Contents

Add normative reference

NIST SP 800-38A, Recommendation for Block Cipher Modes of Operation, Methods and Techniques, 2001

11 Security-oriented mechanisms

11.1 Introduction

11.2 Overview of SIEPON.4 security architecture

11.2.1 Encryption entity

An encryption entity is a distinct logical element within a communication system that is responsible for maintaining the confidentiality of data exchanged within the communication system. It achieves this by applying encryption algorithms to prevent unauthorized access and eavesdropping of sensitive information.

Each encryption entity operates independently from other encryption entities within the system and utilizes its distinct set of cryptographic parameters, including encryption keys, initialization vectors (IVs), and cryptographic configurations.

11.2.1.1 Mapping between the encryption entities and logical links

As explained in 4.5.1, all logical links of an ONU (whether provisioned or assigned during registration) are categorized as either bidirectional or unidirectional. The ONU is capable of receiving data from all provisioned logical links, while it can only transmit data through bidirectional links.

All bidirectional links terminated at a specific ONU are mapped to a single encryption entity. Correspondingly, the traffic on all bidirectional links terminated at a specific ONU is encrypted using a single ONU-wide encryption key. When a key switch event occurs, it affects all bidirectional links, although for 50G-EPON ONUs, this event may happen at different times on different channels.

The unidirectional logical links are typically provisioned as point-to-multipoint (P2MP) links and carry downstream multicast traffic. Each envelope transmitted by the OLT is delivered to multiple ONUs. Therefore, the encryption key used to encrypt the multicast traffic needs to be shared among all ONUs that are part of the multicast group. Consequently, each unidirectional LLID is mapped to a separate encryption entity.

Overall, a SIEPON.4 system that includes *N* ONUs and is provisioned to use *M* multicast LLIDs instantiates $N + M$ encryption entities.

11.2.2 Location of encryption/decryption functions

The Multi-channel Reconciliation Sublayer (MCRS) is defined in IEEE Std 802.3, Clause 143. When security mechanisms are implemented within the MCRS sublayer, such enhanced sublayer is referred to as secure MCRS (MCRS_{SEC}) sublayer. The encryption function is located in the transmit path of the MCRS_{SEC} sublayer, as illustrated in Figure 11-1(a), and the decryption function is located in the receive path of the MCRS_{SEC} sublayer, as illustrated in Figure 11-1(b).

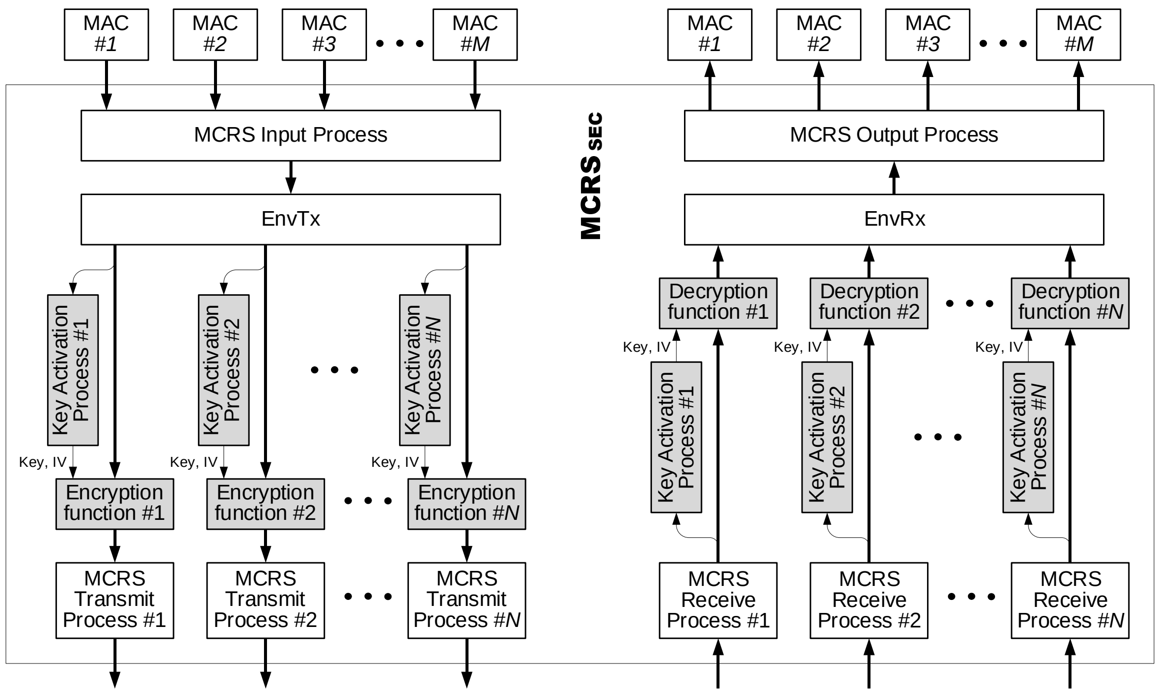

Figure 11-1 – Location of encryption/decryption function within MCRS$_{\text{SEC}}$

A separate instance of the encryption function is located within every transmit channel between the EnvTx buffer and the MCRS Transmit Process. The encryption function is driven by the Encryption Key Activation process defined in 11.6.2.

