

## 11 Security-oriented mechanisms

~~Clause 11 introduces the security-related mechanisms, focusing in particular on ONU identity and various aspects of ONU authentication access management (see 11.2) and data encryption / decryption (see 11.3), achieved in an interoperable manner.~~

### 11.1 ~~Introduction~~ **Overview of threats and mitigation measures**

~~Clause 11 introduces the security related mechanisms, focusing in particular on various aspects of ONU authentication (see 11.3) and data encryption / decryption (see 11.7), achieved in an interoperable manner.~~

#### 11.1.1 Threat model

The security threat model in PONs encompasses various risks and vulnerabilities that need to be addressed to ensure a secure and reliable network infrastructure:

— **Broadcast of downstream data:**

One of the inherent vulnerabilities in PONs is the broadcast nature of downstream data transmission. As downstream data is broadcasted to all ONUs attached to the OLT's PON port, a malicious user who gains control of an ONU or attaches an un-authorized device could intercept and access downstream data intended for all connected users. This poses a significant risk to the confidentiality and privacy of user data.

— **Impersonation and service theft:**

Due to the nature of upstream data transmission in PONs, where data can originate from any ONU attached to the ODN, a malicious user with control over an ONU could forge packets to impersonate a different ONU. This "theft of service" attack allows the attacker to masquerade as a legitimate ONU, potentially gaining unauthorized access to network resources or services.

— **Infrastructure Tampering:**

An attacker could compromise the security of the PON infrastructure by physically tampering with street cabinets, spare ports, or fiber cables. By connecting a malicious device at various points within the network, the attacker could intercept and manipulate network traffic. Depending on the location of the malicious device, it could impersonate the OLT, allowing unauthorized access and control over the network, or impersonate an ONU to intercept and tamper with data.

— **Packet Replay and Bit-Flipping Attacks:**

In PONs, a malicious user who successfully captures network packets transmitted on the PON could store and replay them at a later time. This replay attack poses a threat to data integrity and can lead to unauthorized access or service disruption. Additionally, an attacker could conduct bit-flipping attacks, altering the content of transmitted packets, potentially compromising the integrity and accuracy of the data.

— **Data Capture and Offline Decrypt:**

In PONs, a malicious user who successfully captures network packets transmitted on the PON could store and attempt to decrypt them at a later time if/when the private key(s) of the peers is cracked or compromised – gaining access to any encryption keys derived from the public key exchange. This attack could compromise tremendous amounts of data retroactively – even if/when compromised public/private key pairs are taken out of service.

1 **11.1.2 Physical protection of security data in the ONU**

2 The physical protection measures outlined in this sub-clause collectively ensure the secure storage and  
3 protection of certificates and keys within ONUs in PON networks, safeguarding against unauthorized  
4 access and potential security breaches.

5 **11.1.2.1 Secure non-volatile storage**

6 ONUs shall implement the requirements outlined in SECv3.0 for secure non-volatile storage of certificates  
7 and their associated public/private keys. This ensures that sensitive cryptographic information remains  
8 protected from unauthorized disclosure and modification.

9 **11.1.2.2 Protection against unauthorized access**

10 ONUs should employ measures to prevent unauthorized access to the ONU certificate private key,  
11 particularly in production devices. One recommended approach is to restrict or block physical access to the  
12 memory that contains the private key. This prevents unauthorized individuals from using debugger tools to  
13 read the key.

14 **11.1.2.3 Use of One-Time-Programmable (OTP) Memory**

15 To enhance security and make it more challenging to replicate a valid ONU, ONUs should use one-time-  
16 programmable (OTP) memory for storing certificates and keys. OTP memory is designed to prevent  
17 alteration or cloning of the stored data, thereby bolstering the overall security of the ONU.

18 **11.1.2.4 Compliance with FIPS-140-2 Security Requirements**

19 The ONU shall meet the security requirements specified in the Federal Information Processing Standard  
20 (FIPS) 140-2 for all instances of permanent private and public key storage. FIPS-140-2 provides a rigorous  
21 set of cryptographic security standards, ensuring the confidentiality and integrity of the stored keys.

22 **11.1.2.5 Compliance with FIPS-140-2 Security Level 1**

23 The ONU shall also comply with FIPS-140-2 Security Level 1. This level requires minimal physical  
24 protection and shall be achieved through the use of production-grade enclosures. The enclosures for ONUs  
25 shall meet production-grade quality standards, including standard passivation sealing. The circuitry within  
26 the ONU shall be implemented as a production-grade multi-chip embodiment, typically in the form of an  
27 IC printed circuit board. The ONU itself should be contained within a metal or hard plastic enclosure,  
28 which may include doors or removable covers.

