IEEE P802.11
Wireless LANs

|  |
| --- |
| Clarification of AP PeerKey |
| Date: 2013-05-10 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Henry Ptasinski | Self |  |  | henry at logout dot com |
| Alex Ashley | Self | 23 Mill Street, Redhill, Surrey, UK |  | Alex dot Ashley at outlook dot com |
| Dan Harkins | Aruba Networks | 1322 Crossman ave, Sunnyvale, CA | +1 408 227 4500 | dharkins at aruba networks dot com |

Abstract

This submission addresses CIDs 1709, 1710, and 1711

***Instruct the editor to modify section 8.5.8.24 as indicated:***

**8.5.8.24 Public key frame**

The Public Key frame is transmitted by an AP to provide its public key to peer APs and to request the peer’s

public key. The format of the Public Key frame body is defined in Figure 8-479 (Public Key frame body

format).

Public Key

(optional)

Group

Request

 Type

Action

Category

 Octets 1 1 1 1 variable

The Category field is set to the value indicating a Public Action frame, as specified in Table 8-39 (Category

values).

The Action field is set to the value specified in Table 8-220 (Public Action field values) for a Public Key

frame.

The Request Type field is set to a number to identify the usage mode of this frame. The Request Types are

shown in Table 8-234 (Request Type definitions).

|  |  |
| --- | --- |
|  Request Type |  Usage mode |
|  Request |  0 |
|  Response |  1 |
|  NAK |  2 |
|  Reserved |  3-255 |

The Request Type field is set to “Request” to indicate that a public key is being requested from a peer AP.

The Request Type field is set to “Response” to indicate that this frame is in response to a Public Key frame.

The Request Type field is set to “NAK” to indicate rejection of a received Public Key frame.

The Group field is used to indicate which cryptographic group was used when generating the public key and

is defined in 8.4.1.42 (Finite Cyclic Group field), or the preferred group when the Request Type field is set to “NAK”.

When the Request Type field is set to “NAK”, the Public Key field is not included. In all other cases, the Public Key field contains the public key of the AP that is sending this Public Key frame and is defined in 8.4.1.39 (Scalar field)

The use of the Public Key frame is described in 11.10 (AP PeerKey support).

***Instruct the editor to modify section 11.10.1 and 11.10.2 as indicated:***

**11.10.1 AP PeerKey overview**

The AP PeerKey protocol provides session identification and data confidentiality for an AP-to-AP

connection. An AP PeerKey association is composed of a Mesh PMKSA and a Mesh TKSA.

AP PeerKey uses various functions and data to accomplish its task and assumes certain properties about

each function as follows:

* H is an “extractor” function (see IETF RFC 5869) that concentrates potentially dispersed entropy from an input to create an output that is a cryptographically strong, pseudorandom key. This function takes as input a non-secret “salt” and a secret input and produces a fixed-length output.
* A finite cyclic group is negotiated for which solving the discrete logarithm problem is computationally infeasible.

When used with AKM 10 from Table 8-102 (AKM suite selectors) to indicate AP PeerKey, H shall be

instantiated as HMAC-SHA256:

H(salt, ikm) = HMAC-SHA256(salt, ikm)

Other instantiations of function H require creation of a new AKM identifier.

**11.10.2 AP PeerKey protocol**

AP PeerKey uses the same discrete logarithm cryptography as SAE (as described in 11.3 (Authentication

using a password)) to achieve key agreement. Each party to the exchange has a public and private key with

respect to a particular set of domain parameters that define a finite cyclic group. Groups may be based on

elliptic curve cryptography (ECC) or finite field cryptography (FFC). Each component of a group is referred

to as an “element.” Groups are negotiated using an identifying number from a repository maintained by

IANA as “Group Description” attributes for IETF RFC 2409 (IKE) [B17][B29]. The repository maps an

identifying number to a complete set of domain parameters for the particular group. For the purpose of

interoperability, APs that have dot11ProtectedHCCATXOPNegotiationImplemented true or

dot11ProtectedQLoadReportImplemented true shall support group nineteen (19), an ECC group defined

over a 256-bit prime order field.

AP PeerKey uses one arithmetic operator that takes one element and one scalar value to produce another

element (called the “scalar operation”). The convention used here is to represent group elements in

uppercase bold italic and scalar values in lowercase italic. The scalar operation takes an element and a scalar

and is denoted scalar-op(**x**,**Y**).

The private key **d** shall be chosen randomly so that 1 < **d** < **r**, where **r** is the order of the group. The public

key **Q** shall be produced using Equation (11-1).