A separate instance of the decryption function is located within every receive channel between the MCRS Receive Process and the EnvRx buffer. The decryption function is driven by the Decryption Key Activation process defined in 11.6.2.

11.2.3 Latency requirements

The latency introduced into the MCRS transmit path by the encryption function (see Figure 11-1(a)) shall remain constant (to within 1 EQT), regardless of whether the encryption is enabled or disabled.

The latency introduced into the MCRS receive path by the decryption function (see Figure 11-1(b)) shall remain constant (to within 1 EQT), regardless of whether the decryption is enabled or disabled.

11.2.4 Establishment of security mechanisms

(11.2.1 in D1.4 now becomes 11.2.4)

Delete the existing 11.2.2 on D1.4 including all text

Delete the existing 11.2.3 in D1.4 (empty subclause)

11.3 ONU authentication

11.4 Initial Security Association Key exchange

11.5 Session key distribution protocol

11.5.1 Protocol overview

The session key distribution protocol is a function of the OAM client at the OLT (see 4.7.2) and the ONU (see 4.8.2).

After the initial SAK exchange (defined in 11.4), all the subsequent keys are generated by the OLT and are distributed using the *acConfigEncrKey* action (14.6.5.1). The OAMPDU carrying the next key is transmitted within the MLID envelope encrypted using the current key. The OLT shall not transmit the OAMPDU carrying the *acConfigEncrKey* action over an unencrypted MLID channel.

Figure 11-x illustrates the process of updating the session key (hereinafter referred as "key") and the subsequent activation of that key, first by the OLT and then by the ONU.

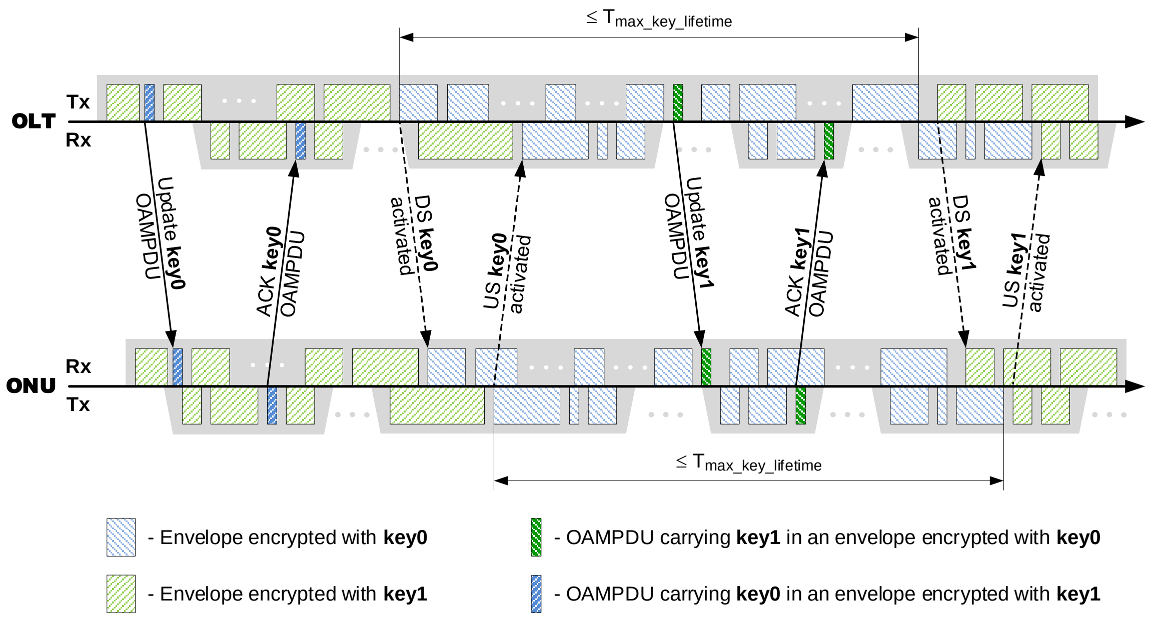
≤ $T_{max\_key\_lifetime}$
OLT
Tx
Rx
Update key0 OAMPDU
ACK key0 OAMPDU
DS key0 activated
US key0 activated
Update key1 OAMPDU
ACK key1 OAMPDU
DS key1 activated
US key1 activated
ONU
Rx
Tx
≤ $T_{max\_key\_lifetime}$
- Envelope encrypted with key0
- Envelope encrypted with key1
- OAMPDU carrying key1 in an envelope encrypted with key0
- OAMPDU carrying key0 in an envelope encrypted with key1

Figure 11-x – Key update and subsequent key activation time diagram

The next key may be distributed at any time within the lifetime of the currently-active key. In case multiple keys were distributed for the same encryption entity (i.e., under the same context object, which can be either the ONU or a multicast LLID), the last-distributed value is saved.

For each encryption entity, the OLT maintains the key lifetime timer and activates the next key upon the timer's expiry. The OLT shall distribute the next key to an ONU sufficiently in advance of the expiration time of the current key in order to allow possible retransmission attempts in case of OAMPDU delivery failure (either the OAM request or the OAM response).

The OLT may issue different key sizes to different ONUs. Different multicast LLIDs may also use different key sizes, even if these multicast LLIDs are received by the same ONU.

