# PKM version 3

PKM version 3 is based on PKM version 2. It provides limited cryptographic schemes that are approved by all security standards and reduces key exchange messaging over the network.

## Architecture

PKMv3 has two layers:

a) An encapsulation protocol for securing packet data across the 802.16 network. This defines a set of supported *cryptographic suites*, and the rules for applying those algorithms to a MAC PDU payload.

b) A key management protocol providing the secure distribution of keying data between the Authenticator (BS) and the supplicant (SS) and optionally an authentication server. Through this key management protocol, the SS and the BS synchronize keying data; in addition, the BS uses the protocol to enforce conditional access to network services.

The security components of the system are:

—Authentication Processing with optional EAP Encapsulation/Decapsulation: This stack uses TLS 1.3 [RFC 8446] authentication and key exchange layer, with optional EAP-TLS 1.3 [RFC 9190] wrapping to provides the interface with an authentication server.

— Control Message Processing: This stack processes the various PKM-related MAC messages. The MAC messages in this PKM version are partial to the PKMv2 set of messages.

— Message Authentication Processing: This stack uses HMAC to execute message authentication function.

— Authorization/SA Control: This stack controls the authorization state machine and the traffic encryption key state machine.

— Traffic Data Encryption: This stack encrypts or decrypts the traffic data and executes the authentication function for the traffic data. This component is identical to PKMv2 but with the use of TLS 1.3 cryptographic suites.

### Key management protocol

The PKMv3 protocol enforces mutual authentication between BS and MS. It also supports periodic reauthentication/reauthorization and key refresh. The authentication process is based on X.509 digital certificates [IETF RFC 3280] public-key encryption algorithm supported in TLS 1.3. The authentication protocol establishes a shared secret (i.e., the AK) between the SS and the BS. The shared secret is then used to derive subsequent keys such as TEKs. This two-tiered mechanism for key distribution permits refreshing of TEKs without incurring the overhead of computation-intensive operations.

An authentication server mutually authenticates both BS and SS during the initial authorization exchange. The BS and SS presents their credentials, which shall be a unique X.509 digital certificate issued by the operator-specified credential.

The BS associates an SS’s authenticated identity the data services that subscriber is authorized to access.

Since the authentication server authenticates both BS and SS, it may protect against an attacker employing a cloned SS or clone BS that masquerades as a legitimate SS or BS.

The traffic key management portion of the PKM protocol adheres to the TLS client/server model, where the SS (a TLS “client”) requests keying material and the BS (a TLS “server”) responds to those requests. This model ensures that individual SS clients receive only keying material for which they are authorized.

## Authentication

PKMv3 authentication is based on TLSv1.3. When an external authentication server exists, EAP-TLSv1.3 [RFC-9190] is used, otherwise the BS acts the TLS server while the SS acts as TLS client. EAP-TLS specifies an EAP authentication method with certificate-based mutual authentication utilizing the TLS handshake protocol for cryptographic algorithms and protocol version negotiation and establishment of shared secret keying material.

The key exchange is protected by the TLS key exchange algorithm Diffie-Hellman over either finite fields or elliptic curves (EC)DHE. Authentication can happen via asymmetric cryptography such as RSA, Elliptic Curve Digital Signature Algorithm (ECDSA), or the Edwards-Curve Digital Signature Algorithm (EdDSA), or a symmetric pre-shared key (PSK).

### EAP Encapsulation

PKMv3 uses the *PKMv2 EAP Transfer* (6.3.2.3.9.16) to encapsulate EAP messages. This message is bidirectional and used for transmission of EAP packet. This message is sent unprotected in “Not Authenticated” state. In Authenticated state, HMAC Digest and Key Sequence Number attributes shall be included in the message.

### TLS Encapsulation

When PKMv3 uses the *PKMv3 TLS Transfer* (6.3.2.3.9.33) to encapsulate TLS messages. This message is bidirectional and used for transmission of EAP packet. This message is sent unprotected in “Not Authenticated” state. In Authenticated state, HMAC Digest and Key Sequence Number attributes shall be included in the message.

