IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Some SA2 SAE comments | | | | |
| Date: 2020-09-16 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Jouni Malinen | Qualcomm, Inc. |  |  | jouni@qca.qualcomm.com |
|  |  |  |  |  |

Abstract

This document discusses and proposes solutions to the following REVmd/D4.0 comments from SA2: CID 5071, 5073, 5074, 5075.

**CID 5071**

Clause: 12.4.4.2.3 and 12.4.4.3.3 (REVmd/D4.0 pages 2559 and 2563)

Comment:

Followup to CID 4671. We need to specify what bits are set to

Proposed Change:

Change "by setting the SAE hash-to-element bit" to "by setting to 1 the SAE hash-to-element bit" in 12.4.4.2.3 Hash-to-curve generation of the password element with ECC groups(M137) and 12.4.4.3.3 Direct Generation of the password element with FFC groups(M137)

Discussion:

The comment is proposing following changes:

REVmd/D4.0 page 2559 line 48

**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 to 1 the SAE hash-to-element bit in the Extended RSN Capabilities field in all Beacon and Probe Response frames.

REVmd/D4.0 page 2563 line 47

**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 to 1 the SAE hash-to-element bit in the Extended RSN Capabilities field in all Beacon and Probe Response frames.

The current draft seems to include both uses "set <a bit>" and "set <a bit> to one" (or "set <a bit> to 1") for indicating that the specified bit (single bit field) will be set to have the value 1. The simpler "set <a bit>" language assumes the bit has value 0 by default and setting it would change to have a value 1. While P802.11 does not seem to use "clear <a bit>" construction, that is used outside our standard to mean clearing the bit to 0. Since it is possible to set a bit to either a value 0 or 1, it seems to be less ambiguous to avoid the "set <a bit>" construction. Unfortunately, this results in somewhat more inconvenient language, so selecting which variant to use is a compromise between readability and exactness. While the baseline includes examples of both variants, it feels like the explicit "set to 1" variant is more common and it does not result in overly complex sentences in these two cases.

Proposed Resolution:

ACCEPTED.

**CID 5073**

Clause: 12.4.4.2.3 (REVmd/D4.0 page 2561 line 62)

Comment:

CSEL and CEQ are not the only operators that need to be performed in constant time to avoid security vulnerability. See e.g. https://www.bearssl.org/constanttime.html, which indicates multiplication and divison need to operate in constant time for cryptographic purposes

Proposed Change:

At the referenced location add "x × y operates in constant time and returns the product of x and y", and another line "x / y operates in constant time and returns x divided by y", making x and y italic and with tabbing before "operates" as for the existing text

Discussion:

The comment is proposing following changes:

REVmd/D4.0 page 2561 line 62

**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* = (*z*2 × *u*4 + *z* × *u*2) modulo *p*

*l* = CEQ(*m*, 0)

*t* = *inv0*(*m*)

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

*gx1* = (*x1*3 + *a* × *x1* + *b*) modulo *p*

*x2* = (*z* × *u*2 × *x1*) modulo *p*

*gx2* = (*x2*3 + *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).

*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.

*x* × *y* operates in constant time and returns the product of *x* and *y*

*x* / *y* operates in constant time and returns *x* divided by *y*

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

...

The "all operations" statement just couple of lines below this applies to the modular multiplication and division operations. No duplication is needed for this case unlike the CSEL/CEQ cases where the operations have been explicitly designed for the purpose of performing constant time and memory access operation and that specificity is included in the name and description of the individual operations. Furthermore, these modular operations are common in modular arithmetic which the reader of this algorithm is expected to be familiar with.

Proposed Resolution:

REJECTED. Modular multiplication and division are operations in the SSWU algorithm and as such, are already covered by practically identical requirement on line 63.

**CID 5074**

Clause: 12.4.4.2.3 (REVmd/D4.0 page 2561 line 57)

Comment:

Followup to CID 4669 and 4670. The determination of quadratic residueness could result in branching, so its constant timeness needs to be made explicit

Proposed Change:

Change the "where" for x to "is a quadratic residue if x((p-1)/2) modulo p is zero or one; this determination operates in constant time." (preserving the existing formatting)

Discussion:

The comment is proposing following changes:

REVmd/D4.0 page 2561 line 62

**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* = (*z*2 × *u*4 + *z* × *u*2) modulo *p*

*l* = CEQ(*m*, 0)

*t* = *inv0*(*m*)

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

*gx1* = (*x1*3 + *a* × *x1* + *b*) modulo *p*

*x2* = (*z* × *u*2 × *x1*) modulo *p*

*gx2* = (*x2*3 + *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).

*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; this determination operates in constant time.

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.

...

The proposed change is unnecessary since just four lines below it there is a statement noting that all operations in the algorithm are done in constant time which applies for the determination of quadratic residueness. No duplication is needed for this case unlike the CSEL/CEQ cases where the operations have been explicitly designed for the purpose of performing constant time and memory access operation and that specificity is included in the name and description of the individual operations.

Proposed Resolution:

REJECTED. Determination of quadratic residue on page 2561 line 58 is one of the operations in the SSWU algorithm and as such, is already covered by practically identical requirement on line 63.

**CID 5075**

Clause: 12.4.4.2.3 (REVmd/D4.0 page 2561 line 62)

Comment:

Followup to CID 4669 and 4670. It's quite possible that an implementation of sqrt() could result in branching/short-circuiting, so the requirement for constant time needs to be specified. Also sqrt() is not defined

Proposed Change:

At the referenced location add "sqrt(x) operates in constant time and returns the square root of x", making x italic and tabbing before "operates" as for the existing text

Discussion:

The comment is proposing following changes:

REVmd/D4.0 page 2561 line 62

**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* = (*z*2 × *u*4 + *z* × *u*2) modulo *p*

*l* = CEQ(*m*, 0)

*t* = *inv0*(*m*)

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

*gx1* = (*x1*3 + *a* × *x1* + *b*) modulo *p*

*x2* = (*z* × *u*2 × *x1*) modulo *p*

*gx2* = (*x2*3 + *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).

*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.

sqrt(*x*) operates in constant time and returns the square root of *x*.

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

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*.

...

The "all operations" statement just couple of lines below this applies to the sqrt() operation.

No duplication is needed for this case unlike the CSEL/CEQ cases where the operations have been explicitly designed for the purpose of performing constant time and memory access operation and that specificity is included in the name and description of the individual operations.

The NOTE following this algorithm description indicates how sqrt(x) is implemented. The condition in that note applies to all the applicable curves. As such, there is no need to define sqrt(x) in the SSWU algorithm itself as it is a common operation in modular arithmetic which the reader of this algorithm is expected to be familiar with and is implemented in the manner described immediately following this text.

Proposed Resolution:

REJECTED. sqrt(x) is one of the operations in the SSWU algorithm and as such, is already covered by practically identical requirement on line 63. The NOTE following this algorithm description indicates how sqrt(x) is implemented. The condition in that note applies to all the applicable curves. As such, there is no need to define sqrt(x) in the SSWU algorithm itself as it is a common operation in modular arithmetic which the reader of this algorithm is expected to be familiar with and is implemented in the manner described immediately following this text.