IEEE P802.11
Wireless LANs

|  |
| --- |
| Resolving Some Random SAE Comments |
| Date: 2020-05-08 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE |  |  |  |
|  |  |  |  |  |

Abstract

This submission resolves a set of comments received from a reviewer.

Comment #1: "The polynomial x3 + a × x + b – n is irreducible" is different from Criterion #3 in Section 6.6.2 of the Internet Draft (https://www.ietf.org/id/draft-irtf-cfrg-hash-to-curve-07.txt). If the irreducibility of (x3 + a × x + b – n) is tested instead of (x3 + a × x + b – (-n)), then some z values in Table 12-2 would be different. I suppose the algorithm in the Internet Draft (and therefore Table 12-2) is correct, and it's just a typo here.

Suggested Change: Change to "x3 + a × x + b – (– n)".

Comment #2: "n is not -1" is a redundant criterion, as this algorithm assumes n > 0, and the search for z in the > 0 direction will never reach (-1) mod p, or (p - 1). It will have exhausted all possible candidates at (p - 1) / 2.

Suggested Change: Remove "n is not -1".

Comment #3: The bullet numbering of the algorithm is messed up.

Suggested Change: Correct the bullet numbering.

Discussion: The text was confusing. The criteria is identical to the Internet-Draft it’s just that the same tests have to be made for both +n and -n and the description was odd. It’s probably better to describe in it a way that more closely matches the Internet-Draft, so the technique to derive z should be rewritten to have a single formula called twice one with a positive value and once with a negative value. If the positive one returns success then it’s the end. Serendipitously the bullet numbering issue goes away!

Proposed Change: *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**

The SSWU method takes a curve-specific parameter, z, which is determined from the following formula, given p, a, and b, from the curve's domain parameter set:

Assign a counter, ctr, the value 1. If the following conditions are true for n = ctr then z = ctr. Otherwise, if they are true for n = -ctr then z = -ctr. Otherwise increment ctr and repeat until a value for z is found.

i) n is not a quadratic residue modulo p

ii) n is not –1

iii) the polynomial x3 + a × x + b – n is irreducible

iv) (b/(n × a))3 + a × (b/(n × a)) + b is a quadratic residue modulo p

Values for some defined groups based on their IANA-assigned values are listed in Table 12-2 (Unique curve parameter).

Comment #4: There is a typo in x((p-1)-2)

Proposed Change: It shoud be x(p-1)/2

Comment #5: Strictly speaking, "inverse" is not the best term, as m can be 0 here, and 0 does not have an inverse. The way to calculate the value of t on Line 31 of next page is correct and does cover the case where m = 0.

Suggested Change: Use a different term such as "extended\_inverse" and make a note for m = 0 in Line 31 of next page.

Discussion: Yes, good catch on the typo. Agree that “inverse” is not the best term so looking to align more with the Internet-Draft let’s call it inv0 since that’s what it’s called there. We define the function below anyway so there’s no issue with having a new name. No need to make a note for m=0 because if m=0 then t is not used (per the definition of CSEL() and CEQ())

Proposed Change: *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**

Algorithmically, the Simplified SWU method is:

SSWU(u) {

m = (z2 × u4 + z × u2) modulo p

l = CEQ(m, 0)

t = inv0(m)

x1 = CSEL(l, (b / (z × a) modulo p), ((– b/a) × (1 + t)) modulo p)

gx1 = (x13 + a × x1 + b) modulo p

x2 = (z × u2 × x1) modulo p

gx2 = (x23 + a × x2 + b) modulo p

l = gx1 is a quadratic residue modulo p

v = CSEL(l, gx1, gx2)

x = CSEL(l, x1, x2 )

y = sqrt(v)

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

P = CSEL(l, (x,y), (x, p – y))

output P

}

where:

p, a, and b are all defined in the domain parameter set for the curve.

z is a curve-specific parameter from Table 12-2 (Unique curve parameter(M137)).

inv0(x) is calculated as x(p-2) modulo p.

x is a quadratic residue if x((p-1)/2) modulo p is zero or one.

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 operations in the SSWU algorithm shall be done in constant time.