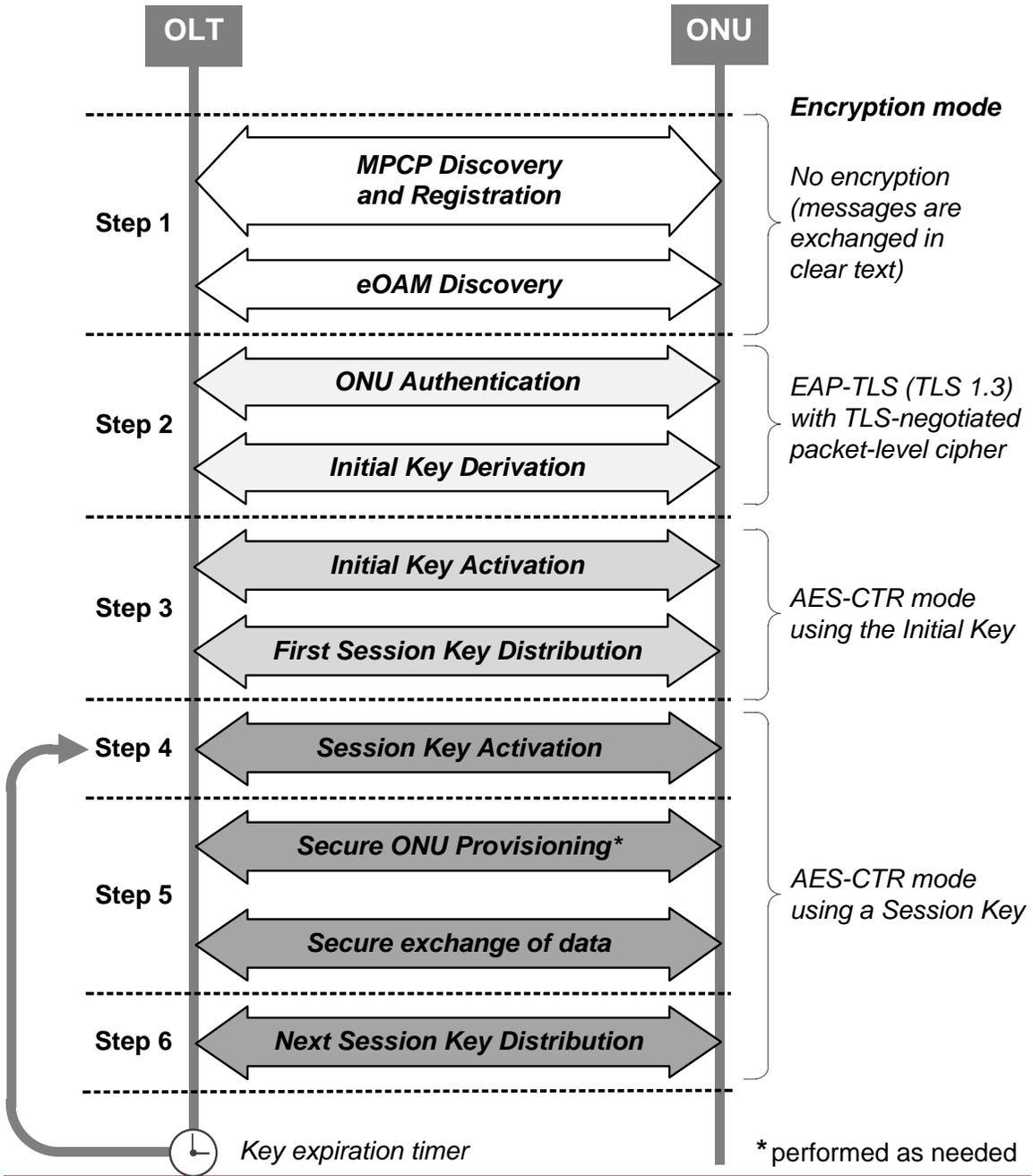
29 **11.1.3 Security mechanisms**

30 The physical security and tamper-proof installation of the ODN, optical splitters, and ONUs can enhance  
31 the overall security of the PON infrastructure. By implementing physical security measures, such as those  
32 listed in 11.1.2 ~~secure enclosures, access controls, and tamper-evident mechanisms~~, the risk of unauthorized  
33 access or tampering can be significantly reduced. ~~The recommended physical security measures are~~  
34 outlined in 11.10.

35 However, the specific threat landscape of PON installations requires deployment of specific security  
36 measures, such as robust authentication protocols, encryption mechanisms, network monitoring systems,  
37 and intrusion detection systems, to mitigate the identified threats effectively. By adopting a comprehensive  
38 and multi-layered security approach, PON operators can ensure the confidentiality, integrity, and  
39 authenticity of data transmitted over the network, even in scenarios where physical security measures are  
40 compromised or absent.

1 **11.1.2-11.1.3.1 Establishment of security mechanisms**

- 2 The process of adding a new ONU to a PON follows a series of defined steps, ensuring secure ONU  
3 integration and subsequent operation. Figure 11-4 illustrates these steps:



