### IEEE P802.11Wireless LANs

|  |
| --- |
| Protected Password Identifiers for Privacy |
| Date: 2021-03-30 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Abstract

This submission proposes an efficient way to provide privacy protections to SAE password identifiers.

Password identifiers can be used to track users. Therefore, it is proposed to provide a way of providing a STA with a pseudonymous, and stateless identity that can be used for one-time access and a way to obtain a new pseudonym for use with a single subsequent connection.

The idea is that the Authenticator maintains a secret which is used to encrypt the plaintext identifier to produce a pseudonym which the Authenticator is able to decrypt. STAs then add the pseudonym to their network profile and use that, instead of the plaintext identifier, on subsequent connections. Each new connection will result in the Authenticator generating a new pseudonym for the STA which it overwrites in its network profile for the next connection.

Pseudonymous, or encrypted, password identities are useful when randomized MAC addresses are used for privacy—if a fixed MAC address is used then there would be little point in attempting to protect the creation of PII from the password identity when PII from the fixed MAC address is possible. Pseudonyms must obscure the underlying password such that a passive attacker will not know, based on inspection of pseudonyms, whether two STAs are using the same password identifier or different password identifiers. This works in an ESS, where the AP presents a fixed identity to a STA. Different STAs can use different pseudonyms, even if they represent the same underlying identity, and the AP can decrypt them all. A mesh poses a different case though.

If mesh points use random MAC addresses and each pairwise connection between mesh points uses a different password identifier (the degenerate case in an ESS when password identifiers equate to usernames) then there is no way for a mesh point to determine which password identifier to use or which pseudonym to include in the Commit message. To get around this issue, when a mesh uses encrypted password identities all mesh points in the mesh must share the same symmetric secret used to construct pseudonyms (in the same way that APs in an ESS must share the same symmetric secret) and that every mesh point needs to know the password/identifier of every other mesh point. But this obviates the need for per-link passwords—security is reduced to that where a single password is shared for the whole mesh. While permitted, there’s little point in using password identifiers in a mesh.

This scheme has the following properties:

* A passive attacker cannot determine a protected identity;
* Identifiers are protected against active attack insofar as SAE is resistant to active attack;
* A passive attacker cannot connect protected identities across SAE protocol runs to generate PII;
* Password identifiers can be arbitrarily padded to foil passive traffic analysis;
* Protected identities are secure under a birthday bound of 232 encryptions;
* An attacker cannot tamper with or substitute identifiers to connect distinct runs of SAE;
* An AP/mesh point needs to only manage a single symmetric secret;
* APs in an ESS, and mesh points in a mesh, share a single symmetric secret (in an out of band, out of scope manner);
* AP can use the same symmetric secret to protect all groups in the ESS that use password identifiers;
* In an ESS, identities are protected against members of the same group (not so with a mesh);
* The interface for password identifiers on a STA is unchanged;
* The overhead is minimal—25 octets plus padding;
* Uses symmetric cryptography for speed and DOS resistance;
* Protected password identifiers in a mesh is supported although there is little point in using password identifiers in a mesh.

Proposed change:

*Instruct the editor to modify section 9.3.3.11 as indicated:*

**9.3.3.11 Authentication frame format**

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

|  |  |  |
| --- | --- | --- |
|  **Order** |  **Information** |  **Notes** |
|  22 |  Password Identifier | The Password Identifier element is optionally present in certain Authentication frames as defined in Table 9-43 (Presence of fields and elements in Authentication frames) |
|  23 |  Rejected Groups | 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). |
|  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). |
|  25 | Protected Password Identifier | The Protected Password Identifier element is optionally present 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 | Presence of fields and elements from order 4 onwards |
|  SAE |  1 |  Any | The Scalar field is present if the Status Code field is zero or 126.The FFE field is present if the Status Code field is zero or 126.When the hunting-and-pecking method is used to drive the PWE, the Anti-Clogging Token field is present if the Status Code field is ANTI\_CLOGGING\_TOKEN\_REQUIRED or if the Authentication frame is in response to a previous rejection with the Status Code field equal to ANTI\_CLOGGING\_TOKEN\_REQUIRED.The Finite Cyclic Group field is present if the Status Code field is zero, ANTI\_CLOGGING\_TOKEN\_REQUIRED, 77, or 126.The Password Identifier element is optionally present if the Status Code field is zero, 123, or 126, and the Protected Password Identifier element is not present.The Rejected Groups element is present if the Status Code field is 126.When the hash-to-element method is used to derive the PWE, the Anti-Clogging Token Containerelement is present if the Status Code field is ANTI\_CLOGGING\_TOKEN\_REQUIRED or if the Authentication frame is in response to a previous rejection with the Status Code field equal to ANTI\_CLOGGING\_TOKEN\_REQUIRED.The Protected Password Identifier element is optionally present if the Status Code field is zero, 123, or 126, and the Password Identifier field is not present. |

*Instruct the editor to modify table 9-94 as indicated, obtain a new identifier for the new element, and replace <ANA-1> with that number.*

