IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Finding PWE in Constant Time | | | | |
| Date: 2019-07-08 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE | 3333 Scott boulevard  Santa Clara, California,  United States of America | +1 408 555 1212 |  |
|  |  |  |  |  |

Abstract

The countermeasures against side channel attack that SAE puts into the hunting and pecking loop is inefficient and fragile. Recent attacks—e.g. Dragonblood—underscore this problem. PWE should be discovered in a deterministic method which is computable in constant time without repetitive looping. Do to the minimal number of restrictions on curve type, an appealing way of hashing to an elliptic curve is the Shallue-Woestijne-Ulas (SWU) method. For FFC groups, the branching and looping has been removed in order to generate an FFC PWE directly.

**Discussion**: SAE requires use of a secret element, PWE, discovered in an agreed upon finite cyclic group. This element is deterministically discovered by repeatedly hashing the password with some additional information until the resulting hash is the abscissa of a point on the elliptic curve (for ECC) or by exponentiating the hash digest to a constant to produce an element (for FFC). Both of these techniques are prone to side channel attack. While much work has gone into countermeasures to mitigate these attacks, for instance looping 40 times regardless of how soon an abscissa is found, the countermeasures render the whole method of PWE discovery inefficient and very fragile.

**Proposal**: Use the Shallue-Woestijne-Ulas (SWU) to directly hash-to-curve for ECC. This method will work for any Weierstrass curve which makes it ideal for use with SAE.

Since SWU does not generate all points on the elliptic curve, the SWU method by itself could not be used with the current SAE security proof in the random oracle model. Therefore, the SWU method is enhanced by the following technique from Brier et al:

PWE(m) := SWU(h1(m)) + h2(m) \* G

Where *m* is the information being hashed, *h1*() and *h2*() are random oracles based on a hash function, G is the generator of the curve being hashed into, ‘+’ is the element operator (i.e. point arithmetic) for ECC, and ‘\*’ is the scalar operator for ECC.

For FFC groups the results of the hash will be reduced module the prime instead of skipping values which would be larger than the prime when interpreted as an integer. No looping is needed.

These new techniques are not backwards compatible with the “hunting-and-pecking” loop in the standard and therefore must be signalled as new capabilities. This signalling is compounded by two things: SAE happens before association, and selection of PWE happens before any transactional frames have been sent. Since SAE happens before association, it’s not possible to use an AKM so support of this is signalled by an AP using a bit in the Extended Capabilities field. Since the SAE protocol itself is not being changed the signalling to use this new PWE discovery method has to be as unintrusive as possible. Therefore, it is signalled with a new status code in the SAE Commit message—“I’m hashing directly to PWE!”. Everything else in SAE (state machine, message formats, computation, message construction and processing, etc) remains the same.

*Instruct the editor to add the following line to section 2:*

**2. Normative references**

IETF RFC 5869, HMAC-based Extract and Expand Key Derivation Function, H. Krawczyk, P. Eronen May 2010

*Instruct editor to modify table 9-43 as indicated:*

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

|  |  |  |  |
| --- | --- | --- | --- |
| **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 or 124.  Element is present if the Status Code field is zero or 124.  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,76, or 124.  Password Identifier element is optionally present  if the Status Code is zero, 123, or 124. |

*Instruct the editor to modify table 9-52 as indicated:*

**Table 9-52—Status codes**

|  |  |  |
| --- | --- | --- |
| **Status code** | **Name** | **Meaning** |
| 124 | SAE\_HASH\_TO\_PWE | SAE authentication uses an alternate form of direct hashing, as opposed to looping, to obtain the PWE |
| 125-65535 | Reserved |  |

*Instruct the editor to modify table 9-95 as indicated:*

**Table 9-95—BSS membership selector value encoding**

|  |  |  |
| --- | --- | --- |
| **Value** | **Feature** | **Interpretation** |
| 123 | SAE Hash Only | Indicates that support for the direct hashing to element technique in SAE is required in order to join the BSS. |