1

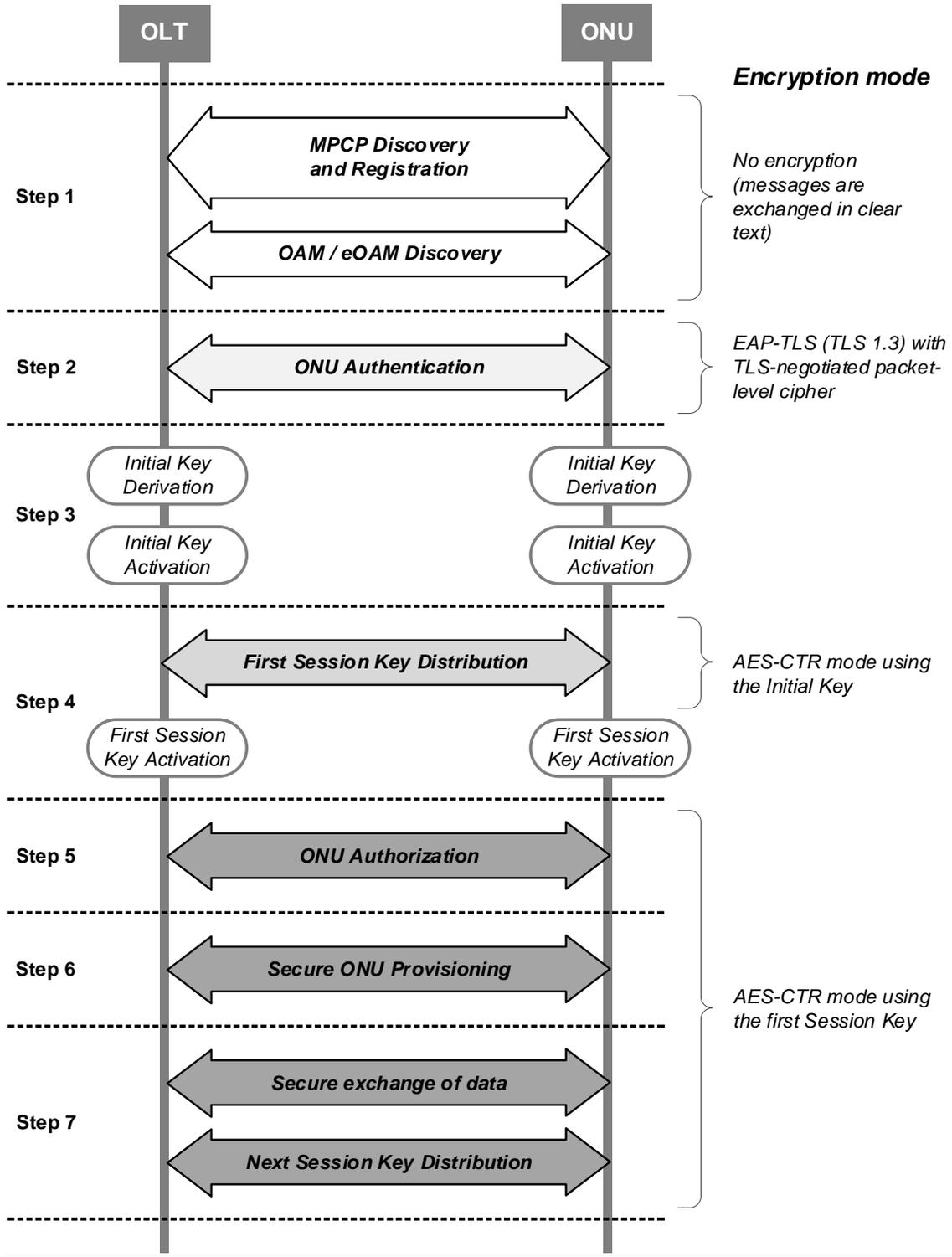


Figure 11-4–Sequence of steps to establish secure ONU operation

**Step 1 – ONU Discovery and Registration:**

Upon completion of the boot/restart sequence, the ONU completes the MPCP, OAM, and eOAM discovery processes as specified in 13.3.1 and 13.3.2. At this time, ONU gets is assigned two

1 logical links: PLID for exchanging the GATE and REPORT MPCPDUs and MLID for  
2 exchanging the OAM control messages (OAMPDUs). The MPCP and OAM discoveries are  
3 performed in the clear, i.e., using unencrypted MPCP and OAM messages.

4 **Step 2 – Establishment of the initial key:**

5 ~~Once the OLT and the ONU have established an MLID, they exchange keying material used for mutual~~  
6 ~~derivation of an initial symmetric cryptographic key that enables private communication between the ONU~~  
7 ~~and OLT and perfect forward secrecy. This is described in detail in 11.3.~~

8 **Step 3 – ONU Authentication authentication:**

9 Upon ~~establishment of the initial key~~ completion of eOAM discovery, the ONU and OLT proceed  
10 with ONU authentication using the method defined in 11.2.4. This step cryptographically  
11 verifiesensures that the ONU is a trusted entityidentity within the network ~~and that the initial key~~  
12 ~~is authentic. If the ONU fails to authenticate, the OLT may retry the authentication procedure with~~  
13 ~~a different ONU's credential. The OLT does not provision user services to an ONU that has failed~~  
14 ~~to authenticate using any of its available certificates.~~

15 **Step 3 – Establishment of the initial key:**

16 Once the ONU and OLT have completed the authentication exchange, both the OLT and ONU  
17 independently derive the initial symmetric encryption key. The key derivation and activation  
18 methods are specified in 11.3.2.

19 **Step 4 – Secure distribution and activation of session keys:**

20 The OLT/NMS distributes the ~~new~~ first session key to the ONU using the session key distribution  
21 protocol (see 11.3.3). The messages exchanged under the session key distribution protocol are  
22 encrypted using the initial key obtained in Step 23. After the encryption key is distributed to the  
23 ONU, the OLT initiates a switch to the new key (i.e., key activation by both the OLT and the  
24 ONU), using the procedure described in 11.3.4.

