IEEE P802.11
Wireless LANs

|  |
| --- |
| Resolution of Some Security Comments from LB202 |
| Date: 2014-07-15 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | Aruba Networks | 1322 Crossan ave, Sunnyvale, CA | +1 408 227 4500 | dharkins at aruba networks dot com |
|  |  |  |  |  |

Abstract

This submission proposes resolutions to CIDs 3037 and 3040.

***Instruct the editor to modify section 11.6.1.7.3 as indicated:***

**11.6.1.7.3 PMK-R0**

The first-level key in the FT key hierarchy, PMK-R0, is derived using the KDF defined in 11.6.1.7.2 (Key derivation function (KDF)). The PMK-R0 is the first level 256-bit keying material used to derive the next level keys (PMK-R1s):

R0-Key-Data = KDF-Hash-Z(XXKey, "FT-R0", SSIDlength || SSID || MDID || R0KHlength ||

R0KH-ID || S0KH-ID)

PMK-R0 = L(R0-Key-Data, 0,Q)

PMK-R0Name-Salt = L(R0-Key-Data, Q, 128)

Where

* KDF-Hash-Z is the KDF as defined in 11.6.1.7.2 (Key derivation function (KDF)) used to generate a key of length 384 bits.
* L(-) is defined in 11.6.1 (Key hierarchy).
* If the AKM negotiated is 00-0F-AC:3, then Hash shall be SHA256, Z shall be 384, Q shall be 256, and XXKey shall be the second 256 bits of the MSK (which is derived from the IEEE Std 802.1X authentication), i.e., XXKey = L(MSK, 256, 256). If the AKM negotiated is 00-0F-AC:4, then Hash shall be SHA256, Z shall be 384, Q shall be 256, and XXKey shall be the PSK. If the AKM negotiated is 00-0F-AC:9, then Hash shall be SHA256, Z shall be 384, Q shall be 256, and XXKey shall be the MPMK generated as the result of SAE authentication. If the AKM negotiated is 00-0FAC:13, then Hash shall be SHA384, Z shall be 512, Q shall be 384, and XXKey shall be the first 384 bits of the MSK (which is derived from the IEEE 802.1X authentication), i.e., XXKey = L(MSK, 0, 384).

***Instruct the editor to modify section 11.3.4.2.2 as indicated:***

**11.3.4.2.2. Generation of the password element with ECC groups**

The blinding technique of determining whether a value, v , is a quadratic residue modulo a prime, p , is then:

*r*  = (random() modulo (*p* – 1) + 1

*num* = (*v* x *r* x *r* ) modulo p

 if (LSB(*r* ) = 1)

then

*num* = (*num* x *qr* ) modulo *p*

 if (LGR(*num* , *p* ) = 1)

then

*v* is a quadratic residue modulo p

fi

else

num = (*num* x *qnr* ) modulo *p*

 if (LGR(*num* , *p* ) == –1)

then

v is a quadratic residue modulo p

fi

fi

v is a quadratic non-residue modulo p

The values qr and qnr may be used for all loops in the hunting-and-pecking process but a new value for r must be generated each time a quadratic residue is checked.

**References:**