11.5.2 Distribution of keys for unicast LLIDs

All unicast LLIDs provisioned at a given ONU are encrypted using the same key value in both upstream and the downstream directions. Therefore, only a single OAMPDU containing the *acConfigEncrKey* action is used to distribute the next key for all unicast LLIDs at the ONU.

A unique unicast key value is distributed to each ONU via an encrypted downstream unicast MLID channel and each ONU generates an individual response OAMPDU, also transmitted using the encrypted upstream unicast MLID channel.

If the ONU's unicast key distribution acknowledgement is not received by the OLT within the OAM message timeout (see `timeoutOLT` definition in 13.3.2.3.1), the OLT shall repeat the key distribution attempt. The maximum number of key distribution attempts is an implementation design choice, but it shall be not less than 3.

ONU's failure to update the key before the expiration of the current key is a critical link condition. It causes the ONU to lose downstream connectivity and leads to OAM and MPCP timeouts and a consequent ONU deregistration.

11.5.3 Distribution of keys for multicast LLIDs

The term *multicast LLID* represents an LLID value provisioned into multiple ONUs (see 7.4.2.1). This term collectively refers to *multicast PLID*, *multicast MLID*, or the *multicast ULID*.

A key for a multicast LLID is used by all ONUs that are members of the given multicast group to decrypt the traffic associated with this LLID. ONUs use the multicast keys only for decrypting the multicast data.

A multicast key is distributed to each member of the multicast group via an encrypted unicast MLID channel. The OLT generates a separate OAMPDU carrying *acConfigEncrKey* to each member ONU and each member ONU generates an individual response OAMPDU (i.e., ACK or NACK). The sequence of the key distribution and key activation events is as shown in Figure 11-x, however the OLT distributes the multicast key to every group member before activating this key.

As is the case with the unicast key distribution, the OLT shall distribute the next multicast key to all member ONUs sufficiently in advance of the expiration time of the current key in order to allow possible retransmission attempts in case of OAMPDU delivery failures.

In case the multicast key distribution acknowledgement from any ONU in the multicast group is not received by the OLT within the OAM message timeout (see `timeoutOLT` definition in 13.3.2.3.1), the OLT shall repeat the key distribution attempt. The maximum number of multicast key distribution attempts to each ONU in the multicast group is an implementation design choice, but it shall be not less than 3. Each subsequent key distribution attempt may distribute the key to all group members or only to the ONUs that failed to acknowledge the key reception in the previous attempt(s).

Note however that the failure of some ONUs to receive or acknowledge the new multicast encryption key is not a sufficient reason for the OLT to deregister the said ONU or to delete the multicast group. The ONUs that failed to update the key will be unable to decrypt the multicast traffic subsequent to the OLT switching to the new key.

11.5.3.1 Distribution of keys for multicast MLIDs

The distribution of encryption keys for the multicast MLIDs allows for a somewhat more optimized approach. The distribution of the initial multicast key require an individual OAMPDU with *acConfigEncrKey* action to be sent to each member ONU via an encrypted unicast MLID channel, as described in 11.5.3.

However, once the encrypted multicast MLID channel is established, the subsequent multicast keys may be distributed sending a single OAMPDU carrying *acConfigEncrKey* action over this multicast channel. This method only requires a single OAMPDU to distribute the next key to the entire multicast group, no matter the group size. It must be noted however that the OLT expects an individual response OAMPDU (over a unicast MLID) from every member ONU.

11.5.3.2 Multicast distribution of multicast keys

The multicast distribution of a multicast MLID keys described in 11.5.3.1 can also be extended to multicast non-MLID channels, such as multicast PLID or multicast ULID.

This approach involves provisioning of a multicast LLID (PLID or ULID) together with a multicast MLID into each member ONU (i.e., creating an MLID multicast group that mirrors the membership of the intended PLID or ULID multicast group). The initial key for the MLID multicast group is distributed by individual OAMPDUs, as described in 11.5.3.

Once the initial key is established, the subsequent keys may be distributed by transmitting a single OAMPDU carrying *acConfigEncrKey* action over the encrypted multicast MLID channel.

This approach requires distribution of two keys each time: a key for the multicast MLID and a key for the multicast PLID/ULID. Both *acConfigEncrKey* actions carrying these keys typically can be placed into the same OAMPDU. Therefore, this method only requires a single OAMPDU to distribute the next MLID key and PLID/ULID key to the entire multicast group, no matter the group size.

As mentioned above, the OLT expects an individual acknowledgement message (over a unicast MLID) from every member ONU. The acknowledgement of the multicast MLID key and the acknowledgement of the multicast PLID/ULID key may be packed into the same OAMPDU, requiring only a single OAM response message transmitted upstream by each ONU.

The benefits of multicast distribution of keys for multicast non-MLID flows are mostly realized with long-lived multicast groups (i.e., groups with expected lifetimes much longer that the lifetime of a single session key).

11.6 Session key activation protocol

11.6.1 Protocol overview

The key activation protocol defines a procedure of switching from the current encryption key to a new encryption key that has been previously distributed by the OLT to one or more ONUs using the session key distribution protocol (see 11.5).

The key activation protocol relies on encryption signaling fields embedded in envelope headers. These fields include the encryption enabled flag (*EncEnabled* field) and encryption key index (*EncKey* field). The *EncEnabled* and *EncKey* fields are described in IEEE Std 802.3, 143.3.2 and 143.3.3.4. The *EncKey* field takes on values of only 0 and 1.