25 **Step 5 – ONU authorization**

26 ONU authorization refers to the verification of ONU's eligibility to operate on the operator's  
27 network. ONU authorization is initiated by the NMS/AAA server after the ONU has successfully  
28 authenticated and symmetric encryption has been established for secure communication between  
29 the OLT and the ONU. The ONU authorization procedure is specified in 11.2.3. The exchange of  
30 information between the OLT and ONU pertaining to the ONU authorization is carried over the  
31 encrypted MLID logical link.

32 **Step 5-6 – Secure provisioning of additional logical links:**

33 ~~With the encryption of the PLID and MLID of the given ONU established~~ If the NMS determines  
34 that the ONU is authorized to operate on a given PON network, it proceeds, the NMS can proceed  
35 with the provisioning of the additional bidirectional and unidirectional (multicast) logical links for  
36 this ONU. The additional logical links are provisioned using the extended action acConfigLlid  
37 (see 14.6.2.8).

38 Note that no additional encryption configuration steps are needed for the newly-provisioned  
39 bidirectional links. These links automatically begin operating with the ONU's currently active  
40 session encryption key. However, this is not the case for the newly provisioned unidirectional  
41 (multicast) logical links. As explained in 11.3.3.3, each of multicast logical link uses a unique  
42 encryption key that is shared among all members of this multicast group. Therefore, before the

1 ONU is able to process the data frames received on the multicast logical link, it needs to obtain the  
2 specific encryption key for that multicast group. The multicast keys are distributed using the same  
3 extended action *acConfigEncrKey* (see 14.6.5.1) as is used for the unicast keys.

4 The control messages that provision additional logical links (*acConfigLlid* actions) and the  
5 messages that convey encryption keys for the multicast LLIDs (*acConfigEncrKey* action) are  
6 securely exchanged between the OLT and an ONU via the encrypted MLID link.

#### 7 **Step 6-7 – Exchange of encrypted data and control messages:**

8 With the session keys distributed by the OLT and additional ULIDs provisioned for carrying the  
9 subscriber data, the OLT and the ONU can securely exchange data frames over the PON. The data  
10 frames are encrypted using the session key to ensure confidentiality and integrity during  
11 transmission. The cryptographic method is defined in 11.3.5. The OLT can also provide  
12 credential and trust store updates to the ONU for future authentication as described in **TBD**.

### 13 **11.2 Authentication of the ONU identity and access management**

14 Before an ONU is allowed access to the operator’s network and symmetric encryption keys established for  
15 secure communication between the OLT and the ONU, the ONU’s identity needs to be determined and  
16 verified. The authenticated ONU identity can then be used by the NMS to reliably establish what level of  
17 access the ONU is granted. This subclause defines how this authentication is performed.

#### 18 **11.2.1 ONU identity**

19 The SIEPON.4 system uses the ONU MAC address associated with the ONU’s PON port as the identity of  
20 the ONU (see 14.4.1.2). ~~However, t~~The ONU identity cannot be trusted until the ONU has successfully  
21 completed the Authentication procedure (see 11.3.4).

22 When an ONU is powered on, it reports its MAC address to the OLT through the MPCP Discovery process,  
23 as defined in IEEE Std 802.3, Clause 144.

24 The OLT shall use the ONU MAC address associated with PON port as the identity of the ONU.

#### 25 **11.2.2 Network Access Control/ONU authentication**

26 The OLT shall initiate the ONU ~~A~~authentication procedure immediately after the completion of  
27 OAM/eOAM discovery (see Figure 11.4). At this time, the ONU’s access to the network is restricted; it has  
28 only two “system” logical links assigned to it: PLID used for providing connectivity (carries GATE and  
29 REPORT MPCPDUs) and MLID used for ONU management (carries OAMPDUs). The ONU cannot  
30 support any data services at this time.

31 ONU authentication in SIEPON.4 is accomplished using EAP-TLS 1.3 (see RFC-9190) with the OLT  
32 operating as the “EAP Authenticator”/“EAP-TLS Server” and the ONU operating as the “EAP  
33 Supplicant”/“EAP-TLS Client” (see RFC-3748). By utilizing TLS 1.3 as described in this clause, the TLS  
34 handshake is encrypted and authenticated using an ephemeral key that provides a shared secret to both  
35 parties while ensuring privacy and perfect forward secrecy (PFS). The shared secret is then used by both  
36 the OLT and ONU to independently derive the SIEPON.4 initial traffic encryption key, as described in  
37 **11.3.2**.

38

39 ~~Before provisioning data services to the ONU, the OLT shall authenticate the ONU identity and verify that~~  
40 ~~the ONU is authorized to operate on the given network. If port based access control is enabled, the OLT~~  
41 ~~shall verify that given ONU is authorized for the particular OLT PON port on which it is discovered.~~

1 The OLT may consult NMS/AAA servers ~~to determine for assistance with~~ the ONU's ~~authorization~~  
2 ~~authentication. to access the network resources.~~ The mechanism of communication with the NMS/AAA  
3 servers or the nature of information passed between the OLT and the servers is outside the scope of this  
4 standard.