**Q** = scalar-op(**d**, **G**) (11-1)

where

 **G** is the generator (also known as the base point) of the group

An AP for which dot11ProtectedTXOPNegotiationActivated is true or dot11ProtectedQLoadReportActivated is true shall support at least one public key from cyclic group nineteen and may support multiple public keys from multiple cyclic groups. An AP that supports the Multiple BSSID capability and has dot11ProtectedTXOPNegotiationActivated true or dot11ProtectedQLoadReportActivated true may use one public key across multiple BSSIDs, or it may choose to generate a public key for each supported BSSID.

The AP Peerkey protocol consists of an exchange of public keys from an AP and a peer AP. An AP requests the public key of a peer AP by sending a Public Key frame with the Request Type field set to “Request.” This frame contains the public key of the initiating AP. The initiating AP awaits a response to its request. The peer AP responds to the “Request” from the initiating AP with a Public Key frame with the Request Type field set to “Response”, indicating an acceptable group was sent by the initiating AP, or set to “NAK” indicating that the group sent by the initiating AP was unacceptable.

It is possible for multiple runs of the AP peerkey protocol to be performed by an AP but there shall be only one instance at a time of the AP peerkey protocol to a peer AP for a particular group. The state associated with each run of the AP peerkey protocol is the group and the two public keys of the participating APs.

An AP for which dot11ProtectedTXOPNegotiationActivated is true or dot11ProtectedQLoadReportActivated is true shall reply to a Public Key frame An AP that has both dot11ProtectedTXOPNegotiationActivated is false and dot11ProtectedQLoadReportActivated is false shall drop all received Public Key frames. If the Group field in the public key request is a group that is supported by the responding AP, the AP shall reply with a public key of the same group as the request, generating such a key pair if required, and setting the Request Type field to “Response”. The receiving AP shall generate a PMK and a Mesh PMKSA, see below, and terminate the Peerkey protocol. If the group field in the public key request is not supported by the responding AP, the responding AP shall respond with a Public Key frame, setting the Request Type to “NAK” and the group set to a group acceptable by the receiving AP.

If the initiating AP does not receive a response to its request after five (5) seconds, it should retransmit its request. The initiating AP should attempt such retransmission a minimum of five (5) times.

If the initiating AP receives Public Key frame from a peer AP with the Request Type field set to “NAK” the initiating AP inspects the group field in the received “NAK”. If the group is acceptable, the initiating AP should initiate to the peer AP using the indicated group. If the group is not acceptable, the AP may choose to initiate to the peer AP using a different group from the dot11RSNAConfigDLCGroup table. Receipt of a Public Key frame from a peer AP with the Request Type field set to “NAK” terminates the Peerkey protocol.

If the initiating AP receives a Public Key frame from a peer AP with the Request Type field set to “Response” and the group set to the value in the initiating AP’s “Request”, the initiating AP shall generate a PMK and a Mesh PMKSA. This terminates the Peerkey protocol.

If the initiating AP receives a Public Key frame from a peer AP with the Request Type field set to “Request” prior to receiving a “Response” it checks the Group field in the request. If the Group in the received “Request”” is the same as the Group the initiating AP sent it its request, the received “Request” is treated as a “Response”—the AP shall generate a PMK and a Mesh PMKSA, see below, and terminate the Peerkey protocol. If the Group field differs and the group indicated is supported, the AP shall respond to the peer AP by replying with a public key from that group, generating a key pair if required, and setting the Request Type field to “Response”. The AP shall generate a PMK and a Mesh PMKSA, see below, and terminate this run of the Peerkey protocol.

Once the AP and peer AP have exchanged public keys from the same finite cyclic group they can compute

the Diffie-Hellman shared secret for an AP-to-AP peer link using scalar-op() and function F from 11.3.4

(Finite cyclic groups):

**k** = F(scalor-op(**d**, Q**p**)) (11-2)

where

**d** is the private key of the AP that is calculating **k**

 Q**p** is the public key of the peer AP

Entropy of the shared secret shall then be extracted using function H to produce **keyseed** using

Equation (11-3).

**keyseed** = H(<0>32, **k**) (11-3)

where

 <0>32 is 32 octets with a value of zero

The PMK shall be derived from the keyseed using the key derivation function (KDF) from 11.6.1.7.2 (Key derivation function

(KDF)) using Equation (11-4) and the PMKID shall be derived according to (11-5).

