IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mods to PKEX | | | | |
| Date: 2016-8-16 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Dan Harkins | HPE | 1322 Crossman avenue, Sunnyvale, California, United States of America | +1 415 555 1212 | First name dot last name at hpe dot com |
|  |  |  |  |  |

Abstract

PKEX should hash the encrypted public keys exchanged over the air and the MAC addresses into the shared key in order to deal with related key attacks. Also, the references need updating.

**Discussion**

One way that security protocols are analysed is by assuming a powerful adversary that can inspect, modify, and transmit all messages exchanged. The adversary starts the protocol by instructing one participant to begin and it can passively pass messages back and forth between the participants or it can actively modify or substitute messages in an attempt to break the security guarantees of the protocol. One of the many powers this adversary has is the power instruct the participants in the exchange to “reveal” derived session keys. Such an instruction terminates the protocol but if it allows the adversary to undermine the security goals of the protocol, an attack is successful.

A related key attack is one where an adversary creates different sessions between the protocol participants that result in keys that are related in a predictable manner. Such an attack can be launched against PKEX, as written, by an adversary performing an active attack as a man-in-the-middle and modifying the encrypted public key the STA sends but faithfully relaying the encrypted public key the peer sends. For instance, assuming finite field PKEX, C = P \* PWE Hash(STA-MAC). If the attacker multiplies C by gr where g is the generator of the group and r is chosen by the attacker and sends the modified C to the peer, the attacker will know that the keys generated by the STA (Ksta) and the peer (Kpeer) will be related. Namely, Kpeer = Ksta \* (C’/ PWE Hash(STA-MAC)) r. Therefore, if the adversary is able to determine the session keys (again, this is assumed to be a capability of the adversary) it can perform an off-line dictionary attack to determine the password used to generate PWE. This voids one of the goals of PKEX: “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.” (see 12.7.12.1).

By including the encrypted public keys in the key derivation step this attack is foiled. Including the MAC addresses will harden the session key and bind it to the only identities PKEX has. Including both MAC addresses in the confirmation check (and verifier) binds the received public keys to the identities as well.

***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 encrypted 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 12.4.4 (Finite cyclic groups).

The PKEX protocol uses a cryptographic hash function with the KDF from 12.7.1.7.2 (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.

***Instruct the editor to modify section 12.7.12.4.1 as indicated:***

**12.7.12.4.1 Initial provisioning for PKEX**

A PKEX message can be sent only when both STAs have been provisioned with a shared key/code/word/phrase (hereinafter referred to as “a credential”). The credential shall be interpreted as a UTF-8 string with no NULL termination. The credential shall be used to generate a password element, PWE per 12.4.4.2.2 (Generation of the password element with ECC groups) (for ECC groups) or 12.4.4.3.2(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 12.4.4.2.2 (Generation of the password element with ECC groups) (for ECC groups) and 12.4.4.3.2 (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 12.4.4 (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))

***Instruct the editor to modify 12.7.12.4.2 as indicated:***

**12.7.12.4.2 Exchange of PKEX Key Commit messages**

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 12.4.7.2.4 (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 the PKEX Key Commit message 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 discard 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 (Initial provisioning for PKEX), 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, C’, is obtained by converting the octet string(s) to an element according to 12.4.7.2.5 (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):

P’ = elem-op(C’, inverse(Q’))

1. 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
2. A shared element, S, is generated using scalar-op() from 12.4.4 (Finite Cyclic groups) with the private analog to the STAs public key, p, and the peer STA's decrypted public key, P':

S = scalar-op(p, P')

1. A key confirmation key, k, whose length, is the length of the digest produced by the hash function, is derived from S using the KDF from 12.7.1.7.2 (Key derivation function) with the label “PKEX Key Confirmation” and a conditional context consisting of the encrypted public keys and the MAC addresses::

if (min(STA-nonce, peer-nonce) == STA-nonce)

x = Hash(peer-nonce ||STA-nonce)

k = KDF-Hash-Length(x,"PKEX Key Confirmation", C’ || C || peer-MAC || STA-MAC || F(S))

else

x = Hash(STA-nonce || peer-nonce)

k = KDF-Hash-Length(x, “PKEX Key Confirmation”, C || C’ || STA-MAC || peer-MAC || F(S))

endif

where min() and max() operations for nonces are encoded as specified in 8.2.2 (Conventions), Hash is the hash algorithm specified in 12.7.12.2, KDF-Hash-Length is the key derivation function defined in 12.7.1.7.2 (Key derivation function (KDF)) with Length equal to the size, in bits, of the digest produced by Hash. F() is the element-to-scalar function defined in 12.4.4 (Finite cyclic groups). STA-nonce and peer-nonce are the nonces produced by the STA and peer, respectively, as part of the PKEX protocol and STA-MAC and peer-MAC are the MAC addresses used by the STA and peer, respectively, to execute the PKEX protocol.

***Instruct the editor to modify section 12.7.12.4.3 as indicated:***

**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 two MAC addresses to the HMAC version of the hash function used by PKEX:

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

where the public keys are converted into a octet strings per 12.4.7.2.4 (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 || STA-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 completes 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:**