IEEE P802.11
Wireless LANs

|  |
| --- |
| Cleanup of PKEX Text |
| Date: 2015-05-12 |
| 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 arubanetworks dot com |
|  |  |  |  |  |

Abstract

Incorporation of 11-15/0291r1 resulted in some section headers being incorrect. In addition, some textual cleanup of the description of PKEX makes it easier to understand.

***Instruct the editor to correct the header designation for the MIC element:***

**8.4.2.118 MIC element**

The MIC element provides message integrity to mesh peering Management frames and Public Key

Exchange management frames. The format of the MIC element is shown in Figure 8-485 (MIC element format).

***Instruct the editor to correct the headings and sub-headings for 8.6.16.7 as indicated:***

**8.6.16.7 Public Key Exchange Key Commit format**

**8.6.16.7.1 Public Key Exchange Key Commit frame self protection**

Protection of this frame is provided by encryption of the contents of the frame using a shared credential.

**8.6.16.7.2 Public Key Exchange Key Commit frame details**

The PKEX Key Commit frame is used to exchange public keys that have been encrypted by a shared credential.

The format of the PKEX Key Commit frame Action field is shown in Table 8-350 (PKEX Key Commit

frame Action field format).

***Instruct the editor to correct the headings and sub-headings for 8.6.16.8 as indicated:***

**8.6.16.8 Public Key Exchange Key Confirmation format**

**8.6.16.8.1 Public Key Exchange Key Confirmation self protection**

Protection of this frame is provided by using a secret key to generate a Message Integrity Code (MIC).

**8.6.16.8.2 Public Key Exchange Key Confirmation details**

The PKEX Key Confirmation frame is used confirm possession of a private key and knowledge of a shared

credential. The format of the PKEX Key Confirmation frame Action field is shown in Table 8-351 (PKEX

Key Confirmation frame Action field format)..

***Instruct the editor to modify section 11.6.12.2 as indicated:***

**11.6.12.2 PKEX overview**

PKEX is a variant of the encrypted key exchange (EKE). For a public key, P, and an encrypted public key,

C, encryption with key k, Ek(), and decryption with key k, Dk(), are defined as:

 Q = f(k)

C = Ek(P) = elem-op(P, Q)

P = Dk(C) = elem-op(C, inverse(Q))

Where where inverse() and elem-op() is defined in section 11.3.4 (Finite cyclic groups) and f() is a function that creates a secret element in the same group as P from a secret k.

**11.6.12.4.1 initial provisioning for PKEX**

If a STA does not have a public key to exchange, it shall generate one in a chosen finite cyclic group from the dot11RSNAConfigDLCGroup table. PKEX uses the same IANA registry to identify a group's domain parameter set as SAE. For interoperability purposes, a conformant STA shall support group nineteen (19), an ECC group defined over a 256-bit prime order field.

Prior to sending a PKEX message, both STAs shall be provisioned with a shared key/code/word/phrase, hereinafter a credential. It shall be interpreted as a UTF-8 string with no NULL termination. The credential shall be used to generate a password element, PWE per section 11.3.4.2.2 (for ECC groups) or 11.3.4.3.2 (for FFC groups), in the same group as the public key with the one minor change: the MAC addresses are removed from the pwd-seed value calculation in section 11.3.4.2.2 (for ECC groups) and 11.3.4.3.2 (for FFC groups) and the equation becomes:

pwd-seed = H(base || counter)

Once generated, PWE shall be converted into a STA-specific key, Q, used to encrypt a public key:

q = H(STA-MAC)

Q = scalar-op(q, PWE)

Where scalar-op() is defined in section 11.3.4 (Finite cyclic groups), and STA-MAC is the MAC address of the STA generating Q.

***Instruct the editor to modify section 11.6.12.4.2 as indicated:***

**11.6.12.4.2 Exchange of PKEX Key Commit messages**

A STA begins the PKEX protocol at anytime after generation ofQ. An AP STA shall not initiate PKEX but shall wait until a non-AP STA has initiated to it.

To begin the PKEX protocol a STA shall first generate a random nonce whose length is equal to the size of the digest of the hash algorithm used by PKEX, as defined in 11.6.12.3. It shall then encrypt its public key using Qin the technique defined in section 11.6.12.2.

Next, the STA determines whether it has sent a PKEX Key Commit message to the STA that transmitted the received message (the peer STA) or to the group address. If not, for example if the recipient is an AP STA, the STA shall generateQ, as defined above in 11.6.12.4.1, generate a PKEX Key Commit message, and transmit it to the peer STA. Otherwise, and in any case, the PKEX Key Commit message is processed:

1. The peer's nonce is retrieved from the Challenge Text field of the Challenge Text element in the received frame;
2. The encrypted public key is obtained by converting the octet string(s) to an element according to 11.3.7.2.5 (Octet string to element conversion). If conversion fails, the PKEX Commit message is silently discarded;
3. A peer-STA-specific decryption element, Q’ shall be generated in the same fashion as Q except using the peer STA’s MAC address:

q’ = H(peer-MAC)

Q’ = scalar-op(q’, PWE)

1. The encrypted public key is decrypted using Q’according to the decryption function definition in 11.6.12.2, to generate the peer’s public key PUB;
2. A shared element, S, is generated using scalar-op() from 11.3.4 (Finite Cyclic groups) with the private analog to the STAs public key, priv, and the peer STA's decrypted public key, PUB, and a secret value, s, is derived from S using function F() from section 11.3.4 (Finite cyclic groups):

S = scalar-op(priv, PUB) (1)

s = F(S) (2)

1. A key confirmation key, k, whose length, i, is the length of the digest produced by the hash function, is derived by first reducing the two nonces with the hash function used with PKEX and then using the result as a key with the KDF from section 11.6.1.7.2 (Key derivation function) with s and the label "PKEX Key Confirmation" as data:

x = Hash(min(STA-nonce, peer-nonce) || max(STA-nonce, peer-nonce)) (3)

k = KDF-i(x, s || "PKEX Key Confirmation") (4)

Where min() and max() operations for nonces are encoded as specified in 8.2.2 (Conventions).

When processing of the PKEX Key Commit message finishes, a STA transitions into the Exchange of

PKEX Key Confirmation messages.

**References:**