PMK = KDF-256(keyseed , “AP Peerkey Protocol”,

 0x00 || Max(LOCAL-MAC, PEER-MAC) || Min(LOCAL-MAC, PEER-MAC) ) (11-4)

 PMKID = SHA-256-128(Q1 || Q2 ||

 Max(LOCAL-MAC, PEER-MAC) ||

 Min(LOCAL-MAC, PEER-MAC)) (11-5)

where

0x00 is a single octet with a value of zero

LOCAL-MAC is the AP’s BSSID

PEER-MAC is the peer AP’s BSSID

Q1 is the public key used by AP1 in the AP PeerKey protocol encoded as an octet stream using the Element to Octet string conversion from section 11.3.7.2.3.

Q2 is thepublic key used by AP2 in the AP PeerKey protocol encoded as an octet stream using the Element to Octet string conversion from section 11.3.7.2.3.

AP1 is the AP whose MAC address equals Max(LOCAL-MAC, PEER-MAC)

AP2 is the AP whose MAC address equals Min(LOCAL-MAC, PEER-MAC)

The Min and Max operations for IEEE 802 addresses are calculated by converting the address to a

positive integer by treating the first transmitted octet as the most significant octet of the integer.

Keyseed shall be irretrievably destroyed after the PMK is generated.

The lifetime of the Mesh PMKSA shall be set to the value dot11RSNAConfigPMKLifetime.

Upon creation of the PMK, an AEK shall be created per 13.5.7 (Keys and key derivation algorithm for the authenticated mesh peering exchange (AMPE)). The Mesh PMKSA for this instance of the AP PeerKey protocol shall then be created using the AP’s BSSID as the STA’s MAC address, the peer AP’s BSSID as the peer STA’s MAC address, the AEK, the lifetime, and the PMKID.

Upon creation of the Mesh PMKSA, the APME protocol (as defined in 13.5 (Authenticated mesh peering exchange (AMPE)) shall be used to prove possession of the PMK (and implicitly the private key that corresponds to the peer’s public key) and generate the Mesh PTKSA.

Note: it is possible for two APs which simultaneously initiated to each other with different, but acceptable, groups to end up with two Mesh PMKSAs. In this unlikely case, the Mesh PMKSA from a group with the largest prime in its domain parameter set shall be used with the AMPE protocol. The other Mesh PMKSA shall be deleted.

If the AMPE protocolcompletes successfully, Protected HCCA TXOP Advertisement frames

and Protected HCCA TXOP Response frames may be used in the HCCA TXOP negotiation procedures, as

defined in 10.28.3 (HCCA TXOP negotiation) using the MTK from the Mesh TKSA. If the AMPE procedure completes successfully, Protected QLoad Request frames and Protected QLoad Report frames may be used in the QLoad report procedures, as defined in 10.28.2 (QLoad Report element) using the MTK from the Mesh TKSA. If the AMPE protocol fails, the peer’s public key, PMK, and PMKSA shall be deleted.

***Change the second paragraph of 13.5.1 as indicated:***

**13.5 Authenticated mesh peering exchange (AMPE)**

**13.5.1 Overview**

The AMPE is also used to establish an authenticated peering between two APs that support the AP Peerkey

protocol (as defined in 11.10 (AP PeerKey support)) under the assumption that a PMK and PMKID have already

been established before the initiation of the AMPE exchange.

***Change the second paragraph of 13.5.2.1 as indicated***

**13.5.2 Security capabilities selection**

**13.5.2.1 Instance Pairwise Cipher Suite selection**

I

f the pairwise cipher suite has not been selected, a STA shall attempt to reach the agreement on the pairwise cipher suite using the following procedure in four steps:

1. The STA shall announce the list of pairwise cipher suites it supports using an ordered list in the RSNE in the Mesh Peering Open frame. The first value in the list is the STA’s most preferred cipher suite, and the last value the least preferred.
2. If the STA receives a Mesh Peering Open frame from the candidate peer STA, the STA shall make its decision on the selected pairwise cipher suite based on the intersection of its own ordered list and the received ordered list.
	1. If the intersection is empty, the pairwise cipher suite selection fails and the STA generates the failure reason code MESH-INVALID-SECURITY-CAPABILITY and then takes the corresponding actions specified in 13.5.6 (AMPE finite state machine).
	2. If the intersection contains more than one value, the selected cipher suite shall be the entry in the intersection list most preferred by the STA that has the largest MAC address in the lexicographic ordering.
3. If the STA receives a Mesh Peering Confirm frame from the candidate peer STA before receiving a Mesh Peering Open frame, the STA shall verify that it supports the pairwise cipher suite chosen by the candidate peer STA. Otherwise, the selection fails and the STA shall generate the failure reason code MESH-INVALID-SECURITY-CAPABILITY.

Furthermore, upon receiving a Mesh Peering Open frame, the STA shall verify that the accepted selected pairwise cipher suite matches the pairwise cipher suite chosen in step b). If they do not match, the selection fails and the STA shall generate the failure reason code MESHINVALID-SECURITY-CAPABILITY. Otherwise, the pairwise cipher suite selection succeeds, and the STA shall proceed to step d).