Comment #6: "The Rejected Groups field contains a concatenated list of unsigned 16-bit integers representing finite cyclic groups that have been rejected…": Since each FC group is represented by 2 octets, endianness matters. Does this list follow the conventions in 9.2.2 (as does the Finite Cyclic Group field defined in 9.4.1.42), or does it follow the data type conversion rules in 12.4.7.2? For example, if Group 20 was rejected, would it show up in the Rejected Groups field as "\x14\x00" or as "\x00\x14"?

Proposed Change: Clarify the byte order.

Discussion: 12.4.5.4 where the KDF is used it says to look at 12.4.7.4 which notes that the bit ordering of the Rejected Groups element is according to the conventions of 9.2.2 but you have to jump around a bit to realize that. Can’t hurt to be repetitive!

Proposed Change: *Instruct the editor to modify section 9.4.2.246 as indicated:*

**9.4.2.246 Rejected Groups element**

The Rejected Groups field contains a concatenated list of unsigned 16-bit integers representing finite cyclic groups, using the octet ordering conventions of 9.2.2 (Conventions), that have been rejected by a peer in a previous authentication attempt.

Comment #7: "… such that 28m > r, where r is the order of the group, …": Section 12.4.7.2.4 uses the domain parameter p for the same purpose as r. Practically, both will yield the same value for m, but it's good to be consistent.

Suggested Change: Change to "… such that 28m > p, where p is the prime number specified by the domain parameters …". Do the same thing for Page 2575, Line 52.

Discussion: The cited text describes the encoding of the scalar. The scalar is always a number less than the order of the group so encoding it according to the length of the order is the right thing to do (even if the length of the order is the same as the length of the prime). The text in section 12.4.7.2.4 uses p because it’s describing how to convert an element into an octet string and back and the coordinates of an elliptic curve are all less than the prime so describing it with p is the right thing to do.

Proposed Change: Reject

Comment 8: "… the order of the particular group, q, …": The test vector in Appendix J.10 implies that the reduced digest "val" was obtained by "modulo (p - 1) + 1", where p is the group prime instead of the group order. Either the spec text is wrong, or the test vector is. Also, if it's indeed the intention to use the group order here, use the letter "r" instead of "q", just to be consistent with later sections, where "r" is used in multiple places to represent the group order.

Suggested Change: Using the curve prime "p" here makes more sense as that would generate a reduced digest "val" within the range (1, p - 1) instead of (1, r - 1).

Discussion: The digest “val” will be the same whether you reduce it modulo (p-1) or (r-1) because the hash that generates the initial value of val is less than r-1. So the test vector is correct. Also, the order should be used here as val is going to be used to generate an element in the group so it should be less than the order. So the text is correct and so is the test vector. Changing the notation is wrong though so there should be a change:

Proposed Change: *Instruct the editor to modify section 12.4.5.2 as indicated:*

**12.4.5.2 PWE and secret generation**

When a STA supports directly hashing to a group element (according to 12.4.4.2.3 (Hash-to-curve generation of the password element with ECC groups) or 12.4.4.3.3 (Direct Generation of the password element with FFC groups)) it computes a secret element, PT, offline at provisioning time for all groups it wishes to support with that password. Prior to initiating SAE to a STA which also supports the direct form of hashing to a group element, or upon receipt of an SAE Commit message indicating it was generated using a direct form of hashing to a group element, it shall generate the PWE by hashing the two peer MAC addresses to produce a digest, reducing the digest modulo the order of the particular group, r, interpreting the reduced digest as an integer and using it with the secret element to generate the PWE:

val = H(0n, MAX(STA-A-MAC, STA-B-MAC) || MIN(STA-A-MAC, STA-B-MAC))

val = val modulo (r – 1) + 1

PWE = scalar-op(val, PT)

Comment #9: It seems that the password identifier has been intentionally removed from the previous version(s). However, "password identifier" still appears in other places throughout the text, including the ensuing H2E sections, Section 12.4.3 (Representation of a password), and the test vectors in Appendix J.10.

Suggested Change: Replace the test vector to be consistent with the spec text. Not sure if H2E will keep the password identifier or not - make changes if appropriate. And the entire section of 12.4.3 seems incoherent - a lot of information regarding the "password" itself is missing from the 2016 version.

