IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Introducing PKEX | | | | |
| Date: 2015-02-27 | | | | |
| 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

This submission addresses CID 7267

***Instruct the editor to add the following references to section 2:***

**2. Normative references**

FIPS PUB 186-4-2013, Digital Signature Standard

ISO/IEC 1488-3:2006, Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms

***Instruct the editor to modify section 8.4.2.118 as indicated:***

**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).

|  |  |  |
| --- | --- | --- |
| Element ID | Length | MIC |

Octets: 1 1 variable

The Element ID and Length fields are defined in 8.4.2.1 (General).

When used by mesh networking, the MIC field is 16 octets and contains a message integrity code that is calculated over the mesh peering Management frame (as specified in 13.5 (Authenticated mesh peering exchange (AMPE))) and the mesh group key handshake frame (as specified in 13.6 (Mesh group key handshake)). When used by PKEX, the MIC field depends on the size of the digest of the hash function used by PKEX (see 11.6.12.1) and is calculated over the Public Key Exchange Commit frame (as specified in 11.6.12 (Authenticated Public Key Exchange).

***Instruct the editor to modify table 8-349 in section 8.6.16.1 as indicated:***

**8.6.16.1 Self-protected Action fields**

**Table 8-349—Self-protected Action field values**

|  |  |
| --- | --- |
| Self-protected Action field value | Description |
| 6 | Public Key Exchange Key Commit |
| 7 | Public Key Exchange Key Confirm |
| 8-255 | Reserved |

***Insruct the editor to create new section 8.6.16.7 and assign a table number to TBD-1 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 TBD-1 (PKEX Key Commit frame Action field format).

**Table TBD-1—PKEX Key Commit frame Action field format**

|  |  |  |
| --- | --- | --- |
| Order | Information | Notes |
| 1 | Category |  |
| 2 | Self-protected Action | Value of 6 from table 8-349 |
| 3 | Challenge Text | A random nonce |
| 4 | Finite Cyclic Group | An unsigned integer indicating a finite cyclic group as described in 11.3.4 (Finite Cyclic groups). |
| 5 | Element | A field element from a finite field encoded as described in 11.6.12.2 (PKEX Messages). |

The Category field is defined in 8.4.1.11 (Action field).

The Self-protected Action field is defined in 8.6.16.1 (Self-protected Action fields).

The Challenge Text element is defined in 8.4.2.8 (Challenge Text element).

The Finite Cyclic Group field is defined in 8.4.1.42 (Finite Cyclic Group field).

The Element field is defined in 8.4.1.40 (Element field).

***Instruct the editor to create section 8.6.16.8 and assign a table number to TBD-2 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 TBD-2 (PKEX Key Confirmation frame Action field format).

**Table TBD-2—PKEX Key Confirmation frame Action field format**

|  |  |  |
| --- | --- | --- |
| Order | Information | Notes |
| 1 | Category |  |
| 2 | Self-protected Action | Value of 7 from table 8-349 |
| 3 | MIC |  |

The Category field is defined in 8.4.1.11 (Action field).

The Self-protected Action field is defined in 8.6.16.1 (Self-protected Action fields).

The MIC field is defined in 8.4.2.118 (MIC field).

***Instruct the editor to create section 11.6.12, modifying TBD-1 and TBD-2 per assignment above, as indicated:***

**11.6.12 Authenticated Public Key Exchange**

Some authentication protocols for 802.11, like FILS and 802.1X/EAP, use certified public keys to validate signatures (created using the private analog to the certified public kcy) in order to authenticate the peer. Obtaining a certified public key typically entails gaining trust in a Certification Authority (CA), known as a trusted 3rd party, demonstrating an identity, providing a public key, and proving possession of the private analog to a public key. Only then does the CA construct, sign, and return a certificate. Provided that two parties go through the same steps to a common CA, or to CAs which trust a common hierarchical issuer, the two parties can use their certified public keys to authenticate. In some cases, this step and the Public Key Infrastructure (PKI) it implies, can be prohibitive.

**Note: A party typically enrols in a CA by constructing a PKCS#10 request and receiving, from the CA, a certificate in the form of a PKCS#7 response.**

An alternative is for two parties to exchange their uncertified public keys in a manner that proves possession of the private analog to the public key as well as provides a level of authentication to the exchange so each party has reason to trust the other party’s key. Once trust is gained in the public key and the peer proves possession of the private analog to that key, it can be used to verify signatures generated by the peer in service of an authentication protocol.

The Public Key Exchange (PKEX) is a protocol to provide that service. It is a simple exchange consisting of two request-response messages, four messages in total. PKEX uses a shared key/code/word/phrase and public key cryptography in order to achieve the following goals:

* The protocol will result in the exchange of trusted public keys or it will fail;
* A passive adversary is unable to subvert the exchange, insert any different public keys, learn the public keys, or learn the key/code/word/phrase shared by the two peers;
* An active adversary that does not know the shared key/code/word/phrase cannot successfully complete the exchange; and,
* An attacker is not able to perform an off-line dictionary attack against PKEX in order to determine either public key or to determine the shared key/code/word/phrase.

Due to the nature of the exchange, only public keys suitable for DSA (specified in FIPS 186-4) or ECDSA (specified in ISO/IEC 14888-3) can be exchanged using PKEX, and a non-AP STA cannot engage in multiple, simultaneous PEX exchanges with more than one peer.

**11.6.12.1 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:

C = Ek(P) = elem-op(P, ***f***(*k*))

P = Dk(C) = elem-op(C, -***f***(*k*))

Where where 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*.

The PKEX protocol uses a cryptographic hash function with the KDF from section 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.

**11.6.12.2 PKEX Messages**

PKEX messages are self-protected action frames sent from one STA to another. Their format is described in sections 8.6.16.7 and 8.6.16.8.

**11.6.12.3 PKEX Protocol**

**11.6.12.3.1 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 an ASCII 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*)

**11.6.12.3.2 Exchange of PKEX Key Commit Messages**

A STA begins the PKEX protocol at anytime after generation of *PWE*. 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 *PWE* in the technique defined in section 11.6.12.1.

Next, the PKEX Key Commit Message shall be generated in the format of table TBD-1 in 8.6.17.7.2:

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 PEX 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 *PWE*, if necessary, as defined above, 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. The encrypted public key is decrypted using *PWE* according to the decryption function definition in 11.6.12.1;
4. 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*)

*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 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*))

*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.

**11.6.12.3.3 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 TBD-2 in 8.6.16.7.3.

First, a key confirmation and integrity check is calculated by passing the key, k, and data consisting of a concatentation 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*, *STA-pubkey* || *peer-pubkey* || *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 concatentation 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*, *peer-pubkey* || *STA-pubkey* || *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 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 other than the peer’s MAC address and now-trusted public key shall be irretrievabley deleted.

**References:**