Figure 11-xx illustrates the procedure of a key activation, which consists of four sequential steps.

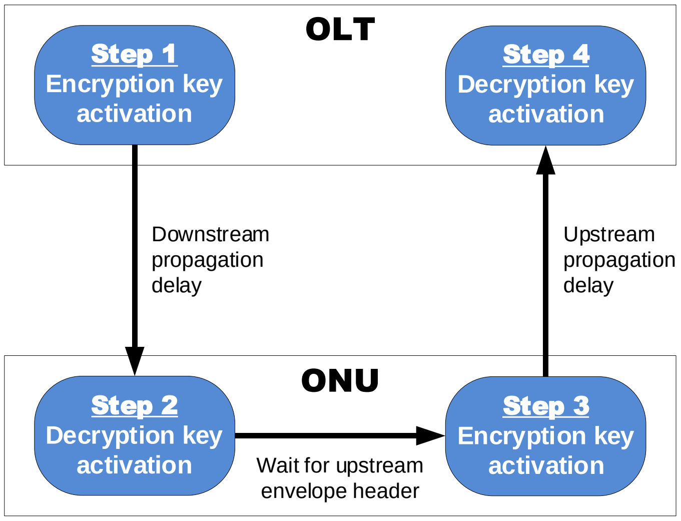
OLT
Step 1
Encryption key activation
Step 4
Decryption key activation
Downstream propagation delay
Upstream propagation delay
ONU
Step 2
Decryption key activation
Step 3
Encryption key activation
Wait for upstream envelope header

Figure 11-xx – Four steps comprising the key activation procedure

Each of the above fours steps is represented by an independent process that runs continuously within the secure Multi-Channel Reconciliation Sublayer (MCRS $_{SEC}$).

11.6.1.1 Step 1: Encryption key activation at the OLT

The process of encryption key activation at the OLT is defined in 11.6.2.4. The OLT activates the new encryption key upon the expiration of the key activation timer.

Generally, every encryption entity (i.e., ONUs and multicast LLIDs) maintains its own key activation timer and these timers may have different intervals and/or be set to expire at different times. However, for practical considerations, it is allowed for all encryption entities to share the same key activation timer.

Once the key activation timer expired, the OLT waits for the next envelope header destined to the given encryption entity. The OLT indicates switching to the new key by toggling the value of the *EncKey* field in the envelope header. The envelope payload following this envelope header is encrypted using the new key.

11.6.1.2 Step 2: Decryption key activation at the ONU

The process of decryption key activation at the ONU is defined in 11.6.2.5. For every received envelope, the ONU retrieves a key associated with the given encryption entity, identified by the LLID field in the envelope header, and the key index, identified by the *EncKey* field. Thus, toggling of the *EncKey* value by the OLT in Step 1 above caused the ONU to also retrieve the new key after it parsed and processed this envelope header.

11.6.1.3 Step 3: Encryption key activation at the ONU

The process of encryption key activation at the ONU is defined in 11.6.2.6. ONU's activation of a new decryption key in Step 2 also serves as a trigger for activating the same key for the encryption of its upstream transmission. The ONU waits for the next envelope header from to the given encryption entity. The ONU indicates switching to the new key by toggling the value of the *EncKey* field in the envelope header. The payload following this envelope header is encrypted using the new key.

11.6.1.4 Step 4: Decryption key activation at the OLT

The process of the decryption key activation at the OLT is identical to that process at the ONU, and is, in fact, described by the same state diagram (see 11.6.2.5). For every received envelope, the OLT retrieves a key associated with the given encryption entity, identified by the LLID field, and the key index, identified by the *EncKey* field. Thus, toggling of the *EncKey* value by the ONU in Step 3 above caused the OLT to also retrieve the new key after it parsed and processed this envelope header.

Optionally, the OLT may implement additional safety check of comparing that the retrieved decryption key matches the previously used encryption key. If implemented, such check shall be performed not earlier than a round-trip time after the activation of the new encryption key in step 1.

11.6.1.5 Multicast key activation

The activation of the multicast key, i.e., the key associated with a multicast LLIDs, involves only step 1 (11.6.1.1) and step 2 (11.6.1.2) because the multicast LLIDs carry traffic only in the downstream direction.

The activation of the encryption key by the OLT, as signaled by toggling of the *EncKey* field in the downstream envelope header is detected by all ONUs that are members of the given multicast group. This causes all member ONUs to activate the new key for the decryption.

11.6.1.6 Location of key activation processes

The encryption and decryption key activation processes are located within the secure MCRS (MCRS $_{\text{SEC}}$) sublayer, as detailed in 11.2.2.

11.6.2 Definition of processes comprising the key activation protocol

11.6.2.1 Variables

`activeKeyIndex[ee]`

TYPE: 1-bit integer

This variable represents the index of the currently active encryption/decryption key for the encryption entity `ee`. Incrementing this variable by 1 causes its value to toggle between 0 and 1.

`decryptionCounter[ch]`

TYPE: 128-bit sequence

The initial value of the counter (initialization vector) used as an input block to the AES forward cipher in the AES-CTR mode for decryption (see NIST SP 800-38A, 6.5). The `decryptionCounter` is calculated independently for every received envelope header on every channel `ch` and is passed to the decryption function (see Figure 11-1(b)).