5 The OLT shall ~~restrict access to the network deny service provisioning~~ to the following categories of  
6 ONUs:

- 7 — An ONU~~s~~ that failed authentication~~;~~;
- 8 — An ONU whose identity matches that of another authenticated ONU~~;~~;
- 9 — An ~~authenticated~~ ONU that ~~has been administratively prohibited from connecting to the given~~  
10 ~~network (i.e., an ONU on a “Deny list”).~~

11 ~~Several possible methods to restrict ONU's access to the network are described in 11.2.3. is unauthorized to~~  
12 ~~connect to the network from any location;~~

13 ~~An authenticated and authorized ONU that has been discovered on a PON port different from the one on~~  
14 ~~which it is authorized.~~

15 ~~The OLT may analyze various ONU parameters, capabilities, and operational characteristics of the ONU in~~  
16 ~~addition to the ONU identity and PON port to determine its access to the PON and connected resources.~~  
17 ~~For example, the OLT may compare the ONU's round-trip time to the previous measurement, deny access~~  
18 ~~to the ONU that has unexpectedly relocated and/or require the ONU to repeat the onboarding process.~~

#### 19 **11.2.2.1 ONU ~~Authentication authentication Credentials credentials~~**

20 The ONU may be authenticated using the built-in credential (the Device Authentication Credential) or an  
21 operator-provided credential (the Network Authentication Credential). In either case, the ONU identity is  
22 attested via the built-in keypair (the Device Authentication Keypair). When properly used, these credentials  
23 provide the ability for an NMS to cryptographically verify the identity of the ONU attempting to access the  
24 PON and determine what access the ONU is granted.

11.2.2.1.1

#### 25 **The Device Authentication Keypair (DAK) ~~R~~requirements**

26 The Device Authentication Keypair (DAK) facilitates robust identification and authentication of ONUs on  
27 the PON network.

28 All SIEPON.4 ONUs and OLTs shall have a unique DAK of type *Curve P-384* (see FIPS 186-4, D.2.4).  
29 ~~The keypair shall be~~ generated in accordance to NIST ST SP 800-57 Part 1R5 section 8.1.5.1. The DAK  
30 ~~shall~~ persist across device power cycles and factory resets.

31 Requirements for securing the DAK private key can be found in ~~11.10.1.1.2~~.

#### 32 **Credential Identification and Selection**

33 In order to provide the NMS/OLT the ability to select the particular ONU credential to use for  
34 authentication, the TLS 1.3 CertificateRequest extension “SIEPON.4 Credential Type” is defined in the  
35 IEEE namespace as OID 1.3.111.2.1904.4.1.1 with the following definition:

```
36 enum {  
37     undefined(0),  
38     dac(1),  
39     nac(2),
```

```
1         reserved(3..255)
2     } SIEPON4CredentialType;
```

### 3 The Device Authentication Credential (DAC) Requirements

4 The DAC is a data structure containing the DAK public key, metadata identifying the ~~PON-ONU~~ device  
5 (including the *aOnuld* attribute), and a cryptographic signature(s) used to verify the authenticity of the  
6 DAC. The DAC shall meet the following requirements:

#### 7 11.2.2.1.3

8 — ~~be formatted and authenticated~~ in accordance to X.509v3 (see X.509/RFC-6818).  
9 Specifically,

10 — ~~Shall contain~~ the *SIEPON4CredentialType* extension with a value of “1” (dac). (see  
11 11.3.3.2)

12 — ~~For ONUs, the~~ Subject Common Name (CN) field shall conform to with the PrintableString  
13 definition as described in RFC-5280 and shall contain the *aOnuld* attribute value encoded  
14 into 12 hexadecimal digits preceded by the string “SIEPON4\_ONU\_”. For example,  
“CN=SIEPON4\_ONU\_0A7FB49E2CF1” (see RFC-4648, section 8)

15 — The public key field shall contain the DAK public key of the device encoded according to  
16 RFC-5480, section 2.1.

17 — The DAC ~~must be~~ is signed using the device’s DAK private key producing an ECDSA  
18 signature with a SHA-256, SHA-384, or SHA-512 HMAC according to RFC-5480, section  
19 2.1.

20 ~~The DAC may also be signed using a device manufacturer’s CA private key producing an~~  
21 ~~ECDSA signature with a SHA-256, SHA-384, or SHA-512 HMAC according to RFC-5480,~~  
22 ~~section 2.1.~~

23 — The Key Usage Extension shall indicate the certificate’s public key usage is “Digital  
24 Signature” and “Key Encipherment”. (see RFC-5280, section 4.2.1.12).

25 — The total size of the DAC shall does not exceed 1491 ~~bytes~~octets.

26 — The certificate shall does not include any extensions marked “critical” unless required by  
27 RFC-5280 or required above.

28 The DAC may also be signed using a device manufacturer’s CA private key producing an ECDSA  
29 signature with a SHA-256, SHA-384, or SHA-512 HMAC according to RFC-5480, section 2.1.

#### 30 11.2.2.1.4

31 ~~Every ONU shall present a DAC during the authentication process (see).~~ An example DAC is provided in  
32 Annex 11A, TBD.

### 33 The Network Authentication Credential (NAC) Requirements

34 The Network Authentication Credential (NAC) is ~~a data structure containing~~ an end-entity certificate (see  
35 RFC-5280) containing the DAK public key, operator-defined metadata, and a cryptographic signature used  
36 to verify the authenticity of the NAC.

