) or 5 (FILS Shared Key authentication with PFS) (see 9.4.1.1 (Authentication Algorithm Number field)) depending on whether PFS is used, and the Authentication transaction sequence number set to 1. The random nonce shall be encoded in the FILS Nonce element (see 9.4.2.188 (FILS Nonce element)). If a list of PMKSA identifiers was generated, it shall be used to construct the PMKID List field in RSNE. The random FILS Session shall be encoded in the FILS Session element (see 9.4.2.178 (FILS Session element)). The EAP-Initiate/Reauth packet, if generated, shall be copied into the FILS Wrapped Data field (see 9.4.2.186 (FILS Wrapped Data element)). If PFS is desired, the chosen finite cyclic group shall be encoded in the Finite Cyclic Group field (see 9.4.1.40 (Finite Cyclic Group field)) and the ephemeral public key shall be encoded in the FFE field (see 9.4.1.38 (FFE field)) according to the element to (#1288)octet string conversion in 12.4.7.2.4 (Element to octet string conversion).

The FILSO transmits the Authentication frame to the FILSR.

* FILSR processing of Authentication frame

Upon reception of the Authentication frame with the Authentication algorithm number equal to 4 or 5, the FILSR shall perform the following procedure:

* If Authentication frame includes a Finite Cyclic Group field, then the FILSR shall first determine whether the indicated finite cyclic group in the received FILS Authentication frame is supported.
* If the indicated finite cyclic group in the received FILS Authentication frame is not supported, the FILSR shall respond with an Authentication frame with the Authentication algorithm number set to 5 (FILS Shared Key authentication with PFS) (see 9.4.1.1 (Authentication Algorithm Number field)) and the Status Code field set to 77 (Authentication is rejected because the offered finite cyclic group is not supported) and shall terminate the exchange.
* If PFS is being used, the FILSO’s public key shall be converted from an octet string to an element according to the conversion in 12.4.7.2.5 (Octet string to element conversion). Then the FILSR shall validate the FILSO’s public key in a group-specific fashion as described in 5.6.2.3 of (#298)NIST Special Publication 800-56A R2. If validation fails, the FILSR shall terminate the exchange.
* The FILSR shall check whether PMKSA caching is being attempted by the presence of the PMKID List field in RSNE:
* If the PMKID List field is present in RSNE, the FILSR checks whether any PMKSA identifier offered in the PMKID List matches an identifier for a cached PMKSA. If so, the FILSR selects a PMKID that matches and continues the FILS Shared Key authentication protocol using the PMK from the identified PMKSA.
* If a PMKID List field is not present in RSNE or if no PMKSA identifier offered in the PMKID list matches any identifier for a cached PMKSA, the FILSR checks whether an EAP‑Initiate/Reauth packet was included. If not, the FILSR shall respond with an Authentication frame with the Authentication algorithm number set to 4 or 5 depending on whether PFS is used and the Status Code field set to 53 (invalid PMKID) and shall terminate the exchange.
* If an EAP-Initiate/Reauth packet is included and PMKSA caching is not used, the FILSR shall extract the (#2043)data needed from the FILS Wrapped Data field (see 9.4.2.186 (FILS Wrapped Data element)) and shall forward it to the Authentication Server. When applicable, the FILSR communicates with the Authentication Server using the same protocols it uses when authenticating with EAP. Suitable protocols include, but are not limited to, remote authentication dial-in user service RADIUS (as specified in IETF RFC 2865 [B33]) and Diameter (as specified in IETF RFC 6942).

If PFS is being used, the FILSR shall also generate an ephemeral private key and perform the group’s scalar-op (see 12.4.4.1 (General)) to produce its own ephemeral public key. The FILSR may delay the generation of its ephemeral public/private key pair until after receiving a response from the Authentication Server, if applicable. The Authentication Server processes the EAP-Initiate/Reauth packet as specified in IETF RFC 6696 and returns an EAP-Finish/Reauth packet to the FILSR. In the case of successful authentication by the Authentication Server, the Authentication Server returns the associated EAP-RP rMSK with the EAP-Finish/Reauth packet. If the Authentication Server responds with a failure indication, then the FILSR shall produce an Authentication frame with the Authentication Algorithm Number field set to 4 (FILS Shared Key authentication without PFS) or 5 (FILS Shared Key authentication with PFS) (see 9.4.1.1 (Authentication Algorithm Number field)), and the Status Code field set to 15 (Authentication rejected because of challenge failure). In the case of successful authentication by the Authentication Server, the Authentication Server returns the associated EAP-RP rMSK with the EAP-Finish/Reauth packet and processing terminates.