`decryptionKey[ch]`

TYPE: sequence of 128 or 256 bits

The value of the key currently used for decryption on channel `ch`. The `decryptionKey` value is fetched for every received envelope header and is passed to the decryption function (see Figure 11-1(b)).

`encryptionCounter[ch]`

TYPE: 128-bit sequence

The initial value of the counter (initialization vector) used as an input block to the AES forward cipher in the AES-CTR mode for encryption (see NIST SP 800-38A, 6.5). The `encryptionCounter` is calculated independently for every transmitted envelope header on every channel `ch` and is passed to the encryption function (see Figure 11-1(a)).

`encryptionEnabled[ee]`

TYPE: boolean

This variable indicates whether the encryption is enabled or disabled for the given encryption entity `ee`.

It the OLT, this variable is set to `true` when the initial encryption key becomes available (calculated or provisioned) to the encryption entity `ee` in both the OLT and an ONU (refer to the definition of variable `initialKeyReady[ee]`). Note that for testing and troubleshooting purposes, the NMS may temporarily disable the encryption for an encryption entity `ee` by overwriting the `encryptionEnabled[ee]` with the value of `false` (see 11.9).

In the ONU, under normal operation, this variable is equal to `receivedEncrypted[ee]`, i.e., an encryption entity `ee` encrypts the outgoing envelopes only if the envelopes this entity receives from the OLT are also encrypted. For testing and troubleshooting purposes, the NMS may force the ONU's encryption to be on or off (via the attribute *aEncryptionMode*, 14.4.5.2)

`encryptionKey[ch]`

TYPE: sequence of 128 or 256 bits

The value of the key currently used for encryption on channel `ch`. The `encryptionKey` value is fetched for every transmitted envelope header and is passed to the encryption function (see Figure 11-1(a)).

`encryptionMode[ee]`

TYPE: boolean

This variable is an alias of the extended attribute *aEncryptionMode* (0xDB/0x04-02) defined in 14.4.5.2.

`initialKeyDone[ee]`

TYPE: boolean

This variable is used only by the OLT encryption key activation process (11.6.2.4), where it causes the replacement of the initial (ephemeral) key by the session key as soon as the first session key is distributed to the ONU (i.e., without waiting for the key lifetime interval to expire).

If the encryption entity `ee` is associated with an ONU object, this variable is set to `true` by the OLT OAM client after the first session key was distributed to the ONU and its reception and processing was acknowledged by the ONU (see step 4 in 11.2.3). This variable is reset to `false` on read.

If the encryption entity `ee` is associated with a multicast LLID object, this variable is equal to `false` at all times.

`initialKeyReady[ee]`

TYPE: boolean

This variable is used only by the OLT encryption key activation process (11.6.2.4), where it causes the encryption entity `ee` to start encrypting traffic in the downstream direction.

If the encryption entity `ee` is associated with an ONU object, this variable is set to `true` by the OLT OAM client after the initial (ephemeral) key was calculated and stored in the array `keys` as element `keys[ee][0]`. The derivation of the initial key is described in 11.4. This variable is reset to `false` on read.

In the encryption entity `ee` is associated with a multicast LLID, this variable is set to `true` by the OLT OAM client after the first session key was distributed to all ONUs in a multicast group and its reception and processing was acknowledged by every ONU. This variable is reset to `false` on read.

`keys[E][2]`

TYPE: array of encryption keys

`keys[E][2]` is a two-dimensional array representing the stored encryption keys. The array size is $E \times 2$, where E represents the total number encryption entities, with two values stored for each entity - the currently-active key and the key to be activated next. The value of E for the OLT (E_{OLT}) is defined in 11.8.1 and its value for the ONU (E_{ONU}) is defined in 11.8.2.

The key values are written into the `keys[E][2]` array by the Security function of the OAM Client as the end result of the Session Key Distribution Protocol (see 11.5). The Session Key Activation protocol stat diagrams have a read-only access to this array.

`keyInterval[ee]`

TYPE: integer

This variable represents an interval of time between updating the encryption keys (i.e., a key lifetime) for encryption entity `ee`. The value of this variable is provisioned to the OLT by the NMS, subject to constraints listed in 11.8.3.

`receivedEncrypted[ee]`

TYPE: boolean

This variable indicates whether the received envelope, whose LLID value maps to the encryption entity `ee`, is encrypted or not. In the OLT and in the ONU, the value of this variable is derived from the *EncEnabled* field of the received envelope headers.

`RxEQ[ch]`

TYPE: EQ

This variable represents an envelope quantum (EQ) received and stored in the `EnvRx` buffer of the MCRS on channel `ch` (see IEEE Std 802.3, 143.3.4).

`TxEQ[ch]`

TYPE: EQ

This variable represents an envelope quantum (EQ) being transmitted from the `EnvTx` buffer of the MCRS on channel `ch` (see IEEE Std 802.3, 143.3.3).

11.6.2.2 Functions

`calculateIV(ch, eq)`

This function calculates the value of the initialization vector (IV) used as an input block to the AES forward cipher in the AES-CTR (see NIST SP 800-38A, 6.5). The argument `ch` is an index of the channel on which the IV is to be used. The argument `eq` is an EQ that represents an envelope header. The IV construction method is defined in TBD.

`isHeader(eq)`

The `IsHeader(eq)` function returns true if the parameter `eq` represents an envelope header. This function is defined in IEEE Std 802.3, 143.3.4.4.

