IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resolution for CIDs 59 and 62 “Remove DLS and STSL” | | | | |
| Date: 2017-09 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Graham SMITH | SR Technology | Davie, FL, USA. | 916 799 9563 | gsmith@srtrl.com |

Abstract

This submission proposes resolutions for CID 59 and 62.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CID | Commenter | Clause | Page | Line | Comment | Proposed |
| 59 | Graham Smith | 11.7 | 1806 | 5 | Time to remove DLS? | Remove |
| 62 | Graham Smith | 12.2.5 | 2060 | 4 | Time to remove STSL support? | Remove |

CID 59 and CID 62 DLS and STSL

1806.10

*The STSL mechanism is obsolete. Consequently, the DLS protocol might be removed in a later revision of the standard.*

STSL = station to station link.There are 60 instances of STSL in the text mostly on key management.

2060.4

*The STSL mechanism is obsolete. Consequently, the PeerKey protocol components that do not support the AP PeerKey protocol might be removed in a later revision of the standard.*

Need to check Peer Key components.

At 1806.20 *A DMG STA shall not use the DLS protocol.*

I think that DLS could safely be removed.

Discussed in Berlin

A question is whether TDLS is reliant upon anything in DLS.

Have reached out to Menzo. Menzo thinks no, Mark wants to check that a generic DLS may be used and needs to be checked.

Consensus to remove.

CIDs 59 and 62

RESOLUTION

REVISED

Note to Editor: References are to D0.1

146.11 delete lines 11 and 12 (STSL, STK, SMK)

161.48 delete lines 48 to 50 (STSL, SMK, STK, SMKSA, STKSA)

166.58 delete lines 58 to 64 (two entries) (STSL, SMKSA)

167.1 delete lines 1 to 16 (four entries) (STSL, SMK, STKSA)

175.64 delete line “DLS direct-link setup

183.21 delete SKCK and SKEK entries

183.32 delete SMK and SMKSA entries

184.1 delete STK, STKSA and STSL entries (3 entries)

195.33 delete “direct-link setup (DLS),”

195.36 delete “, DLS,”

203.47 delete “, unlike with DLS”

209.30 delete “- Direct-link setup (DLS)”

215.11 delete “DLS,”

226.54 delete “and for DLS”

227.50 delete “station-to-station link (STSL) master key security association (SMKSA), STSL transient key security association (STKSA),”

370.1 to 375.28 delete 6.3.27.2 MLME-DLS.request, 6.3.27.3 MLME-DLS.confirm, 6.3.27.4 MLME-DLS.indication, 6.3.27.5 MLME-DLS.response, 6.3.27.6 MLME-DLS-TEARDOWN.request, 6.3.27.7 MLME-DLS-TEARDOWN.indication

770.51 delete “A STA within an infrastructure BSS sets the Privacy subfield to 1 in DLS Request and DLS Response frames if encryption is required for all Data frames exchanged. If encryption is not required, the Privacy subfield is set to 0.”

771.19 delete “A STA sets the Short Preamble subfield to 1 in transmitted Association Request and Reassociation Request frames and in DLS Request and DLS Response frames when dot11ShortPreambleOptionImplemented is true. Otherwise, a STA sets the Short Preamble subfield to 0.”

771.32 edit as shown “A STA sets the Short Slot Time subfield to 1 in transmitted Association Request, and Reassociation Request frames when…”

772.38 delete “DLS Teardown,”

774.34 delete “END\_DLS”

774.40 change entry 42 from PEERKEY\_MISMATCH to Reserved

774.41 Delete “In a DLS Teardown frame: The teardown was initiated by the DLS peer”

774.47 Delete “In a DLS Teardown frame: The teardown was initiated by the AP”

778.41 Change entry 48 to “Reserved”

781.51 Change entry 2 to “Reserved”

783.53 to 64 Delete 9.4.1.13 DLS Timeout Value field entirely`

944.20 delete “and for DLS”

948.11 delete /STKSA

948.13 delete /STKSA

948.13 delete "The number of replay counters per STKSA is the same as the number of replay counters per PTKSA."

948.16 in Table 9-134 delete all occurrences of /STKSA (5x)

948.33 delete /STKSA

949.5 delete "and STKSA"

949.7 delete "and STKSA"

1145.49 Figure 9-529—Multi-band Connection Capability field format, Replace “DLS” in B2 with “Reserved”

1145.63 delete “The DLS subfield is set to 1 to indicate that the STA can perform a DLS on the channel and band indicated in the element. Otherwise, it is set to 0.”