The FILSR proceeds by constructing an Authentication frame.

* FILSR construction of Authentication frame

If PMKSA caching is not used and the FILSR is not connected to, or does not recognize the Authentication Server identified by the FILSO using the realm in the key Name-NAI field of the EAP-Initiate/Reauth packet, then the FILSR shall send Authentication frame with Status Code field set to 113, “Authentication rejected due to unknown Authentication Server” to the FILSO.

Otherwise, the FILSR shall generate its own nonce and construct an Authentication frame for the FILSO. The FILSR shall copy the FILS Session element from the Authentication frame sent by the FILSO to this response Authentication frame. If PMKSA caching is not used, this frame shall contain the FILS wrapped data that encapsulates (#330)the EAP-Finish/Reauth packet received from the Authentication Server. In addition, if PFS is used, the FFE field of the Authentication frame sent by the FILSR contains the FILSR’s ephemeral public key. In this frame, the FILSR shall set the Authentication algorithm number to 4 or 5 depending on whether PFS is used, and the Authentication sequence number to 2. If PMKSA caching is used, the FILSR indicates the selected PMKID in the PMKID List.

If PFS is being used for the exchange, the FILSR shall perform the group’s scalar-op (see 12.4.4.1 (General)) with the FILSO’s ephemeral public key and its own ephemeral private key to produce an ephemeral Diffie-Hellman shared secret, DHss.

The FILSR transmits the Authentication frame to the FILSO. Upon transmission of the FILS Authentication frame, the FILSR proceeds to key establishment per 12.11.2.5 (Key establishment with FILS authentication).

* FILSO processing of Authentication frame

The FILSO processes the received Authentication frame as follows:

* The FILSO shall abandon FILS authentication if any of the following conditions occur:
* The received Authentication frame does not include the Authentication Algorithm Number equal to 4 (FILS Shared Key authentication without PFS) or 5 (FILS Shared Key authentication with PFS) (see 9.4.1.1 (Authentication Algorithm Number field)).
* PMKSA caching was attempted and the received Authentication frame includes a PMKID that does not match a PMKID in the Authentication frame sent by the FILSO.
* The received Authentication frame does not include either a PMKID or an EAP-Finish/Reauth packet.
* The received Authentication frame does not include the FILS Session element.
* The received FILS Session value does not match the one in the Authentication frame sent by the FILSO.
* If the received Authentication frame includes the Status Code field equal to 15 (Authentication rejected because of challenge failure) or 53 (invalid PMKID), then the FILSO shall abandon the FILS authentication.
* The FILSO verifies that the FILSR transmitted PFS parameters are consistent with the FILSO’s previous transmissions (indicated by whether or not the FILSO transmitted an ephemeral public key):
* If the FILSO transmitted an ephemeral public key, and the received Authentication frame does not include an ephemeral public key, then the FILSO shall abandon the FILS authentication.
* If the FILSO did not transmit an ephemeral public key, and the received Authentication frame includes an ephemeral public key, then the FILSO shall abandon the FILS authentication.
* If applicable, the FILSO processes the EAP-Finish/Reauth packet as per IETF RFC 6696:
* If the R flag = 0, indicating success, then the STA shall derive rMSK.
* If the R flag = 1, indicating failure, then the STA shall abandon the FILS authentication.
* If PFS is being used for the exchange, the FILSR’s public key shall be converted from an octet string to an element according to the conversion in 12.4.7.2.5 (Octet string to element conversion). Then the FILSO shall validate the FILSR’s public key in a group-specific fashion as described in 5.6.2.3 of (#298)NIST Special Publication 800-56A R2. If validation fails, the FILSO shall terminate the FILS authentication protocol. Otherwise, the FILSO shall perform the group’s scalar-op (see 12.4.4.1 (General)) with the FILSR’s ephemeral public key and its own ephemeral private key to produce an ephemeral Diffie-Hellman shared secret, DHss.
* The FILSO shall perform key derivation per 12.11.2.5 (Key establishment with FILS authentication) and key confirmation per 12.11.2.6 (Key confirmation with FILS authentication).

If the FILSO was attempting EAP-RP Authentication and did not receive an Authentication frame, then the FILSO should perform retransmission procedure as defined in IETF RFC 6696, and with implementation-specific timeouts as guided by the AuthenticateFailureTimeout parameter in the MLME-AUTHENTICATE.request. If the retransmission procedure fails, then the FILSO shall abandon the FILS authentication and should perform full EAP authentication via IEEE 802.1X authentication.

