IEEE P802.11
Wireless LANs

|  |
| --- |
| Resolution of A Few Security Comments |
| Date: 2021-04-21 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins  | HPE |  |  |  |
|  |  |  |  |  |

Abstract

This submission proposes resolution to CIDs 116, 117, 156, 165, 175, 224, 331, 355, 369, 588, 589, 590, and 595.

**CID 116 (needs more work)**

*Comment*: “SAE: ‘<verify X>. If not, <do Y>. If so, <do Z>’ construction can be ambiguous since it is not always clear what ‘if so’ is referring to (something in ‘verify X’ vs. ‘do Y’).”

*Proposed Change*: “Replace

‘If Sync is not greater than dot11RSNASAESync, the protocol instance shall verify that the finite cyclic group is the same as the previously received SAE Commit message. If not, the frame shall be silently discarded. If so, the protocol instance shall increment Sync, increment Sc, and transmit its SAE Commit message and its SAE Confirm message with the new Sc value.’

with

‘If Sync is not greater than dot11RSNASAESync, the protocol instance shall verify that the finite cyclic group is the same as the previously received SAE Commit message. If not, the frame shall be silently discarded. Otherwise, the protocol instance shall increment Sync, increment Sc, and transmit its SAE Commit message and its SAE Confirm message with the new Sc value.’

*Discussion*: That sounds much more clear.

*Resolution*: Accept.

**CID 156 and 331**

*156 Comment:* “It seems that the ‘method’ is called ‘hash-to-curve’ (per 12.4.4.2.3) or described as ‘direct hash[ing]’, but ‘SAE hash-to-element’ is the name of the capability to use this method. So, places like P2546.38 are using the inappropriate term.

*156 Proposed Change:* “Change ‘hash-to-element’ to ‘hash-to-curve’. Same change at P876.47.”

*331 Comment*: “Is it ‘hash-to-element’ or ‘hash-to-curve’? The former seems more popular”

*331 Proposed Change*: “Change ‘ash-to-curve’ to ‘ash-to-element’ (5x on referenced page)”

*Discussion*: It should be hash-to-element everywhere. And this change is a bit more involved….

*Resolution*: Change “hash-to-curve” to “hash-to-element” in the following:

* the title of 12.4.4.2.3
* the 4 references in the body of 12.4.4.2.3

**CID 165**

*Comment*: “’The following types of security associations are supported by an RSNA STA:’ is missing TPKSA”

*Proposed Change*: “Add a bullet ‘TPKSA: A result of a successful TPK handshake’ at the end of the list”

*Discussion*: Seems correct. An obvious oversight.

*Resolution*: Accept

**CID 175 (to Mark Hamilton)**

*Comment*: “There are two references to ‘a non-NULL dot11RSNAConfigPasswordIdentifier’ but this term is not defined here or in the MIB”

*Proposed Change*: “Change each instance to ‘a dot11RSNAConfigPasswordIdentifier that does not have a zero length’ and in C.3 show a DEFVAL of ‘{ }’ for dot11RSNAConfigPasswordIdentifier”

*Discussion*: What string has a zero length? Where is that defined? At some point we have to have a common understanding of certain words. A non-NULL value for a string is one of those understandings.

*Resolution*: Reject, the intent is obvious. There is no ambiguity and nothing to change.

**CID 224 (to Mark Rison)**

*Comment*: “There are no constraints on the cipher to use with TDLS, other than not using WEP or TKIP. Some recommendations should be given, specifically that it should be at least as strong as the cipher used with the AP”

*Proposed Change*: “After the sentence starting ‘The pairwise cipher suite list field indicating ‘ in 12.7.8.4.2 TPK handshake message 1 add ‘A pairwise cipher suite of key size smaller than that used on the connection between the STA and the AP should not be used.’ and after the sentence starting ‘If none of the pairwise cipher suites are acceptable" add ‘The TDLS responder STA should ignore any pairwise cipher suites of key size smaller than that used on the connection between the STA and the AP, and should reject the TDLS Setup Request frame with status code STATUS\_INVALID\_PAIRWISE\_CIPHER if all the pairwise cipher suites are such.’. In 12.7.8.4.3 TPK handshake message 2 after the sentence starting ‘Include a pairwise cipher suite’ add ‘A pairwise cipher suite of key size smaller than that used on the connection between the STA and the AP should not be used.’“

*Discussion*: There is nothing inhererntly wrong with using a smaller key. It depends on what the TDLS conversation is going to be consisting of. In fact, one might argue the other way around—a longer key protected by an exchange using a smaller key—but that’s if one wants to argue the point which it does not seem useful to do. If we wanted to start making recommendations on ciphers it should involve how the key was generated—i.e. don’t use EAP-TLS with a TLS version less than 1.2 and do (EC)DHE for key generation, make sure you use the appropriate hash algorithms, etc—but for people that get hung up on that we have the Suite B ciphers.

*Resolution*: Reject, aside from Suite B ciphersuites and AKMs, 802.11 does not restrict what ciphers can be used and we shouldn’t start with TDLS.