1151.10 Table 9-244—Session Type subfield values, replace “DLS with “Reserved”

1255.10 to 1258.11 delete clause 9.6.4 DLS Action frame details, entirely

1403.36 delete "STKSA" and move the "and" to before "GTKSA"

1479.56 and 1479.58 delete “DLS or”

1480.12 edit as follows: “A STA that transmits a VHT PPDU to a TDLS peer STA obtains the AID for the peer STA from the TDLS Setup Request or TDLS Setup Response frame.”

1598.58 edit as follows: “NOTE - A STA that transmits a VHT PPDU to a TDLS peer STA obtains the AID for the peer STA from the TDLS Setup Request or TDLS Setup Response frame.”

1600.6 and 1600.42 delete “DLS or”

1720.5 edit as follows: “A non-AP QoS STA may be in PS mode before the setup of block ack. BUs for a TID without a schedule are sent using Normal Ack following a PS-Poll frame as described in rest of 11.2.3. Uplink block ack agreements, block ack agreements for any TID with a schedule, and any block ack agreements to APSD STAs continue to operate normally.”

1748.56 delete “DLS or”

1765.11 delete “ii) Data frames between peers using DLS”

1774.7 delete "SMKSAs, STKSAs and"

1774.15 delete “STSL, DLS and”

1781.49 edit as follows: “In the TDLS direct-link case, it is the responsibility of the STA that is going to send the data to create the TS. In this case, the STA negotiates with the HC to gain TXOPs that it uses to send the data. There is no negotiation between the originator and recipient STAs concerning the TS: the originator can discover the capabilities of the recipient (rates, Block Ack) using TDLS.”

1801.14 delete “If the STA is establishing a block ack agreement with another STA through DLS, then the DLS setup procedure includes the exchange of capability information that allows both STAs to determine whether the other STA is an HT STA.”

1806.5 to 1811.41 Delete Clause 11.7 entirely

1811.29-41 delete 11.7.6 (Secure DLS operation) entirely

1847.15 delete “This enables a QoS AP to query Transmit Stream/Category Measurement metrics for DLS links.”

1888.53 to 57 delete “Simultaneous operation of DLS and TDLS between the same pair of STAs is not allowed. A DLS Request frame shall not be transmitted to a STA with which a TDLS direct link is currently active. A DLS Request frame received from a STA with which a TDLS direct link is currently active shall be discarded.”

1972.42 delete row “DLS 1101 2 0-2 AC\_BE”

2033.3 delete “- A DLS link”

2060.1-18 delete 12.2.5 RSNA PeerKey Support entirely

2103.9 delete "STKSA" and move the "and" to before GTKSA.

2103.17 delete "STKSA" and move the "and" to before GTKSA.

2103.27 delete "STKSA" and move the "and" to before GTKSA.

2108.23 delete "STKSA" and move the "and" to before GTKSA.

2110.8 delete "STKSA" and move the "and" to before GTKSA.

2110.11 delete "STKSA" and move the "and" to before GTKSA.

2116.28 delete "STKSA" and move the "and" to before GTKSA.

2118.22 delete "STKSA" and move the "and" to before GTKSA.

2118.26 delete "STKSA" and move the "and" to before GTKSA.

2119.46 delete " — STKSA: A result of a successful 4-way STK handshake following the initial SMK handshake or subsequent rekeying."

2124.1-40 delete 12.6.1.1.11 (SMKSA) and 12.6.1.1.12 (STKSA) entirely

2130.15 delete “and for DLS”

2130.29 edit as follows “in any of its Beacon or Probe Response frames”

2138.21 delete “and for DLS”

2138.35 delete “and for DLS”

2138.44 delete “and for DLS”

2139.32 delete "any STKSA" and move the "and"

2139.32 delete “and invoke an STSL application teardown procedure for any of its STKSAs. An example of an STSL application teardown procedure is described in 11.7.4 (DLS teardown).”

2140.1 delete “If the SMK handshake fails between a pair of associated STAs and AP, then the STAs and the AP shall invoke an STSL application teardown procedure.”

2140.13 delete “When a STA’s SME receives an MLME-PN-EXHAUSTION.indication primitive and the PN is associated with a STKSA, the STA’s SME shall invoke a STSL application teardown procedure for the STKSA and delete the STKSA.”