If the FILSO was attempting PMKSA caching and did not receive an Authentication frame from the FILSR, the FILSO may attempt to use an alternate authentication method.

* Key establishment with FILS Public Key authentication
* General

This subclause defines the procedure for establishing a shared key between a FILS STA or non-AP MLD and AP or AP MLD using FILS Public Key authentication.

When using FILS Public Key authentication, a STA or non-AP MLD shall initiate authentication using either a public key or a cached PMKSA.

* Prior to exchange

FILS Public Key authentication performs key establishment with a Diffie-Hellman exchange. Prior to beginning the exchange, the non-AP STA or non-AP MLD (i.e. the FILSO) performs the following:

* Selects a finite cyclic group from the dot11RSNAConfigDLCGroupTable to perform the Diffie-Hellman exchange.
* Generates a random 16-octet nonce, generates an ephemeral private key, and uses the selected group’s scalar-op (see 12.4.4.1 (General)) with its private key to generate its ephemeral public key.
* Determine whether to attempt PMKSA caching and if so, generates a list of PMKSA identifiers.
* Constructs an Authentication frame (see 9.3.3.11 (Authentication frame format)) as follows:
* The Authentication algorithm number shall be set to 6 (FILS Public Key authentication) (see 9.4.1.1 (Authentication Algorithm Number field)) and the Authentication transaction sequence number shall be set to 1.
* The random nonce shall be encoded in the FILS Nonce element (see 9.4.2.188 (FILS Nonce element)).
* The chosen finite cyclic group shall be encoded in the Finite Cyclic Group field (see 9.4.1.40 (Finite Cyclic Group field)).
* The FILSO’s ephemeral public key shall be encoded into the FFE field (see 9.4.1.38 (FFE field)) according to the element to (#1288)octet string conversion in 12.4.7.2.4 (Element to octet string conversion).
* The random FILS Session shall be encoded in the FILS Session element (see 9.4.2.178 (FILS Session element)).
* If a list of PMKSA identifiers was generated, it shall be used to construct the PMKID List field in RSNE.

The FILSO then transmits the Authentication frame to the FILSR.

* Processing after receipt

Upon reception of the Authentication frame with the Authentication algorithm number equal to 6, the FILSR shall perform the following procedure:

* If the finite cyclic group indicated by the Finite Cyclic Group field is not acceptable, the FILSR shall respond with an Authentication frame with the status code of 77 (“Authentication is rejected because the offered finite cyclic group is not supported”) and terminate the FILS authentication protocol.
* If the finite cyclic group is acceptable, the FILSR shall verify the validity of the FILSO’s ephemeral public key:
* The public key is converted from an octet string to an element according to the conversion in 12.4.7.2.5 (Octet string to element conversion).
* The public key, as a group element, is verified in a group-specific fashion as described in 5.6.2.3 of (#298)NIST Special Publication 800-56A R2. If verification fails, the FILSR shall terminate the FILS authentication protocol.
* The FILSO’s nonce and validated ephemeral public key are extracted from the Authentication frame.
* The FILSR shall check whether PMKSA caching is being attempted by the presence of the PMKSA List field in RSNE. If the PMKID List field is present in RSNE, the FILSR checks whether any PMKSA identifier offered in the PMKID List matches an identifier for a cached PMKSA. If so, the FILSR selects a PMKID that matches and shall continue the FILS Public Key authentication protocol using PMKSA caching, otherwise the FILSR shall continue the FILS Public Key authentication protocol using digital signatures.
* Post processing

Next, the FILSR shall

* Generate a random 16-octet nonce and a random ephemeral private key, and then uses the agreed-upon group’s scalar-op (see 12.4.4.1 (General)) with its private key to generate its ephemeral public key.
* Construct an Authentication frame (see 9.3.3.11 (Authentication frame format)) as follows:
* The Authentication algorithm number is set to 6 (FILS Public Key authentication) (see 9.4.1.1 (Authentication Algorithm Number field)), and the Authentication transaction sequence number is set to 2.
* The random nonce is encoded in the FILS Nonce element (see 9.4.2.188 (FILS Nonce element)).
* The finite cyclic group is encoded in the Finite Cyclic Group field (see 9.4.1.40 (Finite Cyclic Group field)).
* The FILSR’s ephemeral public key is encoded in the FFE field (see 9.4.1.38 (FFE field)) according to the element to (#1288)octet string conversion in 12.4.7.2.4 (Element to octet string conversion).
* The FILSR copies the FILS Session element from the Authentication frame received from the FILSO.
* If the PMKSA caching is used, the FILSR indicates the selected PMKID in the PMKID List.
* Transmit the Authentication frame to the FILSO.
* Compute the Diffie-Hellman shared secret, DHss, based on the FILSO’s ephemeral public key and its own ephemeral private key with the chosen group’s scalar-op.
* Perform key derivation (see 12.11.2.5 (Key establishment with FILS authentication)).
* Upon receipt