Discussion: Password identifiers have not been removed but have been made to be H2E only. The test vector was not updated when that change was made. Also, the incoherence has been addressed already with CID <TBD>.

Proposed Change: *Instruct the editor to replace the hunting-and-pecking test vector in section J.10 with the following:*

group: 19

password: mekmitasdigoat

local MAC address: 4d:3f:2f:ff:e3:87

peer's MAC address: a5:d8:aa:95:8e:3c

H(addrs, mekmitasdigoat | 1):

a9025368 ef78f7d6 5e8d4d55 6f0d1d0d 758f2f7f 1e116eb1 d11307a7 e8a9621a

candidate x value:

b8e89a72 5c57f18e 8f68a7f7 2613e15f 1c904938 c38800ef a01f1306 f5e454b5

H(addrs, mekmitasdigoat | 2):

954bbbf8 923284e4 ca164e3a f0b9520c e53aa35b e39020e9 ccb23aff 86df2226

candidate x value:

da6eb7b0 6a1ac562 4974f90a fdd6a8e9 d5722634 cf987c34 defc91a9 874e5658

.

.

.

H(addrs, mekmitasdigoat | 40):

cde7b81e b539c87a f5bf1be2 402d315c 45ad4c3d b06c9c56 b7f8b7da ae5e5842

candidate x value:

2e12a1d6 15647963 fd7aa4a9 05fd51b6 f49a902f d917ef8f 0ff20010 2699ecdb

PWE (x,y):

da6eb7b0 6a1ac562 4974f90a fdd6a8e9 d5722634 cf987c34 defc91a9 874e5658

f4fefd13 0bd5be08 fe68af3e 4a290272 ec065fd3 671f3c25 bf8ec419 ddc9b822

local private value:

992465fd 3daa3c60 aa6565b7 f62a2a7f 2e12dd12 f198faf4 fbed89d7 ff1ace94

local mask value:

9507a90f 777a044d 6a0830b9 1ea3d5dd 70bece44 e1acffb8 6983b5e1 bf9fb322

local commit:

13002e2c 0f0db524 40ad146d 967114ce 005ce1ea b0aa2c2e 5c2871b7 74f6c257

5c65d5ad 9e008297 07aa36ba 8b859738 fc961d08 243505f4 7c035376 d7ac4bc8

d7b95083 bf43827d 0fc31ed7 78dd3671 fd21a46d 1091d64b 6f9a1e12 72621325

dbe1

peer's private value:

bedfd9e4 da33a468 7778c0e2 b9b878c7 bae6d595 79e6b6e6 e0a392e8 547259f5

peer's mask value:

9a3bbd0d 5f4c14dd 988f8804 fb97db73 6920fd9b 651f7b9a 5d13a5d2 887c1d7f

peer's commit:

1300591b 96f3397f b9451008 48e7b550 543b6720 d88337ee 93fc49fd 6df7e08b

5223e71b 9bb048d3 873f2055 6953a96c 91536fd8 ee6ca9b4 a68a148b 056a909b

e03e83ae 208f60f8 ef553785 8074db06 68703239 9862999b 511e0a15 52a5fea3

17c2

k:

189f666f 3181b4a1 701c2bf7 d34d644e 66b5a104 acb99adf 25f3d48b 878c081f

keyseed:

06900d37 677ed6c1 03ea1386 d753b56b e74dc3a7 e5fe9652 8e580521 daad121a

KCK:

1e733f6d 9bd53256 28730433 8831b09a 39406d12 1017073a 5c30db36 f36cb81a

PMK:

4e4dfab1 a2dd8ac1 a91790f9 53faaa45 2ae5c687 3ab75b63 605ba663 f8a7fe59

PMKID:

8747a600 eea3f9f2 2475df58 ca1e5498

Comment #10: The KDF defined in 12.7.1.6.2 is used throughout the document (with a long history, I acknowledge), but what is of special interest is the way it is invoked in SAE.