`mapEncrEntity(llid)`

The `mapEncrEntity(llid)` function maps an LLID value to an index of an encryption entity (see 11.2.1.1). Each multicast LLID maps to an encryption entity associated with that multicast LLID. All unicast (bidirectional) LLIDs provisioned on an ONU map to a single encryption entity associated with the given ONU.

11.6.2.3 Timers

`keyTimer[ee]`

This timer is used to count down the time remaining until the next key update. There exists a separate instance of this counter for every encryption entity `ee`. A key for encryption entity `ee` may not be used past the expiration of the `keyTimer[ee]`.

Each timer instance `keyTimer[ee]` is associated with an instance of boolean variable `keyTimer_done[ee]`. Upon expiration of the timer, the value of `keyTimer_done[ee]` becomes true (see 3.3.6).

11.6.2.4 OLT encryption key activation process state diagram

The OLT shall implement the encryption key activation process as depicted in state diagram in Figure 11-xxx. There shall be a separate instance of the encryption key activation process for each transmit channel `ch` in the OLT.

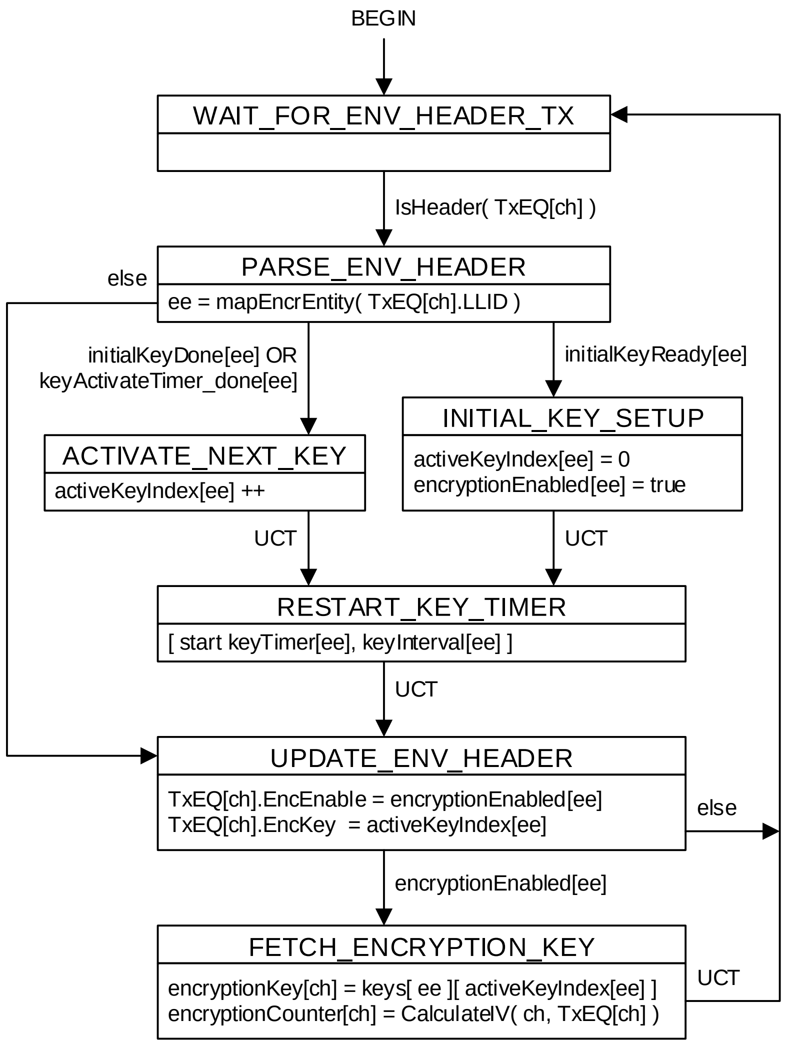
BEGIN
WAIT_FOR_ENV_HEADER_TX
IsHeader(TxEQ[ch])
PARSE_ENV_HEADER
ee = mapEncrEntity(TxEQ[ch].LLID)
else
initialKeyDone[ee] OR keyActivateTimer_done[ee]
initialKeyReady[ee]
ACTIVATE_NEXT_KEY
activeKeyIndex[ee] ++
INITIAL_KEY_SETUP
activeKeyIndex[ee] = 0
encryptionEnabled[ee] = true
UCT
UCT
RESTART_KEY_TIMER
[start keyTimer[ee], keyInterval[ee]]
UCT
UPDATE_ENV_HEADER
TxEQ[ch].EncEnable = encryptionEnabled[ee]
TxEQ[ch].EncKey = activeKeyIndex[ee]
else
encryptionEnabled[ee]
FETCH_ENCRYPTION_KEY
encryptionKey[ch] = keys[ee][activeKeyIndex[ee]]
encryptionCounter[ch] = CalculateIV(ch, TxEQ[ch])
UCT

Figure 11-xxx -- OLT encryption key activation process state diagram

11.6.2.5 OLT and ONU decryption key activation process state diagram

The OLT and the ONU shall implement the decryption key activation process as depicted in state diagram in Figure 11-xxxx. There shall be a separate instance of the decryption key activation process for each receive channel `ch` in the OLT and in the ONU.

