IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PKEX CID | | | | |
| Date: 2016-05-12 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE | 1322 Crossman avenue, Sunnyvale, California, United States of America | +1 408 555 1212 |  |
|  |  |  |  |  |

Abstract

This submission proposes a resolution for CID 20014 and cleans up some erroneous text in the definition of PKEX.

|  |  |  |  |
| --- | --- | --- | --- |
| CID | Comment | Proposed Change | Resolution |
| 20014 | IETF RFC 6090 does not appear to be properly sited in normative text (nor anywhere else). | Delete the reference | Accept |

*Discussion: The comment is valid. But there’s other problems….*

***Instruct the editor to modify section 2 as indicated:***

**2 Normative references**

*No CID.*

*Discussion: in looking to find the reference for RFC 6090, the most obvious place to look was PKEX. It was not there but upon inspection, it appears that PKEX was not updated correctly. Basically, the description of the encryption was changed to use a secret Q and decryption to use inverse(Q) but that was not fully implemented, also it was somewhat confusing to have P be a device’s public key and then to decrypt the peer’s key and call it P too. Peer keys are now “prime”, so a decrypted peer key is P’.*

***Instruct the editor to modify section 12.7.12.2 as indicated:***

**12.7.12.2 PKEX overview**

PKEX is a variant of the encrypted key exchange (EKE).Using a secret Q, a public key P is encrypted with E() to produce an enctypted public key, C, which is decrypted with D() to reproduce P:

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

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

where inverse() and elem-op() is defined in 11.3.4 (Finite cyclic groups).

The private analog to a device’s public key P is referred to as p. A peer’s received encrypted public key is C’ and its decrypted public key is P’.

The PKEX protocol uses a cryptographic hash function with the KDF from 11.6.1.7.2 (Key derivation function) as well as to distill entropy from the shared key/code/word/phrase. The particular hash function to use depends on the size of the prime, p, that defines the finite field in which the STA's public key is defined.

SHA-256: len(p) <= 256

SHA-384: 256 < len(p) <= 384

SHA-512: 384 < len(p)

For purposes of extensibility, PKEX is described as a true peer-to-peer protocol. This allows it to be used between a STA and AP in a role-based exchange as well as between two STAs directly without any sort of client/server roles.

**12.7.12.4.1 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 11.3.4.2.2 (Generation of the password element with ECC groups) (for ECC groups) or 11.3.4.3.2(Generation of the password element with FFC groups) (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 11.3.4.2.2 (Generation of the password element with ECC groups) (for ECC groups) and 11.3.4.3.2 (Generation of the password element with FFC groups) (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 11.3.4 (Finite cyclic groups), and STA-MAC is the MAC address of the STA generating Q.

Encrypt: C = elem-op(P, Q)

Decrypt: P = elem-op(C, inverse(Q))

**12.7.12.4.2 Exchange of PKEX Key Commit messages**

A STA begins the PKEX protocol at anytime after generation of Q. 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 12.7.12.3 (PKEX messages). It shall then encrypt its public key, P, using Q in the technique defined in 12.7.12.1 (General) to produce an encrypted public key, C: C=elem-op(P, Q).

Next, the PKEX Key Commit Message shall be generated in the format of Table 9-365a (PKEX Key Commit frame Action field format) in 9.6.16.7.2 (Public Key Exchange Key Commit frame details):

1. The STA's random nonce shall be copied into the Challenge Text field of the Challenge Text element, with the length being filled in appropriately;
2. The number from the IANA registry for the group in which the public key was created shall be copied into the Finite Cyclic Group field;
3. The encrypted public key shall be converted into (an) octet string(s) according to 11.3.7.2.4 (Element to octet string conversion).

If the STA knows the MAC address of the peer the PKEX Key Commit message shall be sent to that destination MAC address, otherwise it shall be sent to the group address. A STA that has initiated PKEX shall wait for receipt of a PKEX Key Commit from the peer. The STA may choose to retransmit the PKEX Key Commit message after a suitable waiting period of its own choosing and may choose to retransmit a limited number of times, of its own choosing, before abandoning PKEX. The waiting period and retransmit limit are not defined here because they have no effect on interoperability.

A STA that receives a PKEX Key Commit message that has not been provisioned with a credential shall silently drop the message.

Upon receipt of a PKEX Key Commit message the STA will check whether the finite cyclic group is acceptable. If not, the STA shall silently discard the message. If the group is acceptable the STA checks whether it has a public key in that group to exchange. If it does not, and does not wish to create such a public key it shall silently discard the message.

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 generate Q as defined in 12.7.12.4.1, 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 decryptioin 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, C’, is decrypted using Q' to produce the peer’s public key, P’, according to the decryption function definition in 12.7.12.2 (PKEX overview);
2. The decrypted public key P’ is then validated in a group-specific fashion as described in 5.6.2.3 of NIST SP 800-56A R2. If validation fails, PKEX terminates unsuccessfully
3. 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, p, and the peer STA's decrypted public key, P’, and a secret value, s, is derived from S using function F() from 11.3.4 (Finite cyclic groups):

S = scalar-op(p, P’)

s = F(S)

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 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))

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

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.

**12.7.12.4.3 Exchange of PKEX Key Confirm messages**

As soon as PKEY Key Commit message processing completes, a PKEX Key Confirm message is generated in the format of table Table 9-365b (PKEX Key Confirmation frame Action field format) in 9.6.16.8.2 (Public Key Exchange Key Confirmation details).

First, a key confirmation and integrity check is calculated by passing the key, k, and data consisting of a concatenation of the two unencrypted public keys and the STA's MAC address to the HMAC version of the hash function used by PKEX:

check = HMAC-Hash(k, P || P’ || STA-MAC)

where the public keys are converted into an octet string per 11.3.7.2.4 (Element to octet string conversion) prior to concatenation and passing to the HMAC. The value of check shall be copied into the MIC field of the PKEX Key Confirm message and the message transmitted to the peer whose MAC address is the transmitter of the received PKEY Key Commit message. The PKEX Key Confirm message shall not be a group addressed frame. The STA may choose to retransmit the PKEX Key Confirm message after a suitable waiting period of its own choosing and may choose to retransmit a limited number of times, of its own choosing, before abandoning PKEX. The waiting period and retransmit limit are not defined here because they have no effect on interoperability.

Upon receipt of a PKEX Key Confirm message from the peer, a verifier shall be generated based on the expected value of the MIC field of the received PKEX Key Confirm message:

verifier = HMAC-Hash(k, P’ || P || peer-MAC)

The verifier shall then be compared to the value in the MIC field of the received PKEX Key Confirm message. If they differ, the PKEX shall be silently aborted and all state information associated with this exchange shall be irretrievably deleted. Otherwise, PKEX shall be deemed to have completed successfully and the peer's public key can be trusted to be used in a subsequent authentication protocol. All state information other than the peer's MAC address and now-trusted public key shall be irretrievably deleted.

**References:**