Line 42: I suppose this means "i" and "Length" are to be encoded with "LSB 0" bit order within the same octet, then "little endian" byte order across multiple octets (although it has to be said that the clauses in Section 9.2.2 Conventions really do not address bit or byte ordering for values not transmitted as part of a MAC frame).

Line 44: I suppose this means "K", "Label" and "Context" just follow their "natural" byte ordering, for lack of a better word on my part, because "K" is the digest of a hash, and "Label" and "Context" seem to be designed for byte-representation of strings. In most other places in the document, "Label" is an encoded ASCII string, while "Context" is a concatenation of HEX strings such as MAC addresses. (Again it has to be said that 9.2.2 does not really address the issue. But I guess the field just has taken a consensus all these years.)

In SAE, however, an integer takes the place of "Context":

Page 2566, Line 51, and Page 2571, Line 60: "Context" is the group prime p. So the question is, how is the group prime (an integer) represented when it is passed as the "Context" argument to the KDF?

The KDF definition on Page 2655 says "Context" is a bit string. Therefore, to convert p (an integer) to a bit string, the bit- and byte-order matters. Of course, the bit ordering is usually transparent to software. But the byte ordering is not. It is through trial and error that I figured out the test vector in Appendix J.10 used "big endian" to convert "p" to a byte string. But which phrase in 9.2.2 dictated that?

Suggested Change: Explicitly state the byte order of "p" in the KDF invocations on Pages 2566 and 2571.

Or better yet, make the KDF definition in 12.7.1.6.2 clearer. Keep in mind that 9.2.2, a subsection of "MAC frame formats", does not address the byte order of objects that only exist in a STA's internal state machine.

Discussion: It’s an integer but represented as an octet string, per the conversion specified in 12.4.7.2.2

Proposed Change: *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**

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)). The context

passed to KDF-Hash-Length, *p*, is the octet string representation of the prime per 12.4.7.2.2 (Integer to octet string conversion).

len() returns the length of its argument in bits

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

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)). The context

passed to KDF-Hash-Length, *p*, is the octet string representation of the prime per 12.4.7.2.2 (Integer to octet string conversion).

len() returns the length of its argument in bits

Comment #11: “pwd-value”, as the output of the KDF, is a “bytes” object. Before it can be compared with p (in the ensuing “if (pwd-value < p)” statement), it needs to be converted to an “integer” type. However, the conversion is dependent on the byte order (or endianness). The byte order that needs to be used here is not defined by the text. The test vector in Appendix J.10 implies the byte order is "big endian".

Suggested Change: Explicitly define the endianness to “big”.

Comment #12: Earlier in the algorithm, “save = pwd-seed” means “save” is the output of the HMAC extractor function defined in RFC5869. It is a “bytes” object. Before taking its LSB and compare it against the LSB of “y” (which is already an integer, as it’s the output of a sqrt\_modp() function), “save” needs to be converted into an integer. And endianness matters.

Suggested Change: Explicitly define the endianness to “big”.

Comment #13: Similar to Comment #11 above: "pwd-value" needs to be converted to an integer with explicitly specified byte order before taking a modulo operation.

Suggested Change: Explicitly define the endianness to “big”. Similar clarifications are needed for Clauses 12.4.4.3.2 and 12.4.4.3.3 when the finite cyclic group is FFC instead of ECC.

Discussion: yes, the output of the KDF needs to be converted into an integer but the endianness doesn’t matter as long as it’s the consistent. If the big number library represents things in big endian then both pwd-value and p will be in big endian, if it represents things in little endian then both pwd-value and p will be in little endian. The text here is not code, it’s a folksy description of the algorithm. There does not seem to be a problem here. While adding explit mention of endianness of might be possible it would greatly and unnecessarily clutter the description. The input to the KDF matters (see comment #10) but the output doesn’t as long as it’s just used internally.

The only time endianness matters is when information is prepared to be put into a cryptographic function (like H() or KDF()), put on the medium, or taken off the medium and that’s what 12.4.7.2 handles.

While the endianness in the test vector is, indeed, big it’s because the crypto library used to generate it (OpenSSL) represents big numbers in big endian even on a machine whose architecture is little endian.

Proposed Change: Reject all three.

**References:**