2141.58 - 2142.27 delete

“When a STKSA is deleted, the STA\_I may establish a new STSL with the STA\_P. If the SMK between the STA pair has not expired, the STA\_I may initiate a 4-way handshake and create a new STKSA with STA\_P. If the SMK has expired, the STA\_I shall create both a new SMKSA and a new STKSA with the STA\_P.

An Authenticator/STA\_I may initiate a 4-way handshake for the purpose of renewing the key associated with a PTKSA or STKSA. A supplicant/STA\_P may send an EAPOL request message to the authenticator/STA\_I to request rekeying. In addition, if both the Authenticator and the Supplicant support multiple keys for individually addressed traffic, a smooth switchover to the new key is possible using the following procedure.

The IEEE 802.11 MAC shall issue an MLME-PN-WARNING.indication primitive when the Packet Number assignment for a particular PTKSA, GTKSA, or STKSA reaches or exceeds the threshold that is defined in dot11PNWarningThresholdLow and dot11PNWarningThresholdHigh for the first time. The indication shall be issued only once for a given PTKSA, GTKSA, or STKSA. The SME may use the indication as a trigger to establish a new PTKSA, GTKSA, or STKSA before the Packet Number space is exhausted.

A PTKSA or STKSA has a limited lifetime, either in absolute time or due to exhausting the PN space. To maintain an uninterrupted security association, a STA should establish a new PTKSA or STKSA prior to the expiration of the old PTKSA or STKSA.

When both ends of the link support extended Key IDs for individually addressed frames, it is possible to install the new PTKSA or STKSA without data loss, provided the new PTKSA or STKSA uses a different Key ID from the old PTKSA or STKSA. Data loss might occur if the same Key ID is used because it is not possible to precisely coordinate (due to software processing delays) when the new key is used for transmit at one end and when it is applied to receive at the other end. If a different Key ID is used for the new PTKSA or STKSA, then provided the new key is installed at the receive side prior to its first use at the transmit side there is no need for precise coordination. During the transition, received packets are unambiguously identified using the Key ID as belonging to either the old or new PTKSA or STKSA.”

2141.62 delete /STA\_I

2141.64 delete “or STKSA”

2141.64 delete /STA\_P

2141.64-65 delete /STA\_I

2142.4-11 change 3 occurrences of "PTKSA, GTKSA, or STKSA" to " PTKSA or GTKSA"

2142.13-27 delete 8 occurrences of "or STKSA"

2150.58 - 2151.21 delete 12.7.1.6 PeerKey key hierarchy entirely

2157.23 delete "— PeerKey initial SMK handshake to deliver the SMK and final 4-way STK handshake to deliver the STK to the initiating and peer STAs."

2158.11 delete B13 "SMK message" and change B14 to B13 in the Reserved field

2158.59 delete "/SMK"

2159.29 delete "or SMK handshake failure"

2159.31 delete "When the SMK Message bit is 1, Error shall be set to 1 to indicate the key data field contains an Error KDE."

2159.36 delete “, and is set to 1 by the STSL peer STA to request initiator STA rekeying of the STK”, and insert ", and" before "is set to 1 by a Supplicant in a Michael MIC Failure Report"

2159.47 delete "/SMK"

2159.49-51 delete "If an EAPOL-Key frame in which the Request bit is 1 has the SMK Message bit equal to 1, the initiator STA shall take appropriate action to create a new STK (based on 12.7.8 (PeerKey handshake))."

2159.56 delete "or SMK"

2159.58 delete "11) SMK Message (bit 13) specifies whether this EAPOL-Key frame is part of an SMK handshake. If the SMK handshake is not supported, the SMK Message subfield is reserved."

2162.15 change "SMK KDE" to "Reserved"

2162.37 delete "or SMK KDE"

2162.39 delete "or SMK KDE"

2163.7-15 delete "The format of the SMK KDE is shown in Figure 12-38 (SMK KDE format)." and delete Figure 12-38—SMK KDE format

2163.37 delete “Table 12-7 (SMK error types) shows different values of SMK error types.”

2163.49 delete Table 12-7 (SMK error types)

2164.62 delete "PeerKey handshake messages use EAPOL-Key frames as defined in 12.7.8 (PeerKey handshake)."

2166.47 modify 12.7.4 (EAPOL-Key frame notation, 12.7.5 (Nonce generation) and 12.7.6 (4-way handshake) as follows:

* EAPOL-Key frame notation

The following notation is used throughout the remainder of **Error! Reference source not found.** and 13.4 (FT initial mobility domain association) to represent EAPOL-Key frames:

EAPOL-Key(S, M, A, I, K, Reserved, KeyRSC, ANonce/SNonce, MIC, DataKDs)

where

S means the initial key exchange is complete. This is the Secure bit of the Key Information field.