**CID 355**

*Comment*: “’support for the SAE hash-to-element’ -- for the SAE H2E what? Also in 12.4.4.3.2”

*Proposed Change*: “Add ‘technique’ to the end of the cited text in both subclauses mentioned”

*Discussion*: Other parts of the standard discuss “the SAE hash-to-element method” and we wouldn’t want to introduce any *inconsistencies* in wording so “technique” is not appropriate.

*Resolution*:

*Instruct the editor to modify sections 12.4.4.2.2 and 12.4.4.3.2 as indicated:*

**12.4.4.2.2 Generation of the password element with ECC groups by looping**

If the AP does not indicate support for the SAE hash-to-element method in its Extended RSN Capabilities field or the SAE initiator does not set the status code to SAE\_HASH\_TO\_ELEMENT in its SAE Commit message, the password element of an ECC group (PWE) shall be generated in the following random hunt-and-peck fashion.

**12.4.4.3.2 Generation of the password element with FFC groups by looping**

If the AP does not indicate support for the SAE hash-to-element method in its Extended RSN Capabilities field or the SAE initiator does not set the status code to SAE\_HASH\_TO\_ELEMENT in its SAE Commit message, the password element of an FFC group (PWE) shall be generated in the following random hunt-and-peck fashion.

**CID 369**

*Comment*: “’silently drop’ would imply that all the other unadorned ‘drop’[sic]s are to be done noisily”

*Proposed Change*: “Delete ‘silently’ (2x)”

*Discussion*: It implies no such thing. In a protocol when a message is “silently dropped” it means it is discarded without an externally visible notification of dropping. This is well known to people who work on standards and develop standards and implement standards.

*Resolution*: Reject. In a protocol when a message is “silently dropped” it means it is discarded without an externally visible notification of dropping.

**CID 588**

*Comment*: “In SAE (also FILS unless bound to rMSK lifetime), the maximum PMK lifetime (dot11RSNAConfigPMKLifetime) is not communicated to the peer. This can cause inefficiencies if STA attempts use of a cached PMK that has expired”

*Proposed Change*: “Provide a (protected) way for the PMK lifetime to be indicated, at least for communication of AP's lifetime to the STA - e.g. using TIE with Key Lifetime”

*Discussion*: This is not an SAE or FILS issue, it affects 802.1X as well. The PMKSA database on a STA (including an AP) is governed solely by the STA. It can delete a PMKSA any time it wants for any reason, including no reason whatsoever. A STA is attempting to use a cached PMK is opportunistically hoping. There are no guarantees and if an AP communicated a PMK lifetime to the STA somehow it would still be under no obligation to retain that PMK in its PMKSA database for the entire duration.

*Resolution*: Reject, while it may be possible to provide a uniform method of PMK lifetime notification, it would be optional, and there would be no guarantee the PMK will not be deleted beforehand anyway so the utility of this seems to not be worth the effort.

**CID 589**

*Comment*: “Key Data field in M1 and M2 ‘need not be encrypted’”

*Proposed Change*: “be more precise about conditions under which it can be, or must be, encrypted”

*Discussion*: I’m not sure why the text was so imprecise but I think it’s safe to get rid of them.

*Resolution*:

Instruct the editor to modify section 12.7.2 as indicated:

**12.7.2 EAPOL-Key frames**

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 Key Data field to indicate a PMKID when a cached PMKSA 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.
* 4-way handshake message 2 is an EAPOL-Key frame with the Key Type subfield equal to 1. The Key Data field shall contain an RSNE and may contain an RSNXE.

**CID 590**

*Comment*: “AKM 20 is PSK SHA-384 per Table 9-151. However per 12.7.1.3, it seems PMK\_bits is 256. Should it be 384?”

*Proposed Change*: “If intent is for PMK\_bits to be 384 for this AKM, add the AKM to the corresponding sentence in 12.7.1.3. (Also check PTK length in 12.7.1.6.5)”

*Discussion*: Yes, the intent is obviously for PMK\_bits to be 384. The existing text is very fragile because there are so many places that need updating when new AKMs are added. Obviously 20 was added without dotting all the Is and crossing all the Ts.

*Resolution*: Note to commentor: 12.7.1.6.5 does not need resolution because we do not have ciphers which use 384 bit keys.

*Instruct the editor to modify sections 12.7.1.2 and 12.7.1.3 as indicated:*

**12.7.1.2 PRF**

When the negotiated AKM is 00-0F-AC:12 or 00-0F-AC:20, the KDF specified in 12.7.1.6.2 (Key derivation function (KDF)) shall be used instead of the PRF construction defined here. In this case, A is used as the KDF label and B as the KDF Context, and the PRF function is defined as follows:

 PRF-704(K, A, B) = KDF-SHA-384-704(K, A, B)

**12.7.1.3 Pairwise Key Hierarchy**

Except when preauthentication or FILS authentication is used, the pairwise key hierarchy utilizes PRF-384, PRF-512, or PRF-704 to derive session specific keys from a PMK, as depicted in Figure 12-30 (Pairwise key hierarchy). When using AKM suite selector 00-0F-AC:12, 00-0F-AC:15, or 00-0F-AC:20, the length of the PMK, PMK\_bits, shall be 384 bits. When using AKM suite selectors for which the Authentication type column indicates FT authentication (see Table 9-151 (AKM suite selectors)), the FT key hierarchy is used to derive session specific keys from an MPMK as defined in 12.7.1.6 (FT key hierarchy). With all other AKM suite selectors, the length of the PMK, PMK\_bits, shall be 256 bits. The pairwise key hierarchy takes a PMK and generates a PTK. The PTK is partitioned into KCK, KEK, and a temporal key, which is used by the MAC to protect individually addressed communication between the Authenticator’s and Supplicant’s respective STAs. PTKs are used between a single Supplicant and a single Authenticator.

