IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SAE anti-clogging token | | | | |
| Date: 2020-01-13 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Jouni Malinen | Qualcomm, Inc. |  |  | jouni@qca.qualcomm.com |
|  |  |  |  |  |

Abstract

This document discusses the SAE anti-clogging token field design and complexities in P802.11-REVmd/D3.0 and proposes a cleaner design that does not require as complex parsing rules. The proposed design is backwards compatible with the current standard IEEE Std 802.11-2016.

This document proposes resolutions for initial SA ballot CIDs 4133 and 4726.

r1:

- require H2E to be used whenever Password Identifier is used

- add proposed resolution to applicable CIDs

- editorial cleanup

r2:

- updates based on TGmd Jan 13 PM1 discussion (this does not include generic changes to use “hash-to-element method” throughout the standard)

**Discussion**

REVmd has added new fields and elements to SAE Authentication frames: Password Identifier element and Rejected Groups element. These are conditionally included at the end of the frame. Since these are preceded by conditionally included variable length (non-IE) field (Anti-Clogging Token field), parsing of the frame has become significantly more complex. This undesired complexity has resulted in need for 12.4.7.4 describing rules for how to parse the frame. This complexity could be avoided by replacing Anti-Clogging Token field with a new information element, say Anti-Clogging Token Container element, that would encapsulate the Anti-Clogging Token field and by mandating this new element to be used instead of the field whenever using SAE with Password Identifiers or H2E (i.e., the cases that use the new Password Identifier and Rejected Groups elements). This would result in all the non-IE fields being present and all the information elements being at the end of the frame which would simplify parsing significantly.

To cover all different combinations of how SAE Authentication frames are used, use of H2E needs to be mandated whenever password identifiers are used. This removes an ambiguous corner case where a STA requesting use of anti-clogging token had dropped need for anti-clogging mechanism before the peer STA sends its Authentication frame with the previously requested anti-clogging token. Making H2E mandatory for password identifiers is acceptable at this point in the P802.11-REVmd process since both of these mechanisms have been added during the REVmd work (i.e., neither of them has been included in any approved standard).

In addition to simplifying the Anti-Clogging Token field parsing, the proposed changes are addressing couple of errors in the description of when this field is present. The introduction of hash-to-element method for deriving PWE introduced these in Table 9-43.

**CID 4133**

Comment:

REVmd has added new fields and elements to SAE Authentication frames: Password Identifier element and Rejected Groups element. These are conditionally included at the end of the frame. Since these are preceded by conditionally included variable length (non-IE) field (Anti-Clogging Token field), parsing of the frame has become significantly more complex. This undesired complexity has resulted in need for 12.4.7.4 describing rules for how to parse the frame. This complexity could be avoided by replacing Anti-Clogging Token field with a new information element, say Anti-Clogging Token Container element, that would encapsulate the Anti-Clogging Token field and by mandating this new element to be used instead of the field whenever using SAE with Password Identifiers or H2E (i.e., the cases that use the new Password Identifier and Rejected Groups elements). This would result in all the non-IE fields being present and all the information elements being at the end of the frame which would simplify parsing significantly.

Proposed Change:

Incorporate changes from the "Proposed changes" section of https://mentor.ieee.org/802.11/dcn/19/11-19-2154-00-000m-sae-anti-clogging-token.docx

Proposed Resolution:

REVISED. Incorporate changes from the "Proposed changes" section of https://mentor.ieee.org/802.11/dcn/19/11-19-2154-01-000m-sae-anti-clogging-token.docx.

**CID 4726**

Comment:

The approach used does not work in all cases, because the Scalar and Anti-Clogging Token fields are not elements, and hence could contain octet sequences that make them look like FFE, Finite Cyclic Group or Vendor Specific elements

Proposed Change:

Make the Scalar and Anti-Clogging fields self-identifying by making them elements, not fixed fields

Proposed Resolution:

REVISED. Incorporate changes from the "Proposed changes" section of <https://mentor.ieee.org/802.11/dcn/19/11-19-2154-01-000m-sae-anti-clogging-token.docx>. This encapsulates Anti-Clogging field within an element in case hash-to-element mechanism is used. Scalar field does not need such change since its presence in the Authentication frame is unambiguously defined and Anti-Clogging field cannot be modified in the case of hunting-and-pecking without breaking compatibility with deployed implementations.