The FILSO processes the received Authentication frame as follows:

* The FILSO shall abandon FILS authentication if any of the following conditions occur:
* ) The received Authentication frame does not include the Authentication Algorithm Number equal to 6 (FILS Public Key authentication) (see 9.4.1.1 (Authentication Algorithm Number field)).
* ) PMKSA caching was attempted and the received Authentication frame includes a PMKID that does not match a PMKID in the Authentication frame sent by the FILSO.
* ) The received Authentication frame does not include the FILS Session element.
* Verifies that the finite cyclic group in the FILSR’s response is equal to the group selected by the FILSO and that the FILS Session element received from the FILSR is equal to the FILS Session selected by the FILSO. If these differ, the FILSO shall terminate the authentication exchange.
* Verifies the validity of the FILSR’s ephemeral public key:
* The public key is converted from an octet string to an element according to the conversion in 12.4.7.2.5 (Octet string to element conversion).
* The public key, as a group element, is verified in a group-specific fashion according to 5.6.2.3 of (#298)NIST Special Publication 800-56A R2. If public key validation fails the FILSO shall terminate the authentication exchange.
* Extracts the FILSR’s nonce and verified ephemeral public key from the Authentication frame.
* Compute the Diffie-Hellman shared secret, DHss, based on the FILSR’s ephemeral public key and its own ephemeral private key with the chosen group’s scalar-op to derive DHss.
* Performs key derivation (see 12.11.2.5 (Key establishment with FILS authentication)) and begins key confirmation (see 12.11.2.6 (Key confirmation with FILS authentication)).

If the FILSO was attempting PMKSA caching and did not receive an Authentication frame from the FILSR, the FILSR may attempt to use an alternate authentication method.

* Key establishment with FILS authentication
* General

When not using PMKSA caching, a PMK is created according to 12.11.2.5.2 (PMKSA key derivation with FILS authentication). When using PMKSA caching, a new PMKSA is not created. Instead, the PMKSA used for PMKSA caching remains and continues to be identified by the appropriate PMKID. Regardless of whether PMKSA caching is used or not, a PTKSA shall be generated with each FILS authentication exchange.

PTKSA creation uses the KDF from 12.7.1.6.2 (Key derivation function (KDF)) to derive the following keys from the PMK: an integrity check key (ICK), a (#3744)PTK key encryption key (PTK-KEK), and a temporal key (TK).

PTKSA key establishment shall immediately be followed by key confirmation per 12.11.2.6 (Key confirmation with FILS authentication).

* PMKSA key derivation with FILS authentication

The PMK is derived using the two nonces and the secret(s) from FILS Key establishment. A PMKID used to identify the PMKSA is generated using the hash algorithm from the negotiated AKM on input data specific to the FILS Key Establishment step. The length of the PMK shall be either 256 bits or 384 bits depending on the negotiated AKM, and the length of the PMKID shall be 128 bits. If FILS Shared Key authentication was used to generate input keying material, the PMK and PMKID are derived as follows:

PMK = HMAC-Hash(SNonce || ANonce, rMSK [ || DHss ])

PMKID = Truncate-128(Hash(EAP-Initiate/Reauth))

When FILS Public Key authentication is used to generate input keying material, the PMK and PMKID are derived as follows:

PMK = HMAC-Hash(SNonce || ANonce, DHss)

PMKID = Truncate-128(Hash(gSTA || gAP))

where

SNonce is the FILSO nonce and ANonce is the FILSR nonce

rMSK is the shared secret from the EAP-RP exchange

DHss is the shared secret derived from the Diffie-Hellman exchange, when performed; when ECC is used, only the x-coordinate from EDCH is included

Brackets indicate the inclusion of the shared secret when doing a Diffie-Hellman exchange; there is no shared secret to include otherwise

EAP-Initiate/Reauth is the EAP-RP packet sent by the STA during key establishment with FILS Shared Key authentication

gSTA is the FILSO’s Diffie-Hellman value

gAP is the FILSR’s Diffie-Hellman value

Hash is the hash algorithm specific to the negotiated AKM (see Table 9-188 (AKM suite selectors))

Upon completion of PMK and PMKID generation the shared secret, DHss, and rMSK, if applicable, shall be irretrievably deleted.