### TLS Authentication Usage

In PKMv3, mutual TLS authentication is required. Other TLS 1.3 options may be used. The following extensions of TLS should be considered:

ClientHello. The clientHello message sent from the SS to the BS. It may include the key-share extension, with the SS cryptographic parameters according to the value of Authorization Policy Support field included in the SBC-REQ/RSP messages.

ServerHello message is sent from BS to the SS in response to ClientHello. It may include the key-share extension with the BS cryptographic parameters. The message must include the following extensions: CertificateRequest extension, Certificate and CertificateVerify.

The SS TLS response to the BS ServerHello message must include the Certificate and CertificateVerify extensions.

## State Machines

### Authorization State Machine

The authorization state machine for PKMv3 is detailed in the subclauses below.

#### Stopped State:

This is the initial state of the FSM. Nothing is done in this state.

#### Not Authenticated State

The TLS/EAP-TLS authentication is in process. Transition from another state into this state triggers the initialization of the authentication process. In this state the SS can only send management messages. No data can be transmitted at this state. A failure in the authentication process shall change the Auth-FSM state to *Stopped*.

Upon completing the authentication process, the MS derives the PMK and AK keys and the HMAC keys and sends *PKMv2 SA-TEK-Request* message to the BS, with the following attributes: MS\_Random, Key Sequence Number, AKID and HMAC Digest. This action triggers the change of state to *SA-TEK-Response-Wait* state.

Note that PKMv2 SA-TEK-Challenge message is not used in PKMv3, as the AK is derived from PMK.

#### SA-TEK-Response Wait

The SS has sent a PKMv2 SA-TEK-Request and waits for a PKMv2 SA-TEK-Response message in this state. If it does not receive a PKMv2 SA-TEK-Response message within SATEK Timer, the SS may resend the message up to SATEKRequestMaxResends times. If SATEKRequestMaxResends counter expires, the SS switches to *Stopped* state. Any PKMv2 SA-TEK-Request or SA-TEK-Response messages with invalid HMAC Digest or without HMAC Digest are discarded.

Upon receiving PKMv2 SA-TEK-Request, the BS shall respond with PKMv2 SA-TEK-Response with the following attributes: MS\_Random, BS\_Random, Key Sequence Number, AKID, SA-Descriptors (one or more) and HMAC Digest.

Receiving a PKMv2 SA-TEK-Response shall trigger the MS to switch to *Authenticated* state.

Note that in PKMv3, the PKMv2 SA-TEK-Request and PKMv2 SA-TEK-Response messages are sent on initial network only and not on reauthentication or rekeying.

#### Authenticated State

The authentication process has successfully completed a valid PMK context and AK context derived from the TLS master secret or the EAP MSK. Transition to this state triggers a Rekey event in all TEK FSMs. Upon entering this state, the SS shall start a Reauthentication timer. On Reauthentication timeout, the SS shall initiate a new EAP authentication process. If the reauthentication fails, the SS shall switch its state to Stopped. If the reauthentication is successful, the SS shall send a rekey event to all its TEK FSMs.

If the SS has a valid AK context, all the management messages with Basic CID or Primary Management CID should be sent with HMAC Digest. It should be discarded if the messages do not have a valid HMAC Digest. In this state the SS may hold two AK contexts: the old AK context and the new AK context which is created during reauthentication. The old AK context is deleted after AK grace time.

### TEK State Machine

Upon achieving authentication, an SS starts a separate TEK state machine for each of the SAIDs identified in the Authorization Reply message for which data traffic encryption is provisioned for. Each TEK state machine operating within the SS is responsible for generating and managing the keying material associated with its respective SAID. TEK state machines periodically send Key Request messages to the BS, requesting a refresh of keying material for their respective SAIDs. The BS responds to a Key Request with a Key Reply message containing the BS’s active keying information (but not the actual keys) for a specific SAID.