**9.4.2 Elements**

**9.4.2.1 General**

 **Table 9-94—Element IDs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  **Element** | **Element ID** | **Element ID Extension** | **Extensible** | **Fragmentable** |
| Anti-Clogging Token Container (see 9.4.2.247 (Anti-Clogging Token Container element)) | 255 |  93 |  No | No |
| Protected Password Identifier element (see 9.4.2.X (Protected Password Identifier element)) |  255 |  <ANA-1> |  No |  No |
| Reserved |  255 |  <ANA-1> + 1-255 |  |  |

*Instruct the editor to create a new section as below, replacing X with the appropriate number and assigning the figure number appropriately:*

**9.4.2.X Protected Password Identifier element**

The Protected Password Identifier element is used to convey a password identifier duing an authentication exchange in a manner that will hide the actual value from attackers. The format of the Protected Password Identifier element is shown in Figure 9-XYZ (Protected Identifier element format).

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

 Octets: 1 1 1 variable

 **Figure 9-XYZ—Protected Identifier element format**

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

The Protected Identifier field contains an opaque variable-length string.

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

**12.4.3 Representation of passwords and password identifiers**

In an infrastructure BSS for which an SAE AKM is indicated, the AP shall set the SAE Password Identifiers In Use subfield of the Extended Capabilities field of the Extended Capabilities element to 1 if any entry in the dot11RSNAConfigPasswordValueTable has a non-NULL dot11RSNAConfigPasswordIdentifier, and shall set it to 0 otherwise. Similarly, an AP shall set the SAE Password Identifiers Used Exclusively subfield of the Extended Capabilities field of the Extended Capabilities element to 1 if every entry in the dot11RSNAConfigPasswordValueTable has a non- NULL dot11RSNAConfigPasswordIdentifier and shall set it to 0 otherwise.

Password identifiers, and encrypted password identifiers, are supported in a mesh but due to the unique nature of mesh they do not provide any added security benefit. If mesh points do not support MAC privacy (see 12.2.10 (Requirements for support of MAC privacy enhancements)) then the password is associated with their fixed MAC address and there is no need for password identifiers. If mesh points do support MAC privacy then every mesh point will need to have the password, and password identifier, of every other mesh point—since there is no way for a mesh point to know which password identifier to use when constructing an SAE Commit message. This obviates unique the need for per-link passwords; if there is a single password for the entire mesh then there is no need to have password identifiers.

When encrypted password identifiers are supported, all APs in an ESS or all mesh STAs in a mesh shall share a symmetric key to use with AES-SIV (either a 256-bit or 512-bit key). This is referred to as *pk* below.

When encrypted password identifiers are supported in an ESS, a STA’s initial connection to an AP shall use a plaintext password identifier. After that initial connection, an AP provides an encrypted identifier to the STA to use in a subsequent connection. The password identifier from the dot11RSNAConfigPasswordValueTable remains unchanged but the AP encrypts the identifier and sends the encrypted identifier to the STA during the 4-way Handshake. Each subsequent SAE authentication will result in a new encrypted identifier for the next SAE authentication. In this way, each encrypted identifier is used with only one run of the SAE protocol.

When encrypted password identifiers are supported in a mesh, each mesh point generates its own encrypted identifier prior to initiating the SAE exchange. There is no need to assign encrypted identifers to peer mesh points since all mesh points share *pk* and are able to process each other’s encrypted identifiers.

1. The encrypted identifier shall be generated by an AP or mesh STA as follows:The plaintext identifier shall be pre-pended with 1 or more octets of padding, the first of which indicates the length of the pad—e.g. if there are 4 octets of padding then the sequence would be 4-0-0-0, if there is only one octet of padding the sequence would simply be 1—the length of the pad should vary each time an encrypted identifier is generated;
2. The padded, plaintext identifier shall be concatenated onto an 8 octet random string to produce a string *p*;
3. A ciphertext, *c*, is generated using AES-SIV *pk* as the key, and *p* as the plaintext;
4. The encrypted identifier is *c*.

The encrypted identifier is provided to a STA in an infrastructure BSS in message 3 of the 4 Way Handshake in the PPI KDE (see 12.7.3) in the Protected Password Identifier element (see 9.4.2.X). A STA that receives an encrypted identifier in the 4 Way Handshake shall retain it and shall use it in a subsequent SAE authentication to another AP in the ESS (infrastructure).

When a STA or mesh STA uses an encrypted identifier in SAE it shall pass it in the Protected Password Identity element in an SAE Commit message. When the Protected Password Identifier element is present in an SAE Commit message, the Password Identifier element shall not be present.

When an AP or mesh STA receives a Protected Password Identifier element in an SAE Commit message it shall decrypt the identity as follows:

1. A string *c* is extracted from the Protected Password Identifier element;
2. A plaintext, *p*, is generated by decrypting using AES-SIV with *pk* as the key and *c* as the ciphertext;
3. If AES-SIV decryption fails, SAE authentication fails. Otherwise, the first 8 octets of *p* are discarded to produce a string *p’*;
4. The length of the pad, *t*, is determined from first octet of the string *p’*;
5. The first *t* octets of the remaining string *p’* are removed and the remainder becomes the decrypted Password Identifier.