* PTKSA Key derivation with FILS authentication

For PTKSA key generation, the inputs to the PRF are the PMK of the PMKSA, a constant label, and a concatenation of the FILSO’s MAC address, the FILSR’s MAC address, the FILSO’s nonce, and the FILSR’s nonce. When the negotiated AKM is 00-0F-AC:14 or 00-0F-AC:16, the length of (#3744)PTK-KEK shall be 256 bits, and the length of the ICK shall be 256 bits. When the negotiated AKM is 00-0F-AC:15 or 00-0F-AC:17, the length of the (#3744)PTK-KEK shall be 512 bits, and the length of ICK shall be 384 bits. When the negotiated AKM is 00-0F-AC:16, FILS-FT is 256 bits; when the negotiated AKM is 00-0F-AC:17, FILS-FT is 384 bits; otherwise, FILS-FT is not derived(11ba); when WUR frame protection is negotiated, the length of KDK is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)); otherwise, the KDK is not derived. The total amount of bits extracted from the KDF shall therefore be (11ba)(#3260)640+TK\_bits, 1124+TK\_bits, or 1408+TK\_bits depending on the negotiated AKM when WUR frame protection is negotiated, otherwise, shall be (#3260)512+TK\_bits, 896+TK\_bits, or 1280+TK\_bits depending on the negotiated AKM, where TK\_bits are determined from Table 12-8 (Cipher suite key lengths(#1083)(#3532)):

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

ICK = (#3506)ExtractBits(PTK(#1778), 0, ICK\_bits)

(#3744)PTK-KEK = (#3506)ExtractBits(PTK(#1778), ICK\_bits, KEK\_bits)

TK = (#3506)ExtractBits(PTK(#1778), ICK\_bits + KEK\_bits, TK\_bits)

When doing FT initial mobility domain association using FILS authentication,

FILS-FT = (#3506)ExtractBits(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits, FILS-FT\_bits)

(11ba)When WUR frame protection is negotiated while doing FT initial mobility domain association using FILS authentication,

KDK = (#3506)ExtractBits(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits + FILS-FT\_bits, KDK\_bits)

(11ba)When WUR frame protection is negotiated while not doing FT initial mobility domain association using FILS authentication,

 KDK = (#3506)ExtractBits(PTK(#1778), ICK\_bits + KEK\_bits + TK\_bits, KDK\_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

(11ba)KDK\_bits is the length of KDK in bits and is equal to the value of PMK\_bits (see 12.7.1.3 (Pairwise key hierarchy)).

X is 512+TK\_bits, (#3260)768+TK\_bits, 896+TK\_bits, or 1280+TK\_bits from Table 12-8 (Cipher suite key lengths(#1083)(#3532)) depending on the negotiated AKM (11ba)when WUR frame protection is negotiated; otherwise, X

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 MPMK (see 12.7.1.6.3 (PMK-R0))

SPA is the FILSO’s MAC address (i.e. non-AP STA or, for MLO, non-AP MLD MAC address) and the AA is the FILSR’s MAC address (i.e. AP’s BSS’s BSSID or, for MLO, AP MLD’s MAC address)

SNonce is the FILSO’s nonce

ANonce is the FILSR’s nonce

DHss is the shared secret derived from the Diffie-Hellman exchange, when performed, and PMKSA caching is used

Brackets indicate the inclusion of the shared secret when doing a Diffie-Hellman exchange while using PMKSA caching; there is no shared secret to indicate otherwise

Upon completion of PTK(#1778) generation the shared secret, DHss, shall be irretrievably deleted if Diffie‑Hellman exchange was performed.

(11ba)If WUR frame protection is negotiated, the WTK shall be derived from the KDK using the KDF defined in 12.7.1.6.2 (Key derivation function (KDF)):

WTK = (#3686)KDF-*Hash*-*Length*(KDK, “WUR Temporal Key”, SPA || AA || SNonce || ANonce
             [ ||DHss ])

where

* (#3686)KDF-*Hash*-*Length* is the 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 9-188 (AKM suite selectors)).
* (#3686)*Length* is the total number of bits to derive, i.e., number of bits of the WTK, and is equal to 128.
* Key confirmation with FILS authentication
* General

Key confirmation for FILS authentication is a (Re)Association Request frame followed by a (Re)Association Response frame. Components of the (Re)Association Request and (Re)Association Response frames shall be protected using (#3744)PTK-KEK.

* (Re)Association Request for FILS key confirmation

The STA constructs a (Re)Association Request frame for FILS authentication per 9.3.3.5 (Association Request frame format) and 9.3.3.7 (Reassociation Request frame format). Hash algorithms are used to generate the FILS Key Confirmation element and the specific hash algorithm depends on the negotiated AKM (9.4.2.23.3 (AKM suites)).