Note that at all times the BS maintains two diversity sets of keying material per SAID. The lifetimes of the two generations overlap so that each generation becomes active halfway through the life of its predecessor and expires halfway through the life of its successor.

A TEK state machine remains active as long as

a) The SS is authorized and has a valid AK, and

b) The SS is authorized to participate in that particular SA, i.e., the BS continues to provide fresh

keying material during rekey cycles.

The TEK state machine for PKMv3 is detailed in the subclauses below.

#### Start State

This is the initial state of the FSM. No resources are assigned to or used by the FSM in this state—e.g., all timers are off, and no processing is scheduled.

#### Op (operational) Wait State

The TEK state machine has sent its initial request (PKMv2 Key-Request) for SAID generation and is waiting for a reply from the BS. The PKMv2 Key-Request message is sent with the following attributes: current AK Sequence Number, SAID (or GSAID for multicast), Nonce generated by the SS, and HMAC Digest calculated with the HMAC key that was derived from current AK. Upon receiving the PKMv2 Key-Request message, the BS shall validate the HMAC of the message. If the request is for standard (unicast) key the BS shall derive the new TEK as described in ‎19.4.5 and send a PKMv2 Key-Reply message to the SS with Key Sequence, SAID and HMAC Digest attributes. In the case of multicast service key request (GKEK or GTEK) the BS shall act as in PKMv2.

Note: unlike PKMv2, unicast TEK key materials are not sent over the air.

#### Op Reauth (reauthorize) Wait State

The wait state the TEK state machine is placed in if it does not have valid keying material while the Authorization state machine is in the middle of a reauthorization cycle.

#### Operational State

Upon receiving a PKMv2 Key-Reply from the BS, the SS shall derive the new TEK as described in ‎19.4.5. The TEK FSM shall start a TEK Refresh Timer for the TEK lifetime period.

#### Rekey Wait State

Upon TEK Refresh Timer expiration, the SS shall send a PKMv2 Key-Request for this SAID. The PKMv2 Key-Request message is sent with the following attributes: current AK Sequence Number, SAID (or GSAID for multicast), Nonce generated by the SS, and HMAC Digest calculated with the HMAC key that was derived from current AK. Upon receiving the PKMv2 Key-Request message, the BS shall validate the HMAC of the message. If the request is for standard (unicast) key the BS shall derive the new TEK as described in ‎19.4.5 and send a PKMv2 Key-Reply message to the SS with Key Sequence, SAID and HMAC Digest attributes. In the case of multicast service key request (GKEK or GTEK) the BS shall act as in PKMv2.

Note that the newer of the SS two TEKs has not expired and may still be used for both encrypting and decrypting data traffic.

#### Rekey Reauth Wait State

The wait state the TEK state machine is placed in if the TEK state machine has valid traffic keying material, has an outstanding request for the latest keying material, and the Authorization state machine initiates a reauthorization cycle.

## Key derivation and usage

The PKMv3 key hierarchy defines what keys are present in the system and how the keys are generated.

The keys used to protect management message integrity are derived from source key material generated by the authentication and authorization processes. The EAP based authentication process yields the MSK.

PKMv3 key derivation function is based on the HKDF algorithm as defined in RFC-5869.

### Pairwise Master Key (PMK)

#### PMK Derivation

The product of TLSv1.3 is the master secret. The product of the EAP-TLS exchange that is transferred to IEEE 802.16 layer is the Master Session Key (MSK), which is the TLS master secret. The key is 512 bits in length. This key is known to the AAA server (if EAP was used), to the Authenticator (BS) and to the Supplicant (SS). The SS and BS shall use the authentication product as their PMK (Pairwise Master Key):

PMK ⇐ MSK

After the successful initial authentication, the SS shall initiate reauthentication prior to expiration of PMK lifetime. After expiration of the PMK lifetime, authentication shall be performed using initial authentication procedures.