*Instruct the editor to modify table 9-322 as indicated:*

**Table 9-322—Extended RSN Capabilities field**

|  |  |  |
| --- | --- | --- |
| **Bit** | **Information** | **Notes** |
| 5 | SAE hash-to-PWE | The AP supports directly hashing to obtain PWE instead of looping. See 12.4.4.2.3 and 12.4.4.3.3 |
| 6 – (8xn – 1) | Reserved |  |

*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 by looping**

When a direct form of hashing to discover PWE is not signaled by the AP, or if the SAE initiator does not signal its use in its SAE Commit message, the password element of an ECC group (PWE ) shall be generated in the following random hunt-and-peck fashion.

*Instruct the editor to add the following new section:*

**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 the ECC PWE by setting the SAE hash-to-PWE bit in the Extended RSN Capabilities field in all Beacons and Probe Response frames. An SAE initiator that has identified a peer that supports this technique (through receipt of Beacons or Probe Response frames) shall derive PWE according to the following technique and indicate this by setting the Status in the SAE Commit message to “SAE\_HASH\_TO\_PWE”. An SAE initiator shall not indicate support for this form of PWE derivation unless its peer has already signalled support for this method. If an SAE Commit message is received with status equal to “SAE\_HASH\_TO\_PWE” the peer shall generate PWE using the following technique and reply with its own SAE Commit message with Status equal to “SAE\_HASH\_TO\_PWE.”

The direct hashing technique to derive the password element of an ECC group (PWE) is an enhancement of the deterministic hash-to-curve method Shallue-Woestijne-Ulas (SWU). It uses a construct of two distinct functions to produce two points on the elliptic curve. The two points are summed to create PWE.

This hash-to-curve method uses HKDF (RFC 5869) with the hash algorithm identified by the AKM suite selector (see Table 9-151 (AKM suite selectors)) to perform both functions. First HKDF-Extract is passed the MAC addresses, minimum first, the password and optionally a password identifier to produce and intermediary password seed. The resulting seed is passed to HKDF-Expand to produce three distinct strings using different labels. The first value is reduced such that 0 < pwd-value < q, exclusive. That value is then uses as the scalar with the generator of the group in scalar-op() to produce a point on the curve, P1. The final two values are reduced such that 1 < pwd-value < p, exclusive and they are used, as *u* and *v*, with the SWU method to produce three values, x1, x2, and x3, at least one of them will represent an abscissa of a point on the curve. If x1 is the abscissa then x1 becomes the x-coordinate otherwise if is the abscissa then x2 becomes the x-coordinate, otherwise x3 becomes the x-coordinate. The equation of the curve with the x-coordinate produces the square of the y-coordinate which is recovered by taking the square root. The two possible results of the square root are discriminated by checking its least significant bit with the least significant bit of *u*. The result is a point on the curve P2. PWE is then the sum of the two points, PWE = elem-op(P1, P2).

Algorithmically, this process is as follows:

*pwd-seed = HKDF-Extract(MIN(STA-A-MAC, STA-B-MAC) || MAX(STA-A-MAC, STA-B-MAC),*

