IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adding a Password Identifier to SAE | | | | |
| Date: 2018-01-15 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE | 3333 Scott boulevard  Santa Clara, California  United States of America |  |  |
|  |  |  |  |  |

Abstract

This document proposes a way to add a password identifier to SAE allowing for a password to be uniquely identified when an ambiguity exists, for instance when a password is identified by a wildcard peer MAC address.

Discussion:

In practice passwords are shared among multiple STAs. This is effectively having a wildcard MAC address as the peer MAC address in the dot11RSNConfigPasswordValueTable. In some situations, it is necessary to have a multitude of shared passwords where different members of different groups would share a password. In this case it is necessary to provide a password identifier to allow the recipient of an SAE Commit message to further idenfity the particular password to use.

This submission proposes to define a new Element and add it to SAE Commit messages to identify the password to use in the SAE exchange.

***Instruct the editor to modify table 9-40 as indicated:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Authentication algorithm** | **Authentication transaction sequence number** | **Status code** | **Presence of fields 4 onwards** |
| SAE | 1 | Any | Scalar is present if the Status Code field is zero.  Element is present if the Status Code field is zero.  Anti-Clogging Token is present if status is 76 or if frame is in response to a previous rejection with Status 76.  Finite Cyclic Group is present if the Status Code field is zero or 76.  Password Identifier element is optionally present if the Status Code is zero. |

***Instruct the editor to modify table 9-88 as indicated and to add a new sub-section to 9.4.2 to add the following text (currently 9.4.2.a):***

**9.4.2 Elements**

**Table 9-88—Element IDs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Element ID** | **Element ID**  **Extension** | **Extensible** | **Fragmentable** |
| Password identifier (see 9.4.2.a (Password Identifier element)) | 255 | <ANA-1> | No | No |

**9.4.2.a Password identifier element**

The Password identifier element contains a string used to look up a password. See Figure 9-b (Password identifier element format).

|  |  |  |  |
| --- | --- | --- | --- |
| Element ID | Length | Element ID Extension | Identifier |

Octets 1 1 1 variable

**Figure 9-b—Password identifier element format**

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

The Identifier field is a variable-length string which identifies a password as specified in 12.4 (Authentication using a password).

***Instruct the editor to modify section 12.4.3 as indicated:***

**12.4.3 Representation of a password**

Passwords are used in SAE to deterministically compute a secret element in the negotiated group, called a password element. The input to this process needs to be in the form of a binary string. For the protocol to successfully terminate, it is necessary for each side to produce identical binary strings for a given password, even if that password is in character format. There is no canonical binary representation of a character and ambiguity exists when the password is a character string. To eliminate this ambiguity, a STA shall represent a character-based password as an ASCII string. Representation of a character-based password in another character set or use of a password preprocessing technique (to map a character string to a binary string) may be agreed upon, in an out-of-band fashion, prior to beginning SAE. If the password is already in binary form (e.g., it is a binary preshared key) no character set representation is assumed. The binary representation of the password, after being transformed from a character representation or directly if it is already in binary form, is stored in the dot11RSNConfigPasswordValueTable. When a “password” is called for in the description of SAE that follows the credential from the dot11RSNConfigPasswordValueTable is used. When a “password identifier” is called for in the description of SAE that follows, the identifier from the dot11RSNConfigPasswordValueTable is used.

***Instruct the editor to modify section 12.4.4.2.2 as indicated:***

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

The password element of an ECC group (***PWE***) shall be generated in a random hunt-and-peck fashion. The password, optionally a password identifier, and a counter, represented as a single octet and initially set to 1, are used with the peer identities to generate a password seed.

Algorithmically this process is described as follows:

found  = 0;

counter  = 1

Length  = len(p )

base = password [ || identifier ]

  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 )

where

KDF-Hash-Length  is the key derivation function defined in 12.7.1.7.2 (Key derivation

Function (KDF)) using the hash algorithm identified by the AKM suite

selector (see Table 9-145 (AKM suite selectors))

len() returns the length of its argument in bits

[ || identifier ] indicates the optional inclusion of a password identifier, if present

***Instruct the editor to modify section 12.4.4.3.2 as indicated:***

**12.4.4.3.2 Genration of the password element with FFC groups**

The password element of an FFC group (***PWE***) shall be generated in a random hunt-and-peck fashion similar to the technique for an ECC group. The password, optionally a password identifier, 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.

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 || [ identifier || ] counter )

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

where

KDF-Hash-Length  is the key derivation function defined in 12.7.1.7.2 (Key derivation

Function (KDF)) using the hash algorithm identified by the AKM suite

selector (see Table 9-145 (AKM suite selectors))

len() returns the length of its argument in bits

[ || identifier ] indicates the optional inclusion of a password identifier, if present

***Instruct the editor to modify section 12.4.5.4 as indicated:***

**12.4.5.4 Processing of a peer’s SAE Commit message**

If the peer’s SAE Commit message contains a password identifier, the value of that identifier shall be used in construction of the password element (***PWE***) for this exchange. If there is no password with the given identifier a random password shall be used to simulate an SAE exchange which will fail without leaking any information about valid passwords and identifiers.

Upon receipt of a peer’s SAE Commit message both the scalar and element shall be verified.

***Instruct the editor to modify section 12.4.7.4 as indicated:***

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

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 Status codes not equal to SUCCESS indicate a rejection of a peer’s SAE Commit message and are described in 12.4.7.6 (Status codes).

An SAE Commit message shall consist of a Finite Cyclic Group field (9.4.1.44 (Finite Cyclic Group field)) indicating a group, a Scalar field (9.4.1.41 (Scalar field)) containing the scalar, and an FFE field containing the element (9.4.1.42 (Finite field element (FFE) field)). If the SAE Commit message is in response to an Anti-Clogging Token request (see 12.4.7.6 (Status codes)), the Anti-Clogging Token is present (see 9.4.1.40 (Anti-Clogging Token field)). 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 an ASCII string in the Identifier portion of the element (see 9.4.2.a (Password identifier element)).

***Instruct the editor to modify section C.3 as indicated:***

**C.3 MIB detail**

Dot11RSNAConfigPasswordValueEntry ::=

SEQUENCE {

dot11RSNAConfigPasswordValueIndex Unsigned32,

dot11RSNAConfigPasswordCredential OCTET STRING,

dot11RSNAConfigPasswordIdentifier OCTET STRING,

dot11RSNAConfigPasswordPeerMac MacAddress }

dot11RSNAConfigPasswordIdentifier OBJECT-TYPE

SYNTAX Identifier

MAX-ACCESS read-write

STATUS current

DESCRIPTION

“This is a control variable.

It is written by an external management entity.

Changes take effet as soon as practical in the implementation.

This variable is an ASCII string that an implementation uses to uniquely

identify a password when there are multiple passwords for a single

PeerMac configured.”

::= { dot11RSNAConfigPasswordIdentifierEntry 3 }

dot11RSNAConfigPasswordPeerMac OBJECT-TYPE

SYNTAX MacAddress

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This is a control variable.

It is written by an external management entity.

Changes take effect as soon as practical in the implementation.

This variable represents the MAC address of the peer

that is to be authenticated. A wildcard BSSID is

permitted when passwords are shared among peers."

::= { dot11RSNAConfigPasswordValueEntry 4 }

**References:**