If dot11RSNAOperatingChannelValidationActivated is true and AP indicates OCVC(#3505), the STA shall include an OCI element(#426) in the request.

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

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

where

Hash is the hash algorithm specific to the negotiated AKM (see Table 9-188 (AKM suite selectors))

SNonce is the FILSO’s nonce

ANonce is the FILSR’s nonce

STA-MAC is the MAC address of the FILSO (i.e. non-AP STA MAC address or, for MLO, non-AP MLD MAC address) and AP-BSSID is the MAC address of the FILSR (i.e. BSSID of the AP’s BSS or, for MLO, AP MLD’s MAC address)

gSTA is the FILSO’s Diffie-Hellman public value

gAP is the FILSR’s Diffie-Hellman public value

Brackets indicate the inclusion of the Diffie-Hellman public values when doing PFS with FILS Shared Key authentication or PMKSA caching with FILS Public Key authentication; there are no Diffie-Hellman public values to include otherwise

For FILS Public Key authentication when PMKSA caching is not used, the KeyAuth field of the FILS Key Confirmation element is a digital signature using the FILSO’s private key, of the negotiated hash algorithm on a concatenation of the FILSO’s public Diffie-Hellman value, the FILSR’s public Diffie-Hellman value, the FILSO’s nonce, the FILSR’s nonce, the FILSO’s MAC address, and the FILSR’s MAC address, in that order:

Key-Auth = Sig-STA(gSTA || gAP || SNonce || ANonce || STA-MAC || AP-BSSID)

where

Sig-STA( ) indicates a digital signature using the STA’s private key, analog to the STA’s trusted public key

The form of signature depends on the type of public key used by the STA (IETF RFC 3447 for RSA, FIPS 186‑4 for DSA, and ISO/IEC 14888-3 for ECDSA). The data to be signed is first hashed and the hash algorithm used with the appropriate digital signature algorithm shall be specific to the negotiated AKM.

The (Re)Association Request frame shall be encrypted using the AEAD algorithm as defined in 12.11.2.7 (AEAD cipher mode for FILS) with the (#3744)PTK-KEK as the key. The AAD used with the AEAD algorithm for the Association Request frame consists of the following data passed as separate components in the following order:

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

The plaintext passed to the AEAD algorithm is the data that would follow the FILS Session element in an unencrypted frame body. The output of the AEAD algorithm becomes the data that follows the FILS Session element in the encrypted and authenticated (Re)Association Request frame. The output of the algorithm is as specified in IETF RFC 5116. The resulting (Re)Association Request frame shall be transmitted to the FILSR.

The FILSR compares FILS session of the received (Re)Association Request frame with the FILS session that was used to identify the FILS session in the Authentication frames. If they differ, authentication exchange fails.

If dot11RSNAOperatingChannelValidationActivated is true and the FILSO indicates OCVC(#3505) in the RSNE in the request, FILSR shall validate the OCI element in the request by ensuring that all of the following are true:

* OCI element is present.
* Channel information in the OCI matches current operating channel parameters (see 12.2.9 (Requirements for Operating Channel Validation)).

Otherwise, the FILSR rejects the request by discarding the frame.

The FILSR decrypts and verifies the received (Re)Association Request frame with the AEAD algorithm as defined in 12.11.2.7 (AEAD cipher mode for FILS) with the (#3744)PTK-KEK as the key. The AAD is reconstructed as defined above and is passed, along with the cipher text of the received frame, to the AEAD decryption operation.

If the output from the AEAD decryption operation returns a failure, the authentication exchange fails. If the output does not return failure, the output plaintext replaces the cipher text as portion of the frame body that follows the FILS Session element and processing of the received frame continues by checking the value of the FILS Key Confirmation element.

The FILSR verifies that the RSNE received in the (Re)Association Request frame has identical AKM suite and cipher suites and RSN capabilities as were included in the RSNE in the Authentication frame from the FILSO. If these fields differ, the authentication exchange fails.

For FILS Shared Key authentication, the FILSR constructs a verifier, Key-Auth', in an identical manner as the FILSO constructed its Key-Auth above.

The FILSR compares Key-Auth' with the KeyAuth field in the FILS Key Confirmation element of the received frame. If they differ, authentication fails.