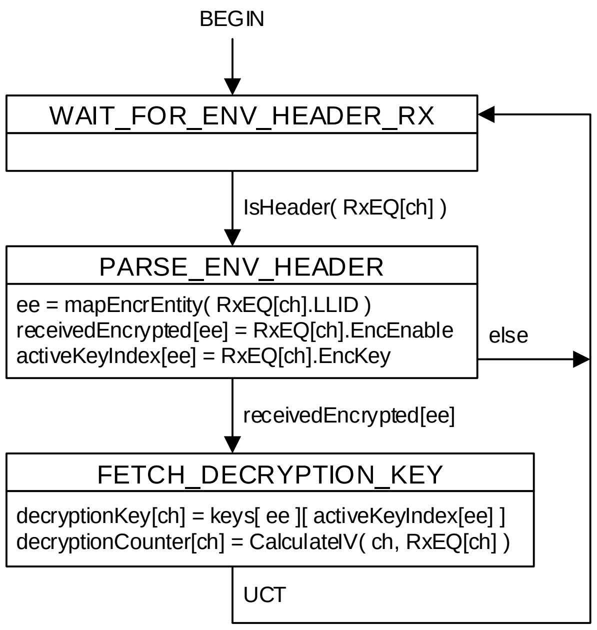
BEGIN
WAIT_FOR_ENV_HEADER_RX
IsHeader(RxEQ[ch])
PARSE_ENV_HEADER
ee = mapEncrEntity(RxEQ[ch].LLID)
receivedEncrypted[ee] = RxEQ[ch].EncEnable
activeKeyIndex[ee] = RxEQ[ch].EncKey
else
receivedEncrypted[ee]
FETCH_DECRYPTION_KEY
decryptionKey[ch] = keys[ee][activeKeyIndex[ee]]
decryptionCounter[ch] = CalculateIV(ch, RxEQ[ch])
UCT

Figure 11-xxxx -- OLT and ONU decryption key activation process state diagram

11.6.2.6 ONU encryption key activation process state diagram

The ONU shall implement the encryption key activation process as depicted in state diagram in Figure 11-xxxxx. There shall be a separate instance of the encryption key activation process for each transmit channel in the ONU.

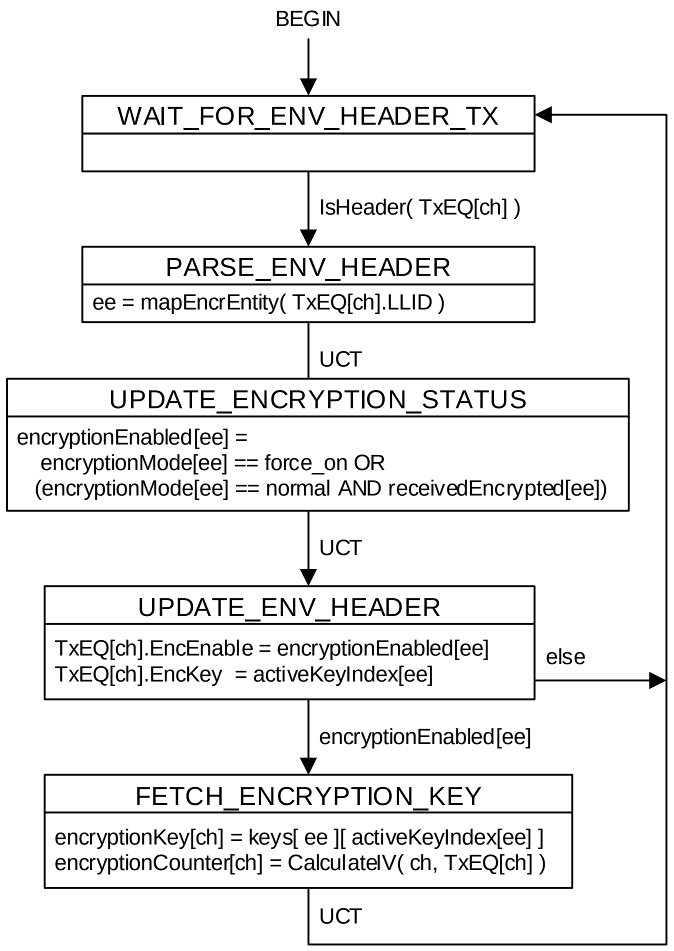
BEGIN
WAIT_FOR_ENV_HEADER_TX
IsHeader(TxEQ[ch])
PARSE_ENV_HEADER
ee = mapEncrEntity(TxEQ[ch].LLID)
UCT
UPDATE_ENCRYPTION_STATUS
encryptionEnabled[ee] =
encryptionMode[ee] == force_on OR
(encryptionMode[ee] == normal AND receivedEncrypted[ee])
UCT
UPDATE_ENV_HEADER
TxEQ[ch].EncEnable = encryptionEnabled[ee]
TxEQ[ch].EncKey = activeKeyIndex[ee]
else
encryptionEnabled[ee]
FETCH_ENCRYPTION_KEY
encryptionKey[ch] = keys[ee][activeKeyIndex[ee]]
encryptionCounter[ch] = CalculateIV(ch, TxEQ[ch])
UCT

Figure 11-xxxxx -- ONU encryption key activation process state diagram

11.7 Data encryption/decryption mechanisms

11.8 Encryption key management

11.8.1 Key storage in the OLT

The OLT encrypts the unicast traffic to each ONUs using a unique key. Every multicast LLID is encrypted using a unique key as well.