**Proposed changes**

**9.3.3.11 Authentication frame format**

*Add new row to Table 9-42 and modify Table 9-43 as shown below.*

The frame body of an Authentication frame contains the information shown in Table 9-42 (Authentication frame body). FT authentication is used when FT support is advertised by the AP and dot11FastBSSTransitionActivated is true in the STA. SAE authentication is used when dot11MeshActiveAuthenticationProtocol is sae (1). FILS authentication is used if support for FILS authentication is advertised by the AP and dot11FILSActivated is true in the STA.(11ai)

**Table 9-42—Authentication frame body**

|  |  |  |
| --- | --- | --- |
| **Order** | **Information** | **Notes** |
| 1 | Authentication algorithm number |  |
| 2 | Authentication transaction sequence number |  |
| 3 | Status code | The status code information is reserved in certain Authentication frames as defined in Table 9-43 (Presence of fields and elements in Authentication frames). |
| 4(11ai) | Finite Cyclic Group | An unsigned integer indicating a finite cyclic group as described in 9.4.1.42 (Finite Cyclic Group field)(#2437). This is present only in certain Authentication frames as defined in Table 9-43 (Presence of fields and elements in Authentication frames). |
| 5(11ai) | Anti-Clogging Token | A random bit string used for anti-clogging purposes as described in 12.4.6 (Anti-clogging tokens). This is present only in certain Authentication frames as defined in Table 9-43 (Presence of fields and elements in Authentication frames). |
| … | | |
| 22(M41) | Password Identifier(M41) | The Password Identifier element is optional present in certain Authentication frames as defined in Table 9-43 (Presence of fields and elements in Authentication frames)(M41). |
| 23(M137) | Rejected Groups(M137) | The Rejected Groups element is present only in certain Authentication frames as defined in Table 9-43 (Presence of fields and elements in Authentication frames). (M137) |
| 24 | Anti-Clogging Token Container | The Anti-Clogging Token Container element is present only in certain Authentication frames as defined in Table 9-43 (Presence of fields and elements in Authentication frames). |
| Last | Vendor Specific | One or more vendor-specific elements are optionally present. These elements follow all other elements. |

**Table 9-43—Presence of fields and elements in Authentication frames**

|  |  |  |  |
| --- | --- | --- | --- |
| **Authentication algorithm** | **Authentication transaction sequence number** | **Status code** | (#2528)**Presence of fields and elements from order 4 onwards** |
| … | | | |
| SAE | 1 | Any | (Ed)The Scalar field(#2531) is present if the Status Code field is zero or 126(M137).  (#2471)(Ed)The FFE field(#2531) is present if the Status Code field is zero or 126(M137).  (#2530)(Ed)When the hunting-and-pecking method is used to derive the PWE, the Anti-Clogging Token field(#2534) is present if the Status Code field is 76 (M137) or if the Authentication frame is in response to a previous rejection with the Status Code field equal to 76 (M137).  (Ed)The Finite Cyclic Group field(#2531) is present if the Status Code field is zero, 76, 77 or 126(M137).(M104)  (#2530)(M41)(Ed)The Password Identifier element is optionally present if the Status Code field is zero, 123 or 126(M137)(Ed). The Rejected Groups element is conditionally present if the Status Code field is 126.(M137)  When the hash-to-element method is used to derive the PWE, the Anti-Clogging Token Container element is present if the Status Code field is 76 or if the Authentication frame is in response to a previous rejection with the Status Code field equal to 76. |
| … | | | |

**9.4.1.38 Anti-Clogging Token field**

*No changes in 9.4.1.38 – only included for context.*

The Anti-Clogging Token field is used with SAE authentication for denial-of-service protection as specified in 12.4 (Authentication using a password). See Figure 9-126 (Anti-Clogging Token field format(#2607)).

|  |
| --- |
| Anti-Clogging Token |

Octets: variable

**Figure 9-126—Anti-Clogging Token field format**(#2607)

**9.4.2 Elements**

**9.4.2.1 General**

*Add a new row in Table 9-94 as shown below.*

**Table 9-94—Element IDs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Element ID** | **Element ID Extension** | **Extensible** | **Fragmentable** |
| … | | | | |
| (M137)Rejected Groups (see 9.4.2.246 (Rejected Groups element(M137))) | 255 | 92 | No | No |
| Anti-Clogging Token Container | 255 | *<ANA>* | No | No |
| Reserved(#1100) | 255 | 93–255(#2693)(#2215)(#1283)(M40) |  |  |
| NOTE— See 10.28.6 (Element parsing) on the parsing of elements.(#283) | | | | |