The AP that receives an SAE Commit message with a Protected Password Identifier shall echo back the same Protected Password Identifier in its SAE Commit message back to the non-AP STA. A non-AP STA that receives a Protected Password Identifier element in an SAE Commit message shall verify that it matches what it sent to the AP in its SAE Commit message. A mesh STA shall generate its own, unique, encrypted identifier to respond back to the peer mesh point.

*Instruct the editor to modify section 12.4.4.2.3 as indicated:*

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

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 to 1 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. The password identifier, when used, shall be the plaintext identifier associated with the password from the dot11RSNAConfigPasswordValueTable. 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 signaled 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 set to SAE\_HASH\_TO\_ELEMENT.

*Instruct the editor to modify section 12.4.4.3.3 as indicated:*

**12.4.4.3.3 Direct Generation of the password element with FFC groups**

An SAE peer indicates support for direct hashing to obtain the FFC password element by setting the SAE hash-to-element bit to 1 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. The password identifier, when used, shall be the plaintext identifier associated with the password from the dot11RSNAConfigPasswordValueTable. 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.

*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 the peer’s SAE Commit message contains an encrypted identifier, the plaintext identifier shall be used in construction of the secret element PT for this exchange (see 12.4.4.2.3 (Hash-to-curve generation of the password element with ECC groups) and 12.4.4.3.3 (Direct Generation of the password element with FFC groups). If a password identifier, or protected password identifier, is present in the peer’s SAE Commit message and there is no password with the given (decrypted) identifier a STA shall fail authentication.

*Instruct the editor to modify sections 12.4.5.5 and 12.4.5.6 as indicated:*

**12.4.5.5 Construction of an SAE Confirm message**

A peer generates a confirmation, confirm, and inserts it into an SAE Confirm message by passing the KCK, the current value of the send-confirm counter (see 9.4.1.37 (Send-Confirm field)), the scalar and element from the sent SAE Commit message, and the scalar and element from the received SAE Commit message to the confirmation function CN.

 *confirm* = CN(KCK, *send-confirm, commit-scalar*, **COMMIT-ELEMENT**, *peer-commit-scalar*,

 **PEER-COMMIT-ELEMENT | [, *send-encrypted-id, peer-encrypted-id]*** )

The send-confirm counter shall be encoded according to 9.2.2 (Conventions). The elements and scalars shall be in the format they were encoded in when transmitted in an SAE Commit message as described in 12.4.7.4 (Encoding and decoding of SAE Commit messages). send-encrypted-id and peer-encrypted-id are protected password identifiers, when optionally used, of the sender and peer, respectively. The message shall be transmitted to the peer as described in 12.4.7 (Framing of SAE).

Note: in an infrastructure BSS, both encrypted identities will be identical.

**12.4.5.6 Processing of a peer’s SAE Confirm message**

Upon receipt of a peer’s SAE Confirm message a verifier is computed, which is the expected value of the peer’s confirmation, peer-confirm, extracted from the received an SAE Confirm message. The verifier is computed by passing the KCK, the peer’s send-confirm counter from the received an SAE Confirm message (see 9.4.1.37 (Send-Confirm field)), the scalar and element from the received SAE Commit message, and scalar and element from the sent SAE Commit message to the confirmation function CN.

 *verifier* = CN(KCK, *peer-send-confirm, peer-commit-scalar*, **PEER-COMMIT-ELEMENT**,

 *commit-scalar*, **COMMIT-ELEMENT [, *peer-encrypted-id, send-encrypted-id*]** )

The peer-send-confirm shall be encoded according to 9.2.2 (Conventions). The elements and scalars shall be in the format they were encoded in when transmitted in an SAE Commit message as described in 12.4.7.4 (Encoding and decoding of SAE Commit messages). peer-encrypted-id and sender-encrypted-id are the protected password identifiers, when optionally used, of the peer and sender, respectively. If the verifier differs from the peer confirm, verification of the peer’s SAE Confirm message shall fail.

Note: in an infrastructure BSS, both encrypted identities will be identical.

*Instruct the editor to obtain a new data type from ANA and modify table 12-9 in section 12.7.3 as indicated, replacing <ANA-2> below with the new data type:*

 **Table 12-9—KDE selectors**

|  |  |  |
| --- | --- | --- |
|  **OUI** |  **Data type** |  **Meaning** |
|  00-0F-AC |  13 |  OCI KDE |
|  00-0F-AC |  14 |  BIGTK KDE |
|  00-0F-AC |  <ANA-2> |  PPI KDE |
|  00-0F-AC | <ANA-2>+1 -255 |  Reserved |
| Other OUI or CID |  Any |  Vendor specific |

*Instruct the editor to append the following to section 12.7.3*

The format of the PPI KDE is shown in Figure 12-AB (PPI KDE).

|  |
| --- |
|  PPI  |

 Octets: (Length – 4)

 **Figure 12-AB—PPI KDE format**

The PPI is an opaque string that shall be retained by a STA and used as a Protected Password Identifier with a subsequent SAE authentication to the same ESS with which it is performing the 4-way Handshake.

**References:**