Given a PON configuration that includes N ONUs and M multicast LLIDs, the OLT shall be able to support E_{OLT} encryption entities, where $E_{OLT} = N + M$ (refer to use of constant E_{OLT} in the definition of `keys[E][2]` in 11.6.2.1). The OLT shall contain enough storage space for $2\times E_{OLT}$ keys, with each key being 128 or 256 bits long.

At any time, in the OLT, at most N keys are active for decryption and $N + M$ keys are active for encryption.

11.8.2 Key storage in the ONU

The ONU encrypts all unicast LLIDs with the same key in both upstream and the downstream directions. Each multicast LLID provisioned into the given ONU is encrypted using its independent key.

For each encryption key, the ONU stores two key values: the currently-active key value and the key value that will become active on the next key switch event.

Given an ONU configuration that includes any number of unicast LLIDs and *m* multicast LLIDs, the ONU shall be able to support E_{ONU} encryption entities, where $E_{ONU} = m + 1$ (refer to use of constant E_{ONU} in the definition of `keys[E][2]` in 11.6.2.1). The ONU shall contain enough storage space for $2 \times E_{ONU}$ keys, with each key being 128 or 256 bits long.

At any time, in the ONU, at most m+1 keys are active for decryption and only one key is active for encryption.

11.8.3 Key lifetime

One of the requirements of AES CTR mode is that the counter values do not repeat for the duration (lifetime) of a single encryption key (see NIST SP 800-38A, 6.5). Therefore the construction method of the Initialization Vector (IV) imposes the upper limit on the encryption key lifetime.

The IV construction is defined in TBD. It relies on the extended 48-bit MPCP clock counter. This counter increments every envelope quantum time (EQT), which equals 2.56 ns (see IEEE Std 802.3, 1.4.245c). Thus, the MPCP counter rolls-over every 200.16 hours. The OLT shall maintain the key lifetime of less than or equal to 200 hours ($T_{max\_key\_lifetime}$). Different encryption entities may optionally have different key lifetimes not exceeding the $T_{max\_key\_lifetime}$.

11.9 Encryption testing and troubleshooting modes of operation

Under the normal operation conditions, for all encryption entities associated with ONU objects, the encryption is always enabled in both downstream and upstream directions. For all encryption entities associated with multicast LLID objects, the encryption is always enabled the downstream direction.

For testing and troubleshooting purposes, the NMS may temporarily disable the encryption for any encryption entity `ee` in the downstream direction and/or in the upstream direction, if applicable.

11.9.1 Encryption entities associated with multicast LLID objects

For an encryption entity `ee` associated with a multicast LLID object, the encryption is disabled by setting the variable `encryptionEnabled[ee]` at the OLT to `false` (see 11.6.2.1). Disabling encryption for a multicast ULID or a multicast PLID does not affect the security of traffic or the key updates for other encryption entities. There are special considerations for disabling encryption for an entity `ee` associated with a multicast MLID, as explained in 11.9.3.

Note that disabling the encryption of multicast ULID/PLID does not stop the periodic key updates for these multicast LLIDs. The key updates continue over the unicast MLID or multicast MLID channels, which remain encrypted and secure.

Upon completion of the testing or troubleshooting analysis, the encryption of multicast ULID or a multicast PLID traffic is re-enabled by simply setting the variable `encryptionEnabled[ee]` at the OLT to `true`.

11.9.2 Encryption entities associated with ONU objects

An encryption entity `ee` associated with an ONU object carries all the unicast traffic to and from that ONU (i.e., it aggregates all the unicast LLIDs at a given ONU, including the unicast MLID).

Setting the variable `encryptionEnabled[ee]` to `false` at the OLT disables the encryption of downstream traffic associated with encryption entity `ee`. This, in turn, causes the ONU encryption key activation process to disable the encryption of the upstream unicast traffic associated with this encryption entity `ee` (see 11.6.2.6).

NOTE -- The action of disabling the downstream encryption for an encryption entity `ee` associated with an ONU object may expose the encryption keys distributed via the MLID channel. Such action shall never be performed on ONUs carrying live user traffic.

Once this value `encryptionEnabled[ee]` is set to false at the OLT, it cannot be changed back to true via NMS anymore. To re-enable the encryption, the OLT shall initiate the ONU authentication process (see 11.3) and the initial key exchange (see 11.4).

At the ONU, the upstream encryption is controlled via the attribute *aEncryptionMode* (see 14.4.5.2). Under the normal operation, the upstream encryption is enabled if the downstream unicast traffic to this ONU is encrypted. For testing and troubleshooting purposes, the *aEncryptionMode* allows forcing the upstream encryption on or off, regardless of the status of the downstream encryption.

Temporarily disabling the upstream encryption, while still using the downstream encryption, is a less invasive operation, since no encryption keys are ever transmitted in the upstream direction. Upon conclusion of the troubleshooting analysis, the upstream encryption can be re-enabled by setting the *aEncryptionMode* attribute to `normal`.

11.9.3 Encryption entities associated with multicast MLID objects

Special care is required when disabling the encryption for multicast MLIDs, as such MLIDs may be used to distribute the multicast encryption keys (see 11.5.3.1 and 11.5.3.2). The multicast MLID encryption shall never be disabled in systems carrying live user multicast traffic.