IEEE P802.11
Wireless LANs

|  |
| --- |
| Some REVmd D3 security comments |
| Date: 2020-01-13 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Jouni Malinen | Qualcomm, Inc. |  |  | jouni@qca.qualcomm.com |
|  |  |  |  |  |

Abstract

This document discusses some REVmd D3.0 comments and proposes resolutions for them. The following CIDs are covered: 4815, 4132

r1:

* add missed AP PeerKey change for CID 4132

r2:

* remove an addition to a NOTE for CID 4132 since it was causing more confusion than help

r3:

* changes to CID 4815 redline based on TGmd discussion (Jan 13 PM1)

**CID 4815**

Clause: 12.4.5.4 Page: 2574 Line: 25

Comment:

The current definition of PMKID derivation for SAE is a bit ambiguous and strictly speaking invalid:

PMKID = L((commit-scalar + peer-commit-scalar) mod r, 0, 128)

That first argument to L is an integer in range 0..r-1. However, L() is defined to use a bit string as the first argument. This leaves it open what the length of such bit string is if there would be leading zeros in
that (commit-scalar + peer-commit-scalar) mod r when comparing it to the order (or even worse in some cases, the prime). Conversion of that integer to a bit string should be noted here and explicitly the length
of the bit string (how many zero bits to pad \_from left\_, I'd assume) so that it can be used unambiguously with L.

This actually resulted in nonsensical result for groups 22-24 since all scalar to octet conversions used the length of the prime (in octets) as the length of the octet string and that included this case here with octet string considered to define the bit string for L() as well. That results in the PMKID being all zeros for those groups due to the "mod r" part making the result significantly smaller than the prime.

I'd guess this was supposed to be defined in a manner that made the bit string be as many bits (or 8 \* octets? hopefully same for all the groups we use) as there are in the order, not the prime. For example, 160 bits in case of group 22.

kck\_and\_pmk derivation is even more confusing:

kck\_and\_pmk = KDF-Hash-512(keyseed, "SAE KCK and PMK",
 (commit-scalar + peer-commit-scalar) mod r)

That third argument to the KDF function is "Context" in 12.7.1.7.2 and that is defined a bit string. In practice, the HMAC-based definition of KDF cannot really use an arbitrary bit string, i.e., this has to be an octet string in practice. However, the length of that bit or octet string is not clear: it could be number of octets/bits in the prime (with left padding using zeros), number of octets/bits in the order (with left padding using zeros), or bit string without zero padding (with difficulties in HMAC implementation..).

Proposed Change:

Replace definitions with:

keyseed = H(salt, k)
context = (commit-scalar + peer-commit-scalar) mod r
kck\_and\_pmk = KDF-Hash-Length(keyseed, "SAE KCK and PMK", context)
KCK = L(kck\_and\_pmk, 0, Q)
PMK = L(kck\_and\_pmk, Q, 256)

where

 salt is either a series of 0 octets or a list of rejected groups (see 12.4.7.4 (Encoding
 and decoding of SAE Commit messages)).

 KDF-Hash-Length is the key derivation function defined in 12.7.1.6.2 (Key derivation function
 (KDF)) using the hash algorithm defined for H().

 Q is the length of the digest of the H(), the hash function used.

 Length is Q plus 256.

 context is encoded as a bit string that is zero padded from left to
 as many full octets as is needed to encode r.

The PMK identifier is defined as follows:

 PMKID = L(context, 0, 128)

Discussion:

This comment was discussed during the December conference call and there was a request to show redline version of the proposed changes. This document provides that redline version. This includes additional changes to clarify language based on the comments during that call.

Proposed Resolution:

REVISED. Incorporate changes from the "Proposed Changes for CID 4815" section of <URL-TO-THIS-DOCUMENT>.

**Proposed Changes for CID 4815**

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

*Change the end of 12.4.5.4 (page 2574 lines 23-44) as shown:*

 *keyseed* = H(*salt*, *k*)

 *context* = (*commit-scalar* + *peer-commit-scalar*) mod *r*

 *Length = Q + 256*

 *kck\_and\_pmk* = KDF-Hash-*Length*(*keyseed*, “SAE KCK and PMK”, *context*)

 *KCK* = L(kck\_and\_pmk, 0, Q)

 *PMK* = L(kck\_and\_pmk, Q, 256)

where

 *salt* is either a series of 0 octets or a list of rejected groups (see 12.4.7.4 (Encoding

and decoding of SAE Commit messages)).