#### PMK Usage

An SS caches a PMK upon successful EAP authentication. A BS caches a PMK upon its receipt via the AAA protocol. Upon caching a new PMK for a particular SS, a BS shall delete any PMK for that SS (as well as all associated AKs).

The BS and SS will additionally delete PMKs and/or associated AKs on lifetime expiration and reauthentication, and as the result of other mechanisms such as tampering event which are beyond the scope of this specification. In the case of reauthentication, the older PMK and its AKs shall be deleted by the SS and the BS after successful completion of the new TEK keys generation.

### Authorization Key (AK)

#### AK Derivation

The BS and the SS shall share the AK which is derived from the PMK. After the EAP based authentication procedure, the MS and the Authenticator will both possess the PMK.

The AK shall be generated as follows:

AK ⇐ HKDF-Expand (PMK, SS MAC Address | BSID | “AK”, Hash.keyLength)

The Hash function used by HKDF is the ciphersuite hash algorithm and Hash.length is its output length in bytes.

#### AK Usage

The BS and SS have two active AKs during an AK transition period; the two active keys have overlapping lifetimes.

Once the AK is derived PKMv2 SA-TEK handshake begins, the BS and SS shall use the new AK matching the new PMK context for the handshake messages. Other messages shall continue to use the old AK until the handshake completes successfully. Upon successful completion of the handshake, all messages shall use the new AK.

Successful completion of the TEK derivation after authentication or reauthentication causes the activation of every AK associated with the new PMK. The BS and SS shall maintain the AK context as long as they retain the AK.

The old AK matching the old PMK context may be used for receiving packets before the “frame number” attribute specified in PKMv2 SA-TEK-response message.

If an SS fails to reauthorize before the expiration of its current AK, the BS shall hold no active AKs for the SS and shall consider the SS *unauthorized*. A BS shall remove from its keying tables all TEKs associated with an unauthorized SS’s SA.

### KEK derivation

The KEK is derived directly from the AK. The KEK is used to encrypt the GKEK that is sent by the BS to SS in multicast messages.

The keys used for KEK key material are as follows:

KEK.Key ⇐ HKDF-expand (AK, SS MAC Address | BSID | “KEK\_Key”, kek.keyLength)

KEK.IV ⇐ HKDF-expand (AK, SS MAC Address | BSID | “KEK\_IV”, kek.keyLength)

### GKEK derivation

GKEK (Group Key Encryption Key) is randomly generated at the BS or a network entity (for example, an ASA server) and transmitted to the SS encrypted with the KEK. There is one GKEK per Group Security Association. GKEK is used to encrypt the GTEKs sent by the BS to the SSs in the same multicast group or MBS group.

A BS transmits the PKMv2 Group-Key-Update-Command message (6.3.2.3.9.25) for the GKEK update mode to each SS served with the specific multicast service, the broadcast service or the MBS before the current GKEK expires and the last M&B TEK Grace Time of the corresponding current GKEK starts. The purpose of the PKMv2 Group-Key-Update-Command message for the GKEK update mode is to distribute the GKEK. The PKMv2 Group-Key-Update-Command message for the GKEK update mode is carried on the Primary Management connection. A BS intermittently transmits the PKMv2 Group-Key-Update-Command message for the GKEK update mode to each SS in order to reduce the BS’s load in refreshing traffic key material. The GKEK is needed to encrypt the new GTEK. The GKEK may be randomly generated in a BS or a network entity (i.e., an ASA server or an MBS server).

The GKEK lifetime corresponds to the *n* (integer being bigger than 1) times of the GTEK lifetime. That is, the GKEK shall be updated once while the GTEK is updated *n* times.

The multicast and broadcast rekeying overflow is shown in Figure 7-24.