*password [|| identifier])*

*pwd-value = HKDF-Expand(pwd-seed, “SAE Hash to Element P1”, olen(p))*

*w = (pwd-value modulo (q-1)) + 1*

*P1 = scalar-op(w, G)*

*pwd-value = HKDF-Expand(pwd-seed, “SAE Hash to Element P2 u”, olen(p))*

*u = (pwd-value modulo (p-2)) + 2*

*pwd-value = HKDF-Expand(pwd-seed, “SAE Hash to Element P2 v”, olen(p))*

*v = (pwd-value modulo (p-2)) + 2*

*x1 = v*

*gx1 = x13 + a \* x1 + b modulo p*

*x2 = (-b/a) \* (1 + 1/(u4 \* gx12 + u2 \* gx1 ) modulo p*

*gx2 = x23 + a \* x2 + b modulo p*

*x3 = u2 \* gx1 \* gx2*

*gx3 = x33 + a \* x3 + b modulo p*

*l = gx2 is a quadratic residue modulo p*

*x = CSEL(l, x2, x3)*

*z = CSEL(l, gx2, gx3)*

*l = gx1 is a quadratic residue modulo p*

*x = CSEL(l, x1, x)*

*z = CSEL(l, gx1, z)*

*y = sqrt(z)*

*l = CEQ(LSB(u), LSB(y))*

*P2 = CSEL(l, (x,y), (x,p-y))*

*PWE = elem-op(P1, P2)*

where

* HKDF-Extract() and HKDF-Expand are the functions defined in RFC 5869 instantiated with the hash algorithm identified by the AKM suite selector (see Table 9-151 (AKM suite selectors))
* olen() returns the length of its argument in octets
* [|| identifier] indicates the optional inclusion of a password identifier, if present.
* G, p, q, a, and b are all defined in the domain parameter set for the curve.
* LSB(x) returns the least-significant-bit of x
* CSEL(x,y,z) operates in constant time and returns y if x is true and z otherwise.
* CEQ(x,y) operates in constant time and returns true if x equals y and false otherwise.

All operatioins shall be done in constant time. Implementations of this modified SWU method shall use the blinding technique from 12.4.4.2.2 to determine a quadratic residue.

Note—For curves based on a prime, p, such that p = 3 mod 4 the square root can be implemented with a single modular exponentiation of (p+1)/4, that is sqrt(w) = w(p+1)/4 modulo p.

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

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

When a direct form of hashing to discover PWE is not signaled by the AP, or if the SAE initiator does not signal its use in the SAE Commit message the password element of an FFC group (**PWE**) shall be generated in the following random hunt-and-peck fashion.

*Instruct the editor to add the following new section:*

**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 PWE by setting the SAE hash-to-PWE bit in the Extended RSN Capabilities field in all Beacons and Probe Response frames. An SAE initiator that has identified a peer that supports the following technique (through receipt of Beacons or Probe Response frames) shall derive PWE according to the following technique and indicate this by setting the Status in the SAE Commit message to “SAE\_HASH\_TO\_PWE”. 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 equal to “SAE\_HASH\_TO\_PWE” the peer shall generate PWE using the following technique and reply with its own SAE Commit message with Status equal to “SAE\_HASH\_TO\_PWE.”

To perform this direct hashing technique, HKDF (RFC 5869) is passed the two peer MAC addresses as a salt, 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 PWE is a generator of order either 1 (if PWE=1) or q (for all other values). The probability of PWE taking the value 1 is 1/q and due to the size of q of the FFC groups used by SAE this probability is negligible.

Algorithmically, this process is as follows:

*pwd-value = HKDF(MAX(STA-A-MAC, STA-B-MAC) || MIN(STA-A-MAC, STA-B-MAC),*

*password [|| identifier ], “SAE Hash to Element”, olen(p) )*

*pwd-value = (pwd-value modulo (p-2)) + 2*

*PWE = pwd-value(p-1)/q modulo p*

where

* HKDF() is the function defined in RFC 5869 instantiated with the hash algorithm identified by the AKM suite selector (see Table 9-151 (AKM suite selectors))
* olen() returns the length of its argument in octets
* [|| identifier] indicates the optional inclusion of a password identifier, if present.
* p and q are defined in the domain parameter set for the group.

**References:**

Brier E., Coron JS., Icart T., Madore D., Randriam H., Tibouchi M. (2010) “Efficient Indifferentiable Hashing into Ordinary Elliptic Curves.” In: Rabin T. (eds) Advances in Cryptology – CRYPTO 2010. Lecture Notes in Computer Science, vol 6223. Springer, 2010

Shallue, A. and van de Woestijne, C, “Construction of rational points on elliptic curves over finite fields”, Lecture Notes in Computer Science, vol 4076, pages 510-524, Springer, 2006.

Ulas, M., “Rational points on certain hyperelliptic curves over finite fields.”, Polish Academy of Sciences, 55(2): 97-104, 2007