 KDF-Hash-*Length* is the key derivation function defined in 12.7.1.6.2 (Key derivation function

(KDF)) using the hash algorithm defined for H().

 Q is the length of the digest of the H(), the hash function used.

 *context* is treated as an integer and converted into an octet string of length *m* such that 28*m* > r

according to 12.4.7.2.2 (Integer to octet string conversion).

The PMK identifier is defined as follows:

 PMKID = L(*context*, 0, 128)

**CID 4132**

Clause: 12.4.4.1 Page: 2564 Line: 6

Comment:

At 2564.6, it states: "For the purpose of interoperability, a STA shall support group 19, an ECC group defined over a 256-bit prime order field.". If a STA suggests group 19 and only group 19, it means that the STA only support group 19 therefore the AP shall not reject it. Perhaps, we should add a clarification.

Proposed Change:

Add a clarification after the quoted sentence ( 2564.6) or at 2564.11: "If group 19 is the only group that the SAE initiator proposes, a SAE peer shall not reject group 19."

Discussion:

The purpose of this “shall support group 19” statement is to guarantee that at least one shared group is implemented by all STAs. This is not to imply that the particular group 19 would always be enabled for use in SAE authentication and as such, it would be incorrect to add a shall statement to disallow rejection of group 19 in general. Taken into account this sentence can be misunderstood in its current form, it would seem useful to clarify it.

It should also be noted that the SAE initiator does not propose a full list of groups in the Authentication frame; it proposed only a single group at a time. As such, when processing a received Authentication frame, the peer STA cannot know whether group 19 is the only group the SAE initiator is planning on proposing.

There is similar language for AP PeerKey protocol, so that should be covered as well.

Proposed Resolution:

REVISED. Incorporate changes from the "Proposed Changes for CID 4132" section of <URL-TO-THIS-DOCUMENT>.

**Proposed Changes for CID 4132**

**12.4.4.1 General**

*Modify 12.4.4.1 (only the beginning of the subclause included here) as shown:*

SAE uses discrete logarithm cryptography to achieve authentication and key agreement. Each party to the exchange derives ephemeral public and private keys with respect to a particular set of domain parameters that define a finite cyclic group. Groups may be based on either Finite Field Cryptography (FFC) or on Elliptic Curve Cryptography (ECC). Each component of a group is referred to as an element. Groups are negotiated using an identifying number from a repository maintained by IANA as “Group Description” attributes for IETF RFC 2409 (IKE) [B14][B28]. The repository maps an identifying number to a complete set of domain parameters for the particular group. Not all groups defined in this repository are suitable. Only FFC groups whose prime is at least 3072 bits and ECC groups defined over a prime field whose prime is at least 256 bits are suitable for use with SAE. ECC groups defined over a characteristic 2 finite field or ECC groups with a co-factor greater than 1 shall not be used with SAE (see NIST Special Publication 800-57). For the purpose of interoperability, a STA shall implement support for group 19, an ECC group defined over a 256-bit prime order field.

More than one group may be configured on a STA for use with SAE by using the dot11RSNAConfigDLCGroup table. Configured groups are prioritized in ascending order of preference. If only one group is configured, it is, by definition, the most preferred group.

NOTE—The preference of one group over another is a local policy issue.

**12.11.2 AP PeerKey protocol**

*Modify 12.11.2 (only the beginning of the subclause included here) as shown:*

AP PeerKey uses the same discrete logarithm cryptography as SAE (as described in 12.4 (Authentication using a password)) to achieve key agreement. Each party to the exchange has a public and private key with respect to a particular set of domain parameters that define a finite cyclic group. Groups may be based on elliptic curve cryptography (ECC) or finite field cryptography (FFC). Each component of a group is referred to as an element. Groups are negotiated using an identifying number from a repository maintained by IANA as “Group Description” attributes for IETF RFC 2409 (IKE) [B14][B28]. The repository maps an identifying number to a complete set of domain parameters for the particular group. For the purpose of interoperability, APs that have dot11ProtectedHCCATXOPNegotiationImplemented true or dot11ProtectedQLoadReportImplemented true shall implement support for group 19, an ECC group defined over a 256-bit prime order field.