### Traffic encryption key (TEK) Derivation

Both BS and MS shall derive the TEK directly from the AK. The first traffic key material derived after successful authentication or reauthentication are derived as follows:

TEK.Key ⇐ HKDF.Expand(AK, SAID|nonce|“key”, Aes.KeyLength)

TEK.Iv ⇐ HKDF.Expand(AK, SAID|nonce|“iv”, Aes.IvLength)

The next traffic keys are derived as follows:

TEK\_N.Key ⇐ HKDF.Expand(TEK\_N-1, SAID|nonce|“key”, Aes.KeyLength)

TEK\_N.Iv ⇐ HKDF.Expand(TEK\_N-1, SAID|nonce|“iv”, Aes.IvLength)

The nonce is the nonce parameter generated by the SS and sent to the BS in the KeyRequest message.

TEK\_N-1 is the current active TEK.

The multicast and broadcast rekeying management overflow is shown in Figure 7-24.

#### Group traffic encryption key (GTEK)

The GTEK is used to encrypt data packets of the multicast service and it is shared among all SSs that belong to the multicast group. There are two GTEKs per GSA. The GTEK is randomly generated at the BS. The GTEK in a PKMv2 Key-Reply message shall be encrypted by the KEK. Also, the GTEK in a PKMv2 Group Key Update Command message shall be encrypted by the GKEK.

A BS transmits the PKMv2 Group-Key-Update-Command message for the GTEK update mode carried on the broadcast connection after each M&B TEK Grace Time starts. The aim of the PKMv2 Group-Key-Update-Command message for the GTEK update mode is to distribute new GTEK and the other traffic keying material to all SSs served with the specific multicast service or the broadcast service.

An SS shall be capable of maintaining two successive sets of traffic keying material per authorized GSAID.

Through operation of its GTEK state machines, an SS shall check whether it receives new traffic keying material or not. If an SS gets new traffic keying material, then its TEK Grace Time is not operated. However, if it does not have that, then an SS shall request a new set of traffic keying material at a configurable amount of time, the TEK Grace Time, before the SS’s latest GTEK is scheduled to expire.

If an SS receives the valid two PKMv2 Group-Key-Update-Command messages and shares new valid GKEK and GTEK with a BS, then that SS does not need to request a new set of traffic keying material.

If an SS does not receive at least one of two PKMv2 Group-Key-Update-Command messages, then that SS sends the Key Request message to get a new traffic keying material. A BS responds to the PKMv2 Key Request message with the PKMv2 Key Reply message. In other words, if an SS does not get valid new GKEK or GTEK, then the GTEK request exchange procedure initiated by an SS shall be performed.

The multicast and broadcast rekeying overflow is shown in Figure 7-24.

### Derivation of message authentication codes

Message authentication code keys are used to sign management messages in order to validate the

authenticity of these messages. The message authentication code to be used is negotiated at SS Basic Capabilities negotiation. There is a different key for UL and DL messages. Also, a different message authentication key is generated for a broadcast message (this is DL direction only) and for a unicast message.

In general, the message authentication keys used to generate the HMAC Digest are derived from the AK.

The keys used for HMAC key and for KEK are as follows:

HMAC\_KEY\_U ⇐ HKDF-Expand (AK, SS MAC Address | BSID | “HMAC\_UPLINK”, HMAC.KeyLength)

HMAC\_KEY\_D ⇐ HKDF-Expand (AK, SS MAC Address | BSID | “HMAC\_DOWNLINK”, HMAC.KeyLength)

HMAC\_KEY\_GD ⇐ HKDF-Expand (GKEK, “GROUP HMAC KEY”, HMAC.KeyLength) (Used for broadcast MAC message such as a PKMv2 Group-Key-Update-Command message)

Changes in Clause 11 to support the above:

|  |  |
| --- | --- |
| Subclause | Required change |
| 11.8.4.1 PKM version support | Add option for PKMv3 |
| 11.8.4.2 Authorization policy support | Add option for TLSv1.3 based authorization |
| 11.9.29 GKEK | Change length to variable (16 or 32) |
| 11.9.6 HMAC Digest | Change length to variable (20, 32 or 48) |