1. If the STA is generating a Mesh Peering Confirm frame, it shall set the Selected Pairwise Cipher Suite to the selected pairwise cipher suite upon successful pairwise cipher suite selection

***Change 13.5.5.2 as indicated:***

**13.5.5.2 Mesh peering open for AMPE**

**13.5.5.2.1 Generating Mesh Peering Open frames for AMPE**

In addition to contents for establishing a mesh peering as specified in 13.3.6.1 (Generating Mesh Peering Open frames), the Mesh Peering Open frame, when used for the AMPE, shall contain the following:

* In the Mesh Peering Management element, the Mesh Peering Protocol Identifier shall be set to 1 “authenticated mesh peering exchange protocol.”
* In the Mesh Peering Management element, the Chosen PMK field shall be set to PMKID that identifies the mesh PMKSA the mesh STA established with the candidate peer mesh STA.
* The RSNE shall be identical to the RSNE in the STA’s Beacon and Probe Response frames.
* In the Authenticated Mesh Peering Exchange element:
	+ The Selected Pairwise Cipher Suite field shall be set to the first cipher suite selector in the Pairwise Cipher Suite List field in RSNE.
	+ The Local Nonce field shall be set to the localNonce value generated by the mesh STA for identifying the current mesh peering instance.
	+ The Peer Nonce field shall be set to 0.
	+ If dot11MeshSecurityActivated is true, the GTKdata field shall be present and shall contain the data for the mesh STA’s MGTK. The GTKdata field shall not be present when AMPE is being used as part of the APPeerKey protocol (11.10.2). The components of the GTKdata field are specified in 13.5.4 (Distribution of group transient keys in an MBSS).

The Mesh Peering Open frame shall be protected using AES-SIV as specified in 13.5.3 (Construction and processing AES-SIV-protected Mesh Peering Management frames).

***Change the second paragraph of 13.5.1 as indicated:***

**13.5.5.3.2 Processing Mesh Peering Confirm frames for AMPE**

If AES-SIV returns plaintext, the following operations shall be performed in order:

1. The Selected Pairwise Cipher Suite is checked. If the security capability selection has been done and the received value from Chosen Pairwise Cipher Suite field is not the same as the agreed pairwise cipher suite, the STA shall reject the received frame and the CNF\_RJCT event is invoked to the corresponding AMPE finite state machine with the failure reason code MESH-INVALIDSECURITY-CAPABILITY.
2. If dot11MeshSecurityActivated is true the Group Cipher Suite is checked. If the received group cipher suite is not supported by the mesh STA, the mesh STA shall reject the received Mesh Peering Confirm frame and the CNF\_RJCT event is invoked to the corresponding AMPE finite state machine with the failure reason code MESH-INVALID-SECURITY-CAPABILITY.

***Modify section 13.5.7 as indicated:***

**13.5.7 Keys and key derivation algorithm for the authenticated mesh peering exchange (AMPE)**

To execute the AMPE and mesh group key handshake with a candidate peer STA, the localSTA shall

derive an authenticated encryption key (AEK) and a mesh temporal key (MTK) using the PMK it shares

with the candidate peer mesh STA.

The AEK is derived statically from the shared PMK. The MTK is derived from the shared PMK and

dynamic information provided by the mesh STA and candidate peer mesh STA.

The AEK is mutually derived by the local STA and the peer STA once a new PMK has been

selected. The AEK shall be derived from the PMK by

AEK 🡨 KDF-256(PMK, “AEK Derivation”, Selected AKM Suite ||

min(localMAC, peerMAC) || max(localMAC, peerMAC)).

The temporal key (MTK) shall be derived from the PMK by

MTK 🡨 KDF-X(PMK, “Temporal Key Derivation”, min(localNonce, peerNonce) ||

max(localNonce, peerNonce) || min(localLinkID, peerLinkID) ||

max(localLinkID, peerLinkID) || Selected AKM Suite ||

min(localMAC, peerMAC) || max(localMAC, peerMAC)).

CCMP uses X = 128. The “min” and “max” operations for IEEE 802 addresses are with the address

converted to a positive integer, treating the first transmitted octet as the most significant octet of the integer

as specified in 11.6.1.3 (Pairwise key hierarchy). The min and max operations for nonces are with the nonces

treated as positive integers converted as specified in 8.2.2 (Conventions).

The MTK is used to protect communications between two peer STAs. The local STA and peer

STA derive an MTK per peering instance and may rekey the MTK using AMPE.

**References:**