For FILS Public Key authentication, the FILSR uses the FILSO’s (certified) public key from the FILS Public Key element to verify that the signature contained in the KeyAuth field corresponds to the purported signature by the FILSO over the concatenation of the following:

* FILSO’s public Diffie-Hellman value gSTA
* FILSR’s public Diffie-Hellman value gAP
* FILSO’s nonce SNonce, the FILSR’s nonce ANonce
* FILSO’s MAC address STA-MAC
* FILSR’s MAC address AP-BSSID

in that order, according to the signature scheme used. Furthermore, the FILSR checks all certificates in the certificate chain, both cryptographically and from a security policy perspective, according to the procedures for checking certificates and certificate chains in IETF RFC 5280. If any of these verifications fail, authentication fails.

If authentication is deemed a failure, ICK, (#3744)PTK-KEK, TK, and the PTKSA shall be irretrievably deleted and the FILSR shall return an Authentication frame with a status code set to 112 (Authentication rejected due to FILS authentication failure). If PMKSA caching was not being employed for this failed authentication attempt, the PMKSA shall also be deleted. If PMKSA caching was being employed, the reason for failure might be an impersonation attack. Therefore, when FILS with PMKSA caching fails, the FILSR may decide to retain the cached PMKSA.

* (Re)Association Response for FILS key confirmation

The FILSR constructs a (Re)Association Response frame for FILS authentication per 9.3.3.6 (Association Response frame format) and 9.3.3.8 (Reassociation Response frame format). As with the (Re)Association Request frame, hash algorithms are used to generate the FILS Key Confirmation element and the specific hash algorithm depends on the negotiated AKM (see 9.4.2.23.3 (AKM suites)).

The FILSR constructs a Key Delivery element indicating the current GTK and (#1406)GTK PN, and the current IGTK and IPN if management frame protection is enabled, and the current BIGTK and BIPN if beacon protection is enabled(11ba), and the current WIGTK and WIPN if WUR frame protection is enabled. For non-MLO, the GTK is carried in a GTK KDE(#3492), the IGTK and IPN are carried in an IGTK KDE, the BIGTK and BIPN are carried in a BIGTK KDE, and the WIGTK and WIPN are carried in a WIGTK KDE. For MLO, the GTKs for all setup links are carried in MLO GTK KDEs, the IGTKs in MLO IGTK KDEs, and the BIGTKs in MLO BIGTK KDEs. The FILSR puts this element into the (Re)Association Response frame.

Tgbe editor: Consider adding an Editor’s note (similar to in clause 13 for FT) as follows: Do we need to add MLO WIGTK? (a KDE for which is not yet defined in 12.7.2)

If dot11RSNAOperatingChannelValidationActivated is true and FILSO indicates OCVC(#3505), the FILSR shall include an OCI element(#426) in the response.

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

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

where

Hash is the hash algorithm specific to the negotiated AKM (see Table 9-188 (AKM suite selectors))

ANonce is the FILSR’s nonce

SNonce is the FILSO’s nonce

AP-BSSID is the FILSR’s MAC address and STA-MAC is the FILSO’s MAC address

gAP is the FILSR’s Diffie-Hellman public value

gSTA is the FILSO’s Diffie-Hellman public value

Brackets indicate the inclusion of the Diffie-Hellman public values when doing PFS with FILS Shared Key authentication; there are no Diffie-Hellman public values to include otherwise

For FILS Public Key authentication when PMKSA caching is not used, the KeyAuth field of the FILS Key Confirmation element is a digital signature using the FILSR’s private key of the output from the negotiated hash algorithm on a concatenation of the FILSR’s public Diffie-Hellman value, the FILSO’s public Diffie-Hellman value, the FILSR’s nonce, the FILSO’s nonce, the FILSR’s MAC address, and the FILSO’s MAC address, in that order. The specific construction of the digital signature depends on the crypto-system of the public/private key pair:

Key-Auth = Sig-AP(gAP || gSTA || ANonce || SNonce || AP-BSSID || STA-MAC)

where

Sig-AP() indicates a digital signature using the AP’s private key analog to the AP’s trusted public key

The form of signature depends on the type of public key used by the AP (IETF RFC 3447 for RSA, FIPS 186‑4 for DSA, and ISO/IEC 14888-3 for ECDSA). The data to be signed is first hashed and the hash algorithm used with the appropriate digital signature algorithm shall be specific to the negotiated AKM.

The (Re)Association Response frame shall be encrypted using the AEAD algorithm as defined in 12.11.2.7 (AEAD cipher mode for FILS) with the (#3744)PTK-KEK as the key. The AAD used with the AEAD algorithm for the (Re)Association Response frame consists of the following data passed as separate components in the following order:

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

The plaintext passed to the AEAD algorithm is the data that would follow the FILS Session element in an unencrypted frame body. The output of the AEAD algorithm becomes the data that follows the FILS Session element in the encrypted and authenticated (Re)Association Response frame. The output of the algorithm is as specified in IETF RFC 5116. The resulting (Re)Association Response frame shall be transmitted to the FILSO.