37 The network operator may create a NAC for an ONU to contain network operator-defined metadata. The  
38 NAC is signed using an operator-defined Certificate Authority (CA) – enabling the ONU to be validated  
39 using a network operator-defined Public Key Infrastructure (PKI) system. For example, a network operator  
may create an NAC containing a unique serial number, an alternate operator-defined identify for the ONU,

1 identify the customer or management entity associated with the ONU, the ONU's assigned service level,  
2 and/or the network locale the ONU can operate within. An example NAC is provided in Annex IIA.1B9.

3 A NAC may be pre-installed into an ONU before the ONU's deployment/installation, or it may be installed  
4 remotely after the ONU has successfully authenticated using the built-in DAC (see 1.2.2.1.3).

5 To ensure that ONUs do not operate with expired NACs, a network operator may revoke the existing NAC  
6 and/or install a new NAC in the ONU at any time using the *eOAM Install NAC Request* eOAMPDU (see  
7 13.4.6.7.1). Intermediate CA certificates and the root CA can also be uploaded along with the NAC as long  
8 as the total size of the certificate chain does not exceed the specified maximum NAC length. NAC creation  
9 is facilitated by the *eOAM Retrieve DAC Request* eOAMPDU (see 13.4.6.7.3), which allows for on-  
10 demand retrieval of the ONU's DAC, containing the DAK public key.

11 The OLT shall not generate the *eOAM Install NAC Request* eOAMPDU containing the NAC that

12 — Is not formatted in accordance with the RFC-5280

13 — Has size exceeding 1489 octets

14 — Is not signed using an ECC Named Curve as defined in RFC-5480, section 2.1.1.1.

15 The ONU shall set the field *CertificateStatus* in the *eOAM Install NAC Response* eOAMPDU  
16 (see 13.4.6.7.2) to the value *Invalid Format (0x03)*, if the received NAC

17 — Is not formatted in accordance with the RFC-5280

18 — Contains a public key that does not match the ONU's DAK public key

19 — Does not contain the *SIEPON4CredentialType* extension with a value of "2 (nac)". (see  
20 11.3.3.2)

21 ~~The NAC should be formatted and authenticated in accordance to RFC 5280. Specifically,~~

22 ~~— The public key field of the NAC leaf certificate should contain the DAK public key of the~~  
23 ~~device encoded according to RFC 5480, section 2.1.~~

24 ~~— Should contain the *SIEPON4CredentialType* extension with a value of "2 (nac)". (see~~  
25 ~~11.3.3.2)~~

26 ~~— The total size of the NAC should not exceed 1491 octets.~~

27 ~~— Should be signed using an ECC Named Curve as defined in section 2.1.1.1.~~

28 ~~The network operator may create a NAC for an ONU to contain network operator defined metadata. The~~  
29 ~~NAC shall be signed using an operator defined Certificate Authority (CA) enabling the ONU to be~~  
30 ~~validated using a network operator defined Public Key Infrastructure (PKI) system. For example, a network~~  
31 ~~operator may create an NAC containing a unique serial number, alternate operator defined identify for the~~  
32 ~~ONU, identify the customer or management entity associated with the ONU, the ONU's assigned service~~  
33 ~~level, and/or the network locale the ONU can operate within. Note that any PKI system is expected to~~  
34 ~~incorporate a robust certificate renewal and revocation system to ensure that ONUs do not operate with~~  
35 ~~expired NACs and to enable ONUs to be disabled on demand. An example NAC is provided in Annex IIA~~  
36 ~~1B9. Once an ONU is authenticated, a network operator may create, install, renew, or revise a NAC on~~  
37 ~~the ONU at any time using the *eOAM Install NAC Request* eOAMPDU (see 13.4.6.7.1). Intermediate CA~~  
38 ~~certificates and the root CA can also be uploaded along with the NAC so long as the total size of the~~

1 ~~certificate chain does not exceed the maximum NAC length. NAC creation is facilitated by the~~  
2 ~~eOAM\_Retrieve\_DAC\_Request eOAMPDU (see 13.4.6.7.3), which allows for on-demand retrieval of the~~  
3 ~~ONU's DAC, containing the DAK public key.~~

#### 4 **Network Authentication Credential (NAC) Intermediate Certificates**

5 The Network Authentication Credential (NAC) may reference intermediate certificates. ~~The -which a NMS~~  
6 ~~way may install the intermediate certificates together with the NAC certificate into an ONU want to include~~  
7 ~~with the NAC certificate~~ for the purposes of authenticating the NAC. For example, intermediate ~~certs~~  
8 ~~certificates~~ can be included to enable the authentication of NACs signed by intermediate CAs with AAA  
9 servers ~~that which~~ only have root certificates in their trust store. NAC intermediate certificates ~~shall be~~  
10 formatted and authenticated in accordance to RFC-5280.