**9.4.2.x Anti-Clogging Token Container element**

*Add a new subclause at the end of 9.4.2 (after all the existing 9.4.2.x subclauses).*

The Anti-Clogging Token Container element is used to carry an Anti-Clogging Token field in contexts where an information element is needed. The format of the Anti-Clogging Token Container element is shown in Figure 9-y (Anti-Clogging Token Container element format).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Element ID | Length | Element ID Extension | Anti-Clogging Token |
| Octets: | 1 | 1 | 1 | variable |

**Figure 9-y—Anti-Clogging Token Container element format**

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1 (General).

The Anti-Clogging Token field is defined in 9.4.1.38 (Anti-Clogging Token field).

**12.4.4.2.2 Generation of the password element with ECC groups by looping**(M137)

*Modify 12.4.4.2.2 as shown below.*

(M137)If the AP does not indicate support for the SAE hash-to-element in its Extended RSN Capabilities field or the SAE initiator does not set the Status Code field 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.

NOTE—This method cannot be used with a password identifier.

The password and a counter, represented as a single octet and initially set to 1, are used with the peer identities to generate a password seed. The password seed shall then be stretched using the key derivation function (KDF) from 12.7.1.6.2 (Key derivation function (KDF)) to a length equal to the bit length of the prime number, p, from the elliptic curve domain parameters with the Label being the string “SAE Hunting and Pecking” and with the Context being the prime number. If the resulting password value is greater than or equal to the prime number, the counter shall be incremented, a new password seed shall be derived and the hunting-and-pecking shall continue. Otherwise, it shall be used as the x-coordinate of a candidate point (x, y) on the curve satisfying the curve equation, if such a point exists. If no solution exists, the counter shall be incremented, a new password-seed shall be derived and the hunting-and-pecking shall continue. Otherwise, there are two possible solutions: (x, y) and (x, p – y). The password seed shall be used to determine which one to use: if the (M101) least significant bit (LSB) of the password seed is equal to that of y, the PWE shall be set to (x, y); otherwise, it shall be set to (x, p – y).

In order to minimize the possibility of side-channel attacks that attempt to determine the number of interactions of the “hunting-and-pecking” loop required for a given <password, STA-A-MAC, STA-B-MAC> tuple, implementations should perform at least k iterations regardless of whether PWE is discovered or not. The value k may be set to any non-negative value and should be set to a sufficiently large number to effectively guarantee the discovery of PWE in less than k iterations. If PWE is discovered in less than k iterations a random “password” can be used in subsequent iterations to further obfuscate the true cost of discovering PWE.

NOTE—The probability that one requires more than n iterations of the “hunting and pecking” loop to find PWE is roughly (r/2p)n, which rapidly approaches 0 as n increases.

Algorithmically this process is described as follows:

found = 0;

counter = 1

Length = len(p )

base = password (M41)

do {

pwd-seed = H(MAX(STA-A-MAC, STA-B-MAC) || MIN(STA-A-MAC, STA-B-MAC),

base || counter )

pwd -value = KDF-Hash-Length (pwd-seed, “SAE Hunting and Pecking”, p )

if (pwd-value < p )

then

if (pwd-value3 + a × pwd-value + b ) is a quadratic residue modulo p

then

if (found ==0)

then

x = pwd-value

save = pwd-seed

found = 1

base = a new random number

fi

fi

fi

counter = counter + 1

} while ((counter <= k ) or (found ==0))

y = sqrt(x3 + ax + b ) mod p

if (LSB(save ) == LSB(y ))

then

PWE = (x , y )

else

PWE = (x , p – y )

fi

where

KDF-Hash-*Length* is the key derivation function defined in 12.7.1.6.2 (Key derivation function (KDF))

using the hash algorithm identified by the AKM suite selector (see Table 9-151 (AKM suite selectors))

len() returns the length of its argument in bits

(M41)