If dot11RSNAOperatingChannelValidationActivated is true and the FILSR indicates OCVC(#3505) in its RSNE, the FILSO shall validate the OCI element in the response by ensuring that all of the following are true:

* OCI element is present
* Channel information in the OCI matches current operating channel parameters (see 12.2.9 (Requirements for Operating Channel Validation))

Otherwise, the FILSO shall discard the frame.

The FILSO decrypts and verifies the received (Re)Association Response frame with the AEAD algorithm as defined in 12.11.2.5 (Key establishment with FILS authentication) with the (#3744)PTK-KEK as the key. The AAD is reconstructed as defined in this subclause above and is passed with the cipher text of the received frame to the AEAD decryption operation.

The FILSO compares FILS Session of the received frame with the FILS Session it selected to identify the FILS session. If they differ, authentication fails.

If the output from the AEAD decryption operation returns failure, the authentication exchange fails. If the output does not return failure, the output plaintext replaces the cipher text as portion of the frame body that follows the FILS Session element and processing of the received frame continues by checking the value of the FILS Key Confirmation element.

The FILSO verifies that the RSNE received in the (Re)Association Response frame has identical AKM suites and cipher suites and RSN capabilities as were included in the RSNE in the Beacon, Probe Response, and Authentication frames from the FILSR. If these fields differ, authentication fails.

For FILS Shared Key authentication, the FILSO constructs a verifier, Key-Auth', in an identical manner as the FILSR constructed its Key-Auth above.

The FILSO compares Key-Auth' with the KeyAuth field in the FILS Key Confirmation element of the received frame. If they differ, authentication fails.

For FILS Public Key authentication, the FILSO uses the FILSR’s (certified) public key from the FILS Public Key element to verify that the signature contained in the KeyAuth field corresponds to the purported signature by the FILSR over the concatenation of the following:

* FILSR’s public Diffie-Hellman value gAP
* FILSO’s public Diffie-Hellman value gSTA
* FILSR’s nonce ANonce
* FILSO’s nonce SNonce
* FILSR’s MAC address AP-BSSID
* FILSO’s MAC address STA-MAC

in that order, according to the signature scheme used. Furthermore, the FILSR checks all certificates in the certificate chain, both cryptographically and from a security policy perspective, according to the procedures for checking certificates and certificate chains in IETF RFC 5280. If any of these verifications fail, authentication fails.

If authentication is deemed a failure, the ICK, (#3744)PTK-KEK, PMK, and TK shall be irretrievably deleted and the FILSO shall abandon the exchange. Otherwise authentication succeeds and the FILSO and FILSR shall irretrievably delete the nonpersistent secret keying material that is created by executing the key establishment with FILS Shared Key authentication scheme (12.11.2.3 (Key establishment with FILS Shared Key authentication)) or the key establishment with FILS Public Key authentication scheme (12.11.2.4 (Key establishment with FILS Public Key authentication)). The KEK and PMK shall be used for subsequent key management as specified in 12.6 (RSNA security association management). If the lifetime of the rMSK is known, the FILSO and FILSR shall set the lifetime of the PMKSA to the lifetime of the rMSK. Otherwise, the FILSO and FILSR shall set the lifetime of the PMKSA to the value dot11RSNAConfigPMKLifetime.

Upon successful completion of the FILS authentication procedure, the FILSO shall process the Key Delivery element in the (Re)Association Response frame. The FILSO installs the GTK and (#1406)GTK RSC, and IGTK and IGTK RSC if management frame protection is enabled, and BIGTK and BIGTK RSC if present in the (#1488)Key Delivery element and dot11BeaconProtectionEnabled is true(11ba), and WIGTK and WIGTK RSC if present in the (#1488)Key Delivery element and dot11RSNAWURFrameProtectionActivated is true. For MLO, the FILSO installs the GTKs, IGTKs and BIGTKs for each setup link.

* AEAD cipher mode for FILS

FILS authentication uses an AEAD cipher mode to protect (Re)Association Request/Response and EAPOL-Key frames. The AEAD cipher mode is determined by the specific FILS negotiated AKM.

AES-SIV-256 is used when the negotiated AKM is 00-0F-AC:14 or 00-0F-AC:16 and AES-SIV-512 is used when the negotiated AKM is 00-0F-AC:15 or 00-0F-AC:17.