**CID 595 and 117**

*595 Comment*: “It is unclear whether the first Send Confirm value should be 0 or 1. 12.4.8.6.3 says Sc is reset and confirm is sent. 12.4.5.5 says message is constructed using current Sc (i.e. 0), but Figure 12-4 shows Inc(Sc) in transition from Nothing to Confirmed.”

*595 Proposed Change*: “Clarify language and, to avoid making existing implementations non-compliant, allow either”

*117 Comment*: “Textual description of the SAE Nothing-to-Confirmed transmission does not match the state machine steps shown in Figure 12-4. The figure indicates that Sc is incremented (to 1) before constructing the Confirm message while the text seems to imply that Sc is only set to zero without incrementing. Figure 12-4 shows this as follows: zero(Sync),zero(Sc),zero(Rc),inc(Sc),1(0),2,set(t0). The text should match that.”

*117 Proposed Change*: “Add "increment Sc" before construction of Commit and Confirm messages by replacing "If the group is supported, the protocol instance shall zero the Sc and Rc counters and it shall generate the PWE and the secret values according to 12.4.5.2 (PWE and secret generation). It shall then process the received SAE Commit message (see 12.4.5.4 (Processing of a peer's SAE Commit message)). If validation of the received SAE Commit message fails, the protocol instance shall send a Del event to the parent process; otherwise, it shall construct and transmit an SAE Commit message (see 12.4.5.3 (Construction of an SAE Commit message)) followed by an SAE Confirm message (see 12.4.5.5 (Construction of an SAE Confirm message))." With "If the group is supported, the protocol instance shall zero the Sc and Rc counters and it shall generate the PWE and the secret values according to 12.4.5.2 (PWE and secret generation). It shall then process the received SAE Commit message (see 12.4.5.4 (Processing of a peer's SAE Commit message)). If validation of the received SAE Commit message fails, the protocol instance shall send a Del event to the parent process; otherwise, it shall increment Sc, construct and transmit an SAE Commit message (see 12.4.5.3 (Construction of an SAE Commit message)) followed by an SAE Confirm message (see 12.4.5.5 (Construction of an SAE Confirm message))."

*Discussion*: There is a discrepancy between the text and the graphic in figure 12-4. The intent is that the first Confirm message have a Sc of 1, state machine resynchronization that results in a new Confirm message being generated will increment Sc. Since SAE is defined not as a client-server protocol, there is no notion of AP and non-AP STA, everyone’s a peer so the behavior of a peer is identical no matter whether you initiated or responded (or both initiated simultaneously).

The next needs to specify that Sc is incremented in the case where a peer is a responder so that it matches the graphic in 12-4. This will have no impact on interoperability because handling just says that the initial value is just stored and during resynchornization of the state machine the received value has to be greater than the previous received value, which will still be true.

*Resolution*:

*Instruct the editor to modify section 12.4.8.6.3 as indicated:*

**12.4.8.6.3 Protocol instance behavior—Nothing state**

Upon receipt of a Com event, the protocol instance shall check the Status of the Authentication frame. If the Status code is not SUCCESS, the frame shall be silently discarded and a Del event shall be sent to the parent process. Otherwise, the frame shall be processed by first checking whether a password identifier is present. If so and there is no password associated with that identifier, BadID shall be set and the protocol instance shall construct and transmit an Authentication frame with Status Code set to UNKNOWN\_PASSWORD\_IDENTIFIER. If there is no password identifier present or if a password is associated with that identifier, the frame shall be processed by next checking the finite cyclic group field to see if the requested group is supported. If not, BadGrp shall be set and the protocol instance shall construct and transmit an Authentication frame with Status code UNSUPPORTED\_FINITE\_CYCLIC\_GROUP indicating rejection with the finite cyclic group field set to the rejected group, and shall send the parent process a Del event. If the group is supported, the protocol instance shall zero the Sc and Rc counters and it shall generate the PWE and the secret values according to 12.4.5.2 (PWE and secret generation). It shall then process the received SAE Commit message (see 12.4.5.4 (Processing of a peer’s SAE Commit message)). If validation of the received SAE Commit message fails, the protocol instance shall send a Del event to the parent process; otherwise, it shall construct and transmit an SAE Commit message (see 12.4.5.3 (Construction of an SAE Commit message)), increment Sc, and construct and transmit an SAE Confirm message (see 12.4.5.5 (Construction of an SAE Confirm message)). The Sync counter shall be set to 0 and the t0 (retransmission) timer shall be set. The protocol instance transitions to Confirmed state.

**References:**