**12.4.4.2.3 Hash-to-curve generation of the password element with ECC groups**(M137)

*Modify 12.4.4.2.3 as shown below.*

An SAE peer, e.g. a mesh STA or an AP, indicates support for direct hashing to obtain an ECC password element by setting the SAE hash-to-element bit in the Extended RSN Capabilities field in all Beacon and Probe Response frames. A STA that uses a password identifier shall use the hash-to-curve method. An SAE initiator that has identified a peer that supports this technique (through receipt of Beacon or Probe Response frames) shall derive a secret element, PT, according to the following technique and indicate this by setting the status code in the SAE Commit message to SAE\_HASH\_TO\_ELEMENT. An SAE initiator shall not indicate support for this form of element derivation unless its peer has already signalled support for this method. If an SAE Commit message is received with status code equal to SAE\_HASH\_TO\_ELEMENT the peer shall generate the PWE using the following technique and reply with its own SAE Commit message with status code equal to SAE\_HASH\_TO\_ELEMENT.

The direct hashing technique to derive an element of an ECC group is the Simplified Shallue-Woestijne-Ulas (SSWU) deterministic hash-to-curve method. The SSWU method is called twice with two distinct functions to produce two points on the elliptic curve. The two points are summed to create a secret element PT.

This method works for all Weierstrass elliptic curves whose constants a and b are both not equal to zero. Other curves shall not be used with the hash-to-curve method.

The hash-to-curve method uses HKDF (IETF RFC 5869) with the hash algorithm taken from Table 12-1 (Hash algorithm based on length of prime(M137)) based on the length of the prime of the ECC group to perform both functions. First HKDF-Extract is passed a salt in the form of the SSID for which the password is to be used, the password, and optionally a password identifier to produce and intermediary password seed. The resulting seed is passed to HKDF-Expand to produce two distinct strings using different labels. Both values are reduced modulo p, the prime defining the curve, and then passed to SSWU to produce distinct points, P1 and P2, whose sum is PT.

This secret PT is stored until needed to generate a session-specific PWE (see 12.4.5.2 (PWE and secret generation)).

**12.4.4.3.2 Generation of the password element with FFC groups by looping**(M137)

*Modify 12.4.4.3.2 as shown below.*

If the AP does not indicate support for the SAE hash-to-element in its Extended RSN Capabilities field or the SAE initiator does not set Status Code field 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.(M137)

NOTE—This method cannot be used with a password identifier.

The password and a counter, represented as a single octet and initially set to 1, are used with the two peer identities to generate a password seed. The password seed shall then be stretched using the key derivation function (KDF) from 12.7.1.6.2 (Key derivation function (KDF)) to a length equal to the bit length of the prime number, p , from the group domain parameters with the Label being the string “SAE Hunting and Pecking” and the Content being the prime number. If the resulting password value is greater than or equal to the prime number, the counter shall be incremented, a new password seed shall be derived, and the hunting-and-pecking shall continue. Otherwise, it shall be raised to the power (p – 1) / r (where p is the prime number and r is the order) modulo the prime number to produce a candidate PWE . If the candidate PWE is greater than 1, the candidate PWE becomes the PWE; otherwise, the counter shall be incremented, a new password seed shall be derived, and the hunting-and-pecking shall continue.

Algorithmically this process is described as follows:

found = 0;

counter = 1

Length = len( p)

do {

pwd-seed = H(MAX(STA-A-MAC, STA-B-MAC) || MIN(STA-A-MAC, STA-B-MAC),

password || counter)(M41)

pwd-value = KDF-Hash- Length( pwd-seed, “SAE Hunting and Pecking”, p)

if ( pwd-value < p)

then

PWE = pwd-value(p-1)/r mod p

if (PWE > 1)

then

found = 1

fi

fi

counter = counter + 1

} while (found==0)

where

KDF-Hash-Length is the key derivation function defined in 12.7.1.6.2 (Key derivation function (KDF))

using the hash algorithm identified by the AKM suite selector (see Table 9-151 (AKM suite selectors))

len() returns the length of its argument in bits

(M41)

