IEEE P802.11
Wireless LANs

|  |
| --- |
| Correcting a Mistake  |
| Date: 2015-06-25 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | Aruba Networks | 1322 Crossman avenue, Sunnyvale, California, United States of America | +1 408 227 4500 | dharkins at aruba networks dot com |
|  |  |  |  |  |

Abstract

A conditional statement in the 11-14/0640r1 was missed when that document was incorporated into the TGmc draft. As a result the new hunting-and-pecking loop for ECC groups would not be backwards compatible with IEEE Std 802.11-2012.

Thanks to Jouni Malinen for discovering this issue before it was too late.

In addition, it is possible for the legendre symbol to be zero and that would cause the quadratic residue and quadratic non-residue to not be what we want. While the patient is on the operating table let’s take care of this.

***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**

Algorithmically this process is described as follows:

*found* = 0;

*counter* = 1

*Length* = len(*p*)

*base* = *password*

do {

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

*base* || *counter*)

*pwd -value* = KDF-Length(*pwd -seed* , “SAE Hunting and Pecking”, *p*)

if (*pwd-value* < *p* )

then

if (*pwd-value*3 + a x *pwd-value* + b ) is a quadratic residue modulo *p*

then

if (*found* == 0)

then

*x* = *pwd-value*

*save* = *pwd-seed*

*found* = 1

*base* = new-random-number

 fi

fi

fi

*counter* = *counter* + 1

} while ((*counter* <= k ) or (*found* =0))

*y* = sqrt(*x*3 + a*x* + b ) modulo *p*

if LSB(*save*) = LSB(*y*)

then

PWE = (*x* , *y*)

else

PWE = (*x* , *p – y*)

fi

This technique involves creation of a quadratic residue, qr , and quadratic non-residue, qnr , prior to

beginning of the hunting-and-pecking loop. These values can be chosen at random by checking their

legendre symbol:

do {

*qr* = random() modulo *p*

while ( LGR(*qr* , *p*) is not equal to 1)

do {

*qnr* = random() modulo *p*

while ( LGR(*qnr* , *p*) is not equal to -1)

**References:**