11 ~~Note that the OLT is able to include the intermediate certificates in the eOAM Install NAC Request~~  
12 ~~eOAMPDU (see 13.4.6.7.1) only if the total size of the encoded certificate chain (i.e., the NAC certificate~~  
13 ~~and all the intermediate certificates) does not exceed 1489 octets.~~

14 ~~Specifically, to limit the size of intermediate certificates, NAC intermediate certificates~~

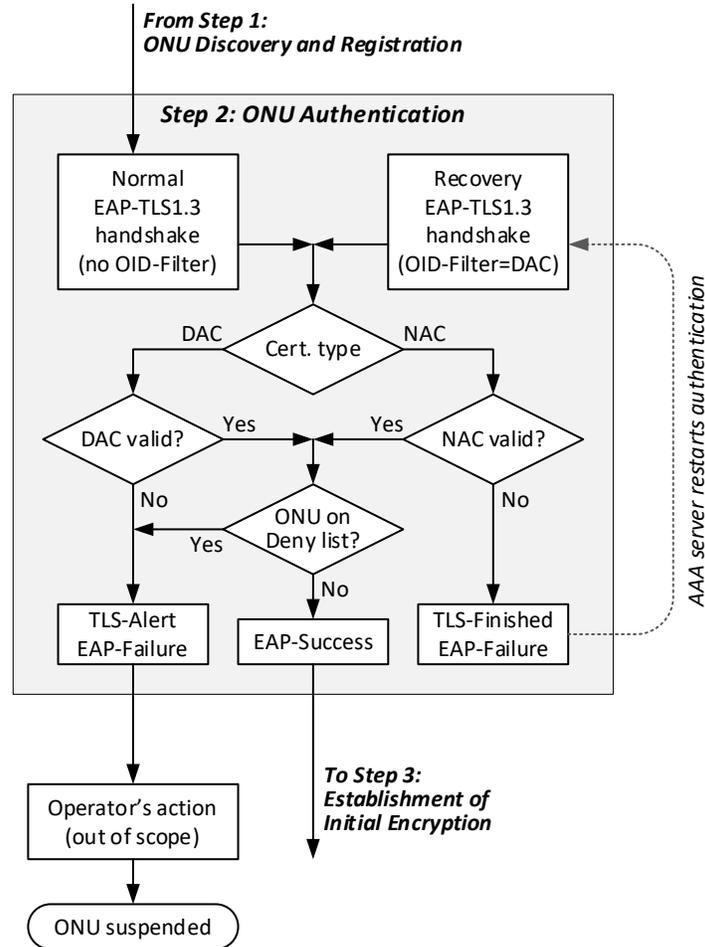
15 ~~— Should contain an ECC public key as defined in RFC 5280 and utilize a Named Curve as~~  
16 ~~defined in section 2.1.1.1.~~

17 ~~— Should be signed using an ECC Named Curve as defined in section 2.1.1.1.~~

#### 18 **11.2.2.2 The ONU Authentication Procedure**

19 ~~Authentication in SIEPON.4 is accomplished using EAP-TLS 1.3 (see RFC 9190) with the OLT operating~~  
20 ~~as the "EAP Authenticator"/"EAP-TLS Server" and the ONU operating as the "EAP Supplicant"/"EAP-~~  
21 ~~TLS Client" (see RFC 3748). By utilizing TLS 1.3 as described in this clause, the TLS handshake is~~  
22 ~~encrypted and authenticated using an ephemeral key that provides a shared secret to both parties while~~  
23 ~~ensuring privacy and perfect forward secrecy (PFS). The shared secret is then used to derive the SIEPON.4~~  
24 ~~initial traffic encryption key, as described in 11.4.~~

25 In order to perform EAP-based authentication, the OLT and ONU exchange EAPOL (Extensible  
26 Authentication Protocol over LAN) frames over the MLID (see IEEE Std 802.1X, Clause 11.). ~~The~~  
27 ~~authentication procedure is illustrated in Figure 11-5.~~



**Figure 11-5 — ONU authentication procedure**

1  
2  
3 Additionally, theThe EAP authentication ~~process-procedure~~ shall be performed according to the following  
4 guidelinesrequirements:

- 5 — The OLT and ONU shall support EAP-TLS 1.3 as defined in RFC-9190 and use EAP type EAP-  
6 TLS (type 13) in all EAP-Request and EAP-Response messages. EAP-Request/Response for  
7 Identity (type 1) are not to be ~~supported-used~~ by the ONU or OLT and the ONU shall return an  
8 EAP-Response/Nak to any EAP-Request/Identity messages.
- 9 — The OLT and ONU shall only advertise TLS 1.3 in their respective TLS ClientHello and  
10 ServerHello messages (exchanged via EAP), as defined in RFC-8446. The  
11 legacy\_version field shall be set to 0x0303 and the supported\_versions extension  
12 shall include TLS 1.3 (versions value 0x0304).
- 13 — The OLT and ONU may utilize any TLS 1.3 supported and negotiated DHE key exchange method.  
14 Session resumption using PSK-based key exchange methods are not defined for use in SIEPON.4  
15 at this time but may be supported in the future.
- 16 — In order to support ONU authentication, the OLT shall issue and the ONU shall honor the  
17 CertificateRequest message, as described in TLS 1.3, section 4.3.2.