**12.4.4.3.3 Direct Generation of the password element with FFC groups**(M137)

*Modify 12.4.4.3.3 as shown below.*

An SAE peer indicates support for direct hashing to obtain the FFC password element by setting the SAE hash-to-element bit in the Extended RSN Capabilities field in all Beacon and Probe Response frames. A STA that uses a password identifier shall use the direct hashing technique. An SAE initiator that has identified a peer that supports the following technique (through receipt of Beacon or Probe Response frames) shall derive PT according to the following technique and indicate this by setting the status code in the SAE Commit message to SAE\_HASH\_TO\_ELEMENT. An SAE initiator shall not indicate support for this form of PWE derivation unless its peer has already signalled support. If an SAE Commit message is received with status code equal to SAE\_HASH\_TO\_ELEMENT the peer shall generate the PWE using the following technique and reply with its own SAE Commit message with status code set to SAE\_HASH\_TO\_ELEMENT.

The direct hashing technique uses HKDF (IETF RFC 5869) with the hash algorithm taken from Table 12-1 (Hash algorithm based on length of prime(M137)) based on the length of the prime of the FFC group.

To perform the direct hashing technique, HKDF (IETF RFC 5869) is passed a salt in the form of the SSID for which the password is to be used, the password, optionally a password identifier, as an input key, a constant label “SAE Hash to Element”, and the length of the prime to produce a password value. The resulting password value shall be reduced into a range such that 1 < pwd-value < p. Then, it shall be raised to the power (p-1) / q and reduced modulo p (where p is the prime number and q is the order). This will ensure PT is a generator of order either 1 (if PT = 1) or q (for all other values). The probability of PT taking the value 1 is negligible.

This secret PT is stored until needed to generate a session-specific PWE.

**12.4.7.4 Encoding and decoding of SAE Commit messages**

*Modify 12.4.7.4 as shown below.*

An SAE Commit message shall be encoded as an Authentication frame with an Authentication Algorithm Number field set to 3, a Transaction Sequence Number of 1 and a Status Code of SUCCESS or SAE\_HASH\_TO\_ELEMENT. Status codes not equal to SUCCESS or SAE\_HASH\_TO\_ELEMENT indicate a rejection of a peer’s SAE Commit message and are described in 12.4.7.6 (Status codes).(M137)

An SAE Commit message shall consist of a Finite Cyclic Group field (9.4.1.42 (Finite Cyclic Group field)) indicating a group, a Scalar field (9.4.1.39 (Scalar field)) containing the scalar, and an FFE field containing the element (9.4.1.40 (FFE field(#2302))). If the SAE Commit message is in response to an Anti-Clogging Token field(#2534) request (see 12.4.7.6 (Status codes)), the Anti-Clogging Token field(#2534) is present (see 9.4.1.38 (Anti-Clogging Token field)). When the PWE is derived using the hash-to-element method, the Anti-Clogging Token field is encapsulated in an Anti-Clogging Token Container element; otherwise, the Anti-Clogging Token field is included in the frame outside of an element as described in Table 9-43 (Presence of fields and elements in Authentication frames). If a password identifier is used in generation of the password element (PWE) the Password identifier element shall be present and the identifier shall be encoded as a UTF-8 string in the Identifier portion of the element (see 9.4.2.216 (Password Identifier element(M41)(M101))).(M41)(M137) If an SAE Commit message with status code equal to SAE\_HASH\_TO\_ELEMENT is being sent in response to rejection of a previous SAE Commit message with status set to UNSUPPORTED\_FINITE\_CYCLIC\_GROUP, the group that was rejected shall be appended, after the rejected groups from previous attempts if applicable, to the Rejected Groups field of the Rejected Groups element. Each rejected group shall be represented as an unsigned 16-bit integer using the bit ordering conventions of 9.2.2 (Conventions).

When transmitting an SAE Commit message, the scalar and element shall be converted to octet strings and placed in the Scalar field and FFE field, respectively. The scalar shall be treated as an integer and converted into an octet string of length m such that 28m > r , where r is the order of the group, according to 12.4.7.2.2 (Integer to octet string conversion), and the element shall be converted into (an) octet string(s) according to 12.4.7.2.4 (Element to octet string conversion). When receiving an SAE Commit message the component octet strings in the Scalar field and FFE field(#2530) shall be converted into a scalar and element, respectively, according to 12.4.7.2.3 (Octet string to integer conversion) and 12.4.7.2.5 (Octet string to element conversion), respectively.