M means the MIC is available in message. This should be set in all messages except message 1 of a 4-way handshake. This is the Key MIC bit of the Key Information field. When the negotiated AKM is 00-0F-AC:14, 00-0F-AC:15, 00-0F-AC:16, or 00-0F-AC:17, this Key MIC bit is set to 0 regardless of the M parameter value(11ai).

A means a response is required to this message. This is used when the receiver should respond to this message. This is the Key Ack bit of the Key Information field.

I is the Install bit: Install/Not install for the pairwise key. This is the Install bit of the Key Information field.

K is the key type: P (Pairwise), G (Group/SMK). This is the Key Type bit of the Key Information field.

Reserved reserved.

KeyRSC is the key RSC. This is the Key RSC field.

ANonce/SNonce is the Authenticator/Supplicant nonce. This is the Key Nonce field.

MIC is the integrity check, which is generated using the KCK. This is the Key MIC field. When the negotiated AKM is 00-0F-AC:14, 00-0F-AC:15, 00-0F-AC:16, or 00-0F-AC:17, this Key MIC field is not included regardless of the MIC parameter value(11ai).

DataKDs is a sequence of zero or more elements and KDEs, contained in the Key Data field, which may contain the following:

RSNE is described in 9.4.2.25 (RSNE).

RSNE[KeyName] is the RSNE, with the PMKID field set to KeyName.

GTK[N] is the GTK, with the key identifier field set to N. The key identifier specifies which index is used for this GTK. Index 0 shall not be used for GTKs, except in mixed environments, as described in **Error! Reference source not found.**.

FTE is the Fast BSS Transition element, described in 9.4.2.48 (Fast BSS Transition element (FTE))

MDE is the Mobility Domain element, described in 9.4.2.47 (Mobility Domain element (MDE))

TIE[IntervalType] is a Timeout Interval element of type IntervalType, as described in 9.4.2.49 (Timeout Interval element (TIE)), containing e.g., for type KeyLifetime, the lifetime of the FT key hierarchy.

IGTK[M] is the IGTK, with key identifier field set to M.

IPN is the current IGTK replay counter value provided by the IGTK KDE

PMKID is of type PMKID KDE and is the key identifier used during 4-way PTK handshake for PMK identification.

* Nonce generation

The following is an informative description of Nonce generation.

All STAs contain a global key counter, which is 256 bits in size. It should be initialized at system boot-up time to a fresh cryptographic-quality random number. Refer to J.5 (Suggestions for random number generation) on random number generation. It is recommended that the counter value is initialized to the following:

PRF-256(Random number, “Init Counter”, Local MAC Address || Time)

The local MAC address should be AA on the Authenticator and SPA on the Supplicant.

The random number is 256 bits in size. Time should be the current time from Network Time Protocol (NTP) or another time in NTP format whenever possible. This initialization causes different initial key counter values to occur across system restarts regardless of whether a real-time clock is available. The key counter can be used as additional input data for nonce generation. A STA derives a random nonce for each new use.

* 4-way handshake
* General

RSNA defines a protocol using EAPOL-Key frames called the *4-way handshake*. The handshake completes the IEEE 802.1X authentication process. The information flow of the 4-way handshake is as follows:

Message 1: Authenticator  Supplicant: EAPOL-Key(0,0,1,0,P,0,0,ANonce,0,DataKD\_M1)

where DataKD\_M1 = 0 or PMKID for PTK generation

Message 2: Supplicant  Authenticator: EAPOL-Key(0,1,0,0,P,0,0,SNonce,MIC,DataKD\_M2)

where DataKD\_M2 = RSNE for creating PTK generation

Message 3: Authenticator  Supplicant:

EAPOL-Key(1,1,1,1,P,0,KeyRSC,ANonce,MIC,DataKD\_M3)

where DataKD\_M3 = RSNE,GTK[N] for creating PTK generation or initiator RSNE

Message 4: Supplicant  Authenticator: EAPOL-Key(1,1,0,0,P,0,0,0,MIC,DataKD\_M4)

where DataKD\_M4 = 0.

The FT initial mobility domain association uses the FT 4-way handshake to establish an initial PTKSA, GTKSA and, if management frame protection is enabled, an IGTKSA, that is based on this protocol. The FT 4-way handshake protocol is described in 13.4 (FT initial mobility domain association).

Here, the following assumptions apply:

* EAPOL-Key() denotes an EAPOL-Key frame conveying the specified argument list, using the notation introduced in **Error! Reference source not found.**.
* ANonce is a nonce that the Authenticator contributes for PTK generation. ANonce has the same value in message 1 and message 3.
* SNonce is a nonce from the Supplicant for PTK generation.
* P means the pairwise bit is set.
* The MIC is computed over the body of the EAPOL-Key frame (with the Key MIC field first zeroed before the computation) using the KCK defined in **Error! Reference source not found.** for PTK generation.
* RSNE represents the appropriate RSNEs.
* GTK[N] represents the GTK with its key identifier.

NOTE—While the MIC calculation is the same in each direction, the Key Ack bit is different in each direction. It is set in EAPOL-Key frames from the Authenticator and 0 in EAPOL-Key frames from the Supplicant. 4‑way handshake requests from the Supplicant have the Request bit equal to 1. The Authenticator and Supplicant need to check these bits to stop reflection attacks. It is important that message 1 contents not be used to update state, in particular the keys in use, until the data are validated with message 3.

* 4-way handshake message 1

Message 1 uses the following values for each of the EAPOL-Key frame fields:

Descriptor Type **=** N – see **Error! Reference source not found.**

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0

Key Type = 1 (Pairwise)

Reserved = 0

Install = 0

Key Ack = 1

Key MIC = 0

Secure = 0

Error = 0

Request = 0

Encrypted Key Data = 0

Reserved = 0 – unused by this protocol version

Key Length = Cipher-suite-specific; see **Error! Reference source not found.**

Key Replay Counter = *n* – to allow Authenticator or initiator STA to match the right message 2 from Supplicant or peer STA

Key Nonce = ANonce

EAPOL-Key IV = 0

Key RSC = 0

Key MIC = 0

Key Data Length = length of Key Data field in octets

Key Data = PMKID for the PMK being used during PTK generation

Processing for PTK generation is as follows:

The Authenticator sends message 1 to the Supplicant at the end of a successful IEEE 802.1X authentication, after (re)association completes for a STA that has authenticated with SAE or PSK authentication is negotiated, when a cached PMKSA is used, or after a STA requests a new key. On reception of message 1, the Supplicant determines whether the Key Replay Counter field value has been used before with the current PMKSA. If the Key Replay Counter field value is less than or equal to the current local value, the Supplicant discards the message. Otherwise, the Supplicant:

* Generates a new nonce SNonce.
* Derives PTK.
* Constructs message 2.
* 4-way handshake message 2

Message 2 uses the following values for each of the EAPOL-Key frame fields:

Descriptor Type **=** N – see **Error! Reference source not found.**

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0 – same as message 1

Key Type = 1 (Pairwise) – same as message 1

Reserved = 0

Install = 0

Key Ack = 0

Key MIC = 0 when using an AEAD cipher or 1 otherwise(11ai)

Secure = 0 – same as message 1

Error = 0 – same as message 1

Request = 0 – same as message 1

Encrypted Key Data = 0

Reserved = 0 – unused by this protocol version

Key Length = 0

Key Replay Counter = *n* – to let the Authenticator or initiator STA know to which message 1 this corresponds

Key Nonce = SNonce

EAPOL-Key IV = 0

Key RSC = 0

Key MIC = Not present when using an AEAD cipher; otherwise(11ai), MIC(KCK, EAPOL) – MIC computed over the body of this EAPOL-Key frame with the Key MIC field first initialized to 0

Key Data Length = length of Key Data field in octets

* Key Data =
* included RSNE – the sending STA’s RSNE for PTK generation or peer RSNE for the current operating band, and when this message 2 is part of a fast BSS transition initial mobility domain association or an association started through the FT protocol, the PMKR1Name calculated by the S1KH according to the procedures of **Error! Reference source not found.** is included in the PMKID field of the RSNE and the FTE and MDE are also included, or;
* The sending STA’s Multi-band element for PTK generation for a supported band other than the current operating band if dot11MultibandImplemented is true, or;
* The sending STA’s RSNE and Multi-band element(s) for generating a single PTK for all involved bands, if dot11MultibandImplemented is true and both the Authenticator and the Supplicant use the same MAC address in the current operating band and the other supported band(s); or;
* The sending STA’s RSNE and Multi-band element(s) for generating a different PTK for each involved band, if dot11MultibandImplemented is true and the Joint Multi-band RSNA subfield of the RSN capabilities field is 1 for both the Authenticator and the Supplicant, and either the Authenticator or the Supplicant uses different MAC addresses for different bands.

Processing for PTK generation is as follows:

The Supplicant sends message 2 to the Authenticator.

On reception of message 2, the Authenticator checks that the key replay counter corresponds to the outstanding message 1. If not, it silently discards the message. Otherwise, the Authenticator:

* Derives PTK.
* Verifies the message 2 MIC or AEAD decryption operation result(11ai).
* If the calculated MIC does not match the MIC that the Supplicant included in the EAPOL-Key frame or the AEAD decryption operation returns failure(11ai), the Authenticator silently discards message 2.
* If the MIC or AEAD decryption(11ai) is valid and this message 2 is part of a fast BSS transition initial mobility domain association or an association started through the FT protocol, the Authenticator checks that all fields of the RSNE other than the PMKID field bitwise matches the fields from the (Re)Association Request frame and that the FTE and MDE are the same as those provided in the AP’s (Re)Association Response frame. If the MIC or AEAD decryption(11ai) is valid and this message 2 is not part of a fast BSS transition initial mobility domain association and this message 2 is not part of an association started through the FT protocol, the Authenticator checks that the RSNE bitwise matches that from the (Re)Association Request frame.
* If these are not exactly the same, the Authenticator uses MLME-DEAUTHENTICATE.request primitive to terminate the association.
* If they do match bitwise, the Authenticator constructs message 3.
* If management frame protection is being negotiated, the AP initializes the SA Query Transaction Identifier to an implementation-specific non-negative integer value, valid for the current pairwise security association.
* 4-way handshake message 3

Message 3 uses the following values for each of the EAPOL-Key frame fields:

Descriptor Type **=** N – see **Error! Reference source not found.**

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0 – same as message 1

Key Type = 1 (Pairwise) – same as message 1

Reserved = 0

Install = 0/1 – For PTK generation, 0 only if the AP does not support key mapping keys, or if the STA has the No Pairwise bit (in the RSN Capabilities field) equal to 1and only the group key is used.

Key Ack = 1

Key MIC = 0 when using an AEAD cipher or 1 otherwise(11ai)

Secure = 1 (keys installed)

Error = 0 – same as message 1

Request = 0 – same as message 1

Encrypted Key Data = 1

Reserved = 0 – unused by this protocol version

Key Length = Cipher-suite-specific; see **Error! Reference source not found.**

Key Replay Counter = *n+1*

Key Nonce = ANonce – same as message 1

EAPOL-Key IV = 0 (Version 2) or random (Version 1)

Key RSC = For PTK generation, starting TSC or PN that the Authenticator’s STA uses in MPDUs protected by GTK.

Key MIC = Not present when using an(Ed) AEAD cipher; or otherwise(11ai), MIC(KCK, EAPOL) or MIC(SKCK, EAPOL) – MIC computed over the body of this EAPOL-Key frame with the Key MIC field first initialized to 0

Key Data Length = length of Key Data field in octets of included RSNEs and GTK

Key Data =

* For PTK generation for the current operating band, the AP’s Beacon/Probe Response frame’s RSNE for the current operating band, and, optionally, a second RSNE that is the Authenticator’s pairwise cipher suite assignment for the current operating band, and, if a group cipher has been negotiated, the GTK and the GTK’s key identifier (see **Error! Reference source not found.**) for the current operating band, and if management frame protection is negotiated, the IGTK KDE, and when this message 3 is part of a fast BSS transition initial mobility domain association or an association started through the FT protocol, the PMKR1Name calculated according to the procedures of **Error! Reference source not found.** in the PMKID field of the RSNE and the FTE with the same contents as in the (Re)Association Response frame, the MDE with the same contents as in the (Re)Association Response frame, the reassociation deadline timeout set to the minimum of dot11FTReassociationDeadline and the key lifetime in the TIE[ReassociationDeadline], and the PTK lifetime in the TIE[KeyLifetime]; or
* For PTK generation for a supported band other than the current operating band, the Authenticator’s Beacon/DMG Beacon/Announce/Probe Response/Information Response frame’s Multi-band element associated with the supported band, and optionally a second Multi-band element that indicates the Authenticator’s pairwise cipher suite assignment for the supported band, and, if group cipher for the supported band is negotiated, the Multi-band GTK KDE for the supported band if dot11MultibandImplemented is true, or;
* For generating a single PTK for all involved bands, the Authenticator’s Beacon/DMG Beacon/Announce/Probe Response/Information Response frame’s RSNE and Multi-band element(s), and optionally, additional RSNE and Multi-band element(s) that indicate the Authenticator’s assignment of one pairwise cipher suite for all involved bands; if a group cipher for all involved bands is negotiated, the GTK and the GTK’s key identifier for all involved bands, if dot11MultibandImplemented is true and both the Authenticator and the Supplicant use the same MAC address in the current operating band and the other supported band(s), or;
* For generating different PTKs for the current operating band and other supported band(s), the Authenticator’s Beacon/DMG Beacon/Announce/Probe Response/Information Response frame’s RSNE and Multi-band element(s), and optionally, additional RSNE and Multi-band elements that are the Authenticator’s pairwise cipher suite assignments for one or more involved bands; if group ciphers for the involved bands are negotiated, the Multi-band GTK KDEs for the involved bands, if dot11MultibandImplemented is true and the Joint Multi-band RSNA subfield is 1 for both the Authenticator and Supplicant, and either the Authenticator or the Supplicant uses different MAC addresses for different bands.

Processing for PTK generation is as follows:

If the Extended Key ID for Individually Addressed Frames subfield of the RSN Capabilities field is 1 for both the Authenticator and the Supplicant, then the Authenticator assigns a new Key ID for the PTKSA in the range 0 to 1 that is different from the Key ID assigned in the previous handshake and uses the MLME-SETKEYS.request primitive to install the new key to receive individually addressed MPDUs protected by the PTK with the assigned Key ID. Otherwise Key ID 0 is used and installation of the key is deferred until after message 4 has been received. The Authenticator sends message 3 to the Supplicant.

NOTE—If an existing PTK is still in effect, the Authenticator IEEE 802.11 MAC continues to transmit protected, individually addressed MPDUs (if any) using the existing key. With the installation of the new key for receive, the Authenticator is able to receive protected, individually addressed MPDUs using either the old key (if present) or the new key.

On reception of message 3, the Supplicant silently discards the message if the Key Replay Counter field value has already been used or if the ANonce value in message 3 differs from the ANonce value in message 1. The Supplicant also:

* Verifies the RSNE. If this message 3 is part of a fast BSS transition initial mobility domain association or an association started through the FT protocol, the Supplicant verifies that the PMKR1Name in the PMKID field of the RSNE is identical to the value it sent in message 2 and verifies that all other fields of the RSNE are identical to the fields in the RSNE present in the Beacon or Probe Response frames and verifies that the FTE and MDE are the same as in the (Re)Association Response frame. Otherwise, the Supplicant verifies that the RSNE is identical to that the STA received in the Beacon or Probe Response frame. If any of these verification steps indicates a mismatch, the STA shall disassociate or deauthenticate. If a second RSNE is provided in the message, the Supplicant uses the pairwise cipher suite specified in the second RSNE or deauthenticates.
* Verifies the message 3 MIC or AEAD decryption operation result(11ai). If the calculated MIC does not match the MIC that the Authenticator included in the EAPOL-Key frame or AEAD decryption operation returns failure(11ai), the Supplicant silently discards message 3.
* Updates the last-seen value of the Key Replay Counter field.
* If the Extended Key ID for Individually Addressed Frames subfield of the RSN Capabilities field is 1 for both the Authenticator and Supplicant: Uses the MLME-SETKEYS.request primitive to configure the IEEE 802.11 MAC to receive individually addressed MPDUs protected by the PTK with the assigned Key ID.
* Constructs message 4.
* Sends message 4 to the Authenticator.
* Uses the MLME-SETKEYS.request primitive to configure the IEEE 802.11 MAC to send and, if the receive key has not yet been installed, to receive individually addressed MPDUs protected by the PTK. The GTK is also configured by MLME-SETKEYS primitive.
* 4-way handshake message 4

Message 4 uses the following values for each of the EAPOL-Key frame fields:

Descriptor Type **=** N – see **Error! Reference source not found.**

Key Information:

Key Descriptor Version = 1 (ARC4 encryption with HMAC-MD5) or 2 (NIST AES key wrap with HMAC-SHA-1-128) or 3 (NIST AES key wrap with AES-128-CMAC), in all other cases 0 – same as message 1

Key Type = 1 (Pairwise) – same as message 1

Reserved = 0 - same as message 1

Install = 0

Key Ack = 0 – this is the last message

Key MIC = 0 when using an(Ed) AEAD cipher or 1 otherwise(11ai)

Secure = 1

Error = 0

Request = 0

Encrypted Key Data = 0

Reserved = 0 – unused by this protocol version

Key Length = 0

Key Replay Counter = *n+1*

Key Nonce = 0

EAPOL-Key IV = 0

Key RSC = 0

Key MIC = Not present when using an AEAD cipher; or otherwise(11ai), MIC(KCK, EAPOL) – MIC computed over the body of this EAPOL-Key frame with the Key MIC field first initialized to 0

Key Data Length = length of Key Data field in octets

Key Data = none required

Processing for PTK generation is as follows:

The Supplicant sends message 4 to the Authenticator. Note that when the 4-way handshake is first used, message 4 is sent in the clear.

On reception of message 4, the Authenticator verifies that the Key Replay Counter field value is one that it used on this 4-way handshake; if it is not, it silently discards the message. Otherwise:

* The Authenticator checks the MIC or AEAD decryption operation result(11ai). If the calculated MIC does not match the MIC that the Supplicant included in the EAPOL-Key frame or AEAD decryption operation returns failure(11ai), the Authenticator silently discards message 4.
* If the MIC is valid, the Authenticator uses the MLME-SETKEYS.request primitive to configure the IEEE 802.11 MAC to send and, if the receive key has not yet been installed, to receive protected, individually addressed MPDUs using for the new PTK.
* The Authenticator updates the Key Replay Counter field so that it uses a fresh value if a rekey becomes necessary.
* 4-way handshake implementation considerations

If the Authenticator does not receive a reply to its messages, it shall attempt dot11RSNAConfigPairwiseUpdateCount transmits of the message, plus a final timeout. The retransmit timeout value shall be 100 ms for the first timeout, half the listen interval for the second timeout, and the listen interval for subsequent timeouts. If there is no listen interval or the listen interval is zero, then 100 ms shall be used for all timeout values. If it still has not received a response after these retries, then for PTK generation the Authenticator should deauthenticate the STA.

For PTK generation, if the STA does not receive message 1 within the expected time interval (prior to IEEE 802.1X timeout), it should disassociate, deauthenticate, and try another AP/STA.

The Authenticator should ignore EAPOL-Key frames it is not expecting in reply to messages it has sent or EAPOL-Key frames in which the Ack bit is 1. This stops an attacker from sending the first message to the Supplicant who responds to the Authenticator.

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

The Supplicant uses the MLME-SETKEYS.request primitive to configure the temporal key from **Error! Reference source not found.** into its STA after sending message 4 to the Authenticator.

If the RSNE check for message 2 or message 3 fails, the SME should log an error and deauthenticate the peer.

2182.43 - 2191.59 delete 12.7.8 (PeerKey handshake) in its entirety

2288.22 delete “direct-link setup (DLS),”

2878.38 delete “and for DLS”

2950.52 delete entry for QB6 (Direct-link setup (DLS))

3071.24 (3074) to 3071.56 (3074) delete dot11DLSAllowedInQBSS and dot11DLSAllowed

3117.30 (3120) to 3119.32 (3122) delete items

dot11RSNAConfigSTKKeysImplemented

dot11RSNAConfigSTKCipher

dot11RSNAConfigSTKRekeyTime

dot11RSNAConfigSMKUpdateCount

dot11RSNAConfigSTKCipherSize

dot11RSNAConfigSMKLifetime

dot11RSNAConfigSMKReauthThreshold

dot11RSNAConfigNumberOfSTKSAReplayCounters

dot11RSNAPairwiseSTKSelected

dot11RSNASMKHandshakeFailures

3475.56 (3478) to 3476.7 (3479) delete dot11NonAPStationAuthDls

3772.40 (3775) delete "Such a situation would support, for example, 40 MHz bandwidth DLS traffic among associated STAs, but only 20 MHz bandwidth traffic between STAs and the AP."

3791.53 (3794) change “attributes are” to attribute is”, then, at 3791.54 delete “— dot11NonAPStationAuthDls is to authorize a non-AP STA to use DLS”

Note to the editor: After deleting as per the above instructions, please check that DLS, STSL, STK, SMK, SMKSA, STKSA, SKCK, SKEK, STA\_I, STA\_P, do not appear. If they do, please add an editors note tagged "#removestsl".