- 1 — The ONU shall support TLS 1.3 OID Filters extension as described in TLS 1.3 (see RFC 8446,  
2 section 4.2.5). The NMS/OLT may use this to perform credential selection, When a  
3 SIEPON4CredentialType is present in the TLS 1.3 CertificateRequest OID Filters  
4 extension (see RFC-8446, section 4.2.5), the filter shall only be considered matched against an  
5 ONU credential if the ONU has an authentication certificate containing the  
6 SIEPON4CredentialType extension with the same value as the one in the  
7 CertificateRequest.
- 8 — If the ONU is not configured with a NAC, or the DAC is matched via the TLS  
9 CertificateRequest OID Filter (see 11.2.2.1.2), the ONU shall include the DAC in its TLS  
10 1.3 Certificate message as the “end-entity” certificate in the certificate\_list.  
11 Requirements for the DAC can be found in 11.2.2.1.5.
- 12 — If the ONU is configured with an NAC, and the DAC is not explicitly selected via a TLS  
13 CertificateRequest OID Filter (see 11.2.2.1.2+11.3.3.2), then the ONU shall include the  
14 NAC in its TLS 1.3 Certificate message as the end-entity certificate with any intermediate  
15 certificates following the NAC in the certificate list. Requirements for the NAC and  
16 associated intermediate certificates can be found in 11.2.2.1.4 and 11.2.2.1.5, respectively.
- 17 — If the ONU is not configured with a credential that is explicitly selected via the TLS  
18 CertificateRequest OID Filter (see 11.2.2.1.2+11.3.3.2), the ONU shall abort the handshake  
19 with an “unsupported\_certificate” alert.

20 ~~An~~The OLT ~~can~~ initiates the EAP authentication process ~~at a time of its choosing~~ by issuing an EAP-  
21 Request with an EAP-Type of EAP-TLS (type 13). The OLT shall not initiate another EAP authentication  
22 session until any ongoing authentication session has been completed with either Success or Failure, per  
23 RFC-3748, section 2.1.

### 24 **11.2.3 ONU authorization**

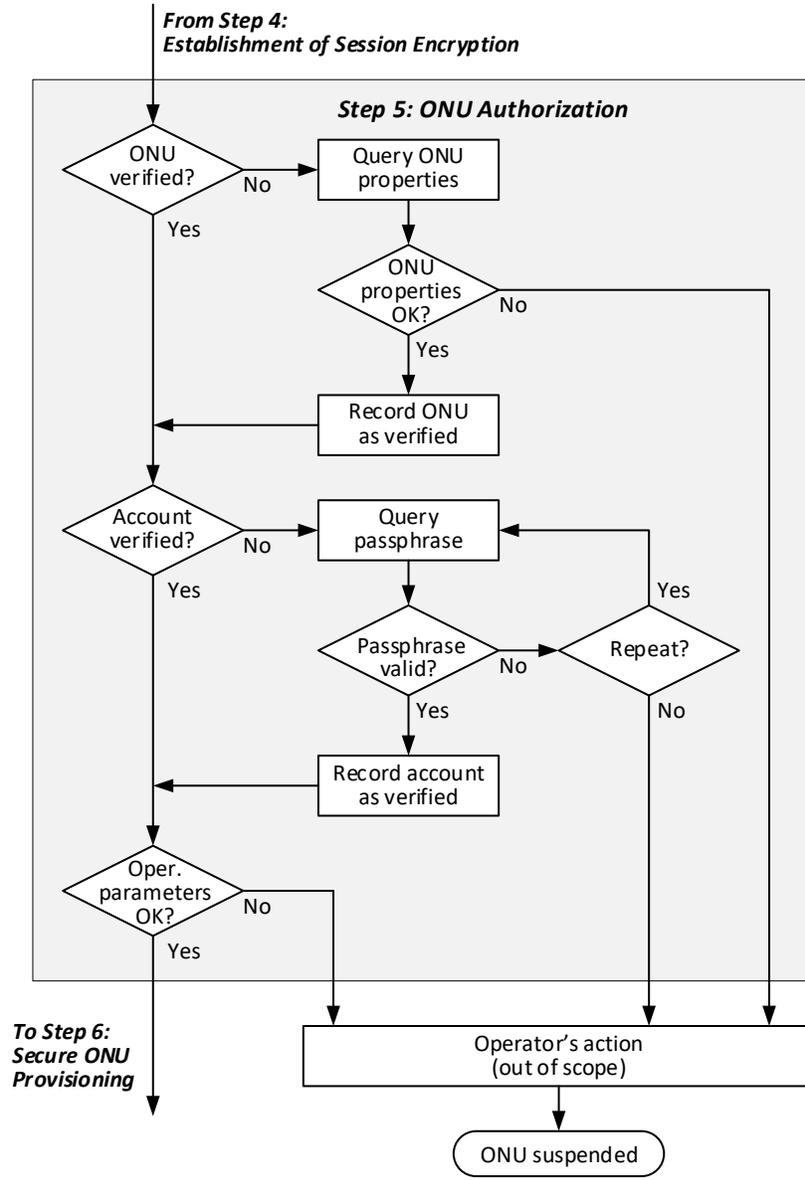
25 ONU authorization refers to the verification of ONU’s eligibility to operate on the operator’s network. The  
26 determination of ONU’s eligibility is based on a reasonable assurance that the ONU is compatible and  
27 interoperable with the rest of equipment used in the operator’s network, that the user served by this ONU is  
28 eligible to receive service, and that the ONU operates within the limits established by the operator.

29 ONU authorization is initiated by the NMS/AAA server after the ONU has successfully authenticated and  
30 symmetric encryption has been established for secure communication between the OLT and the ONU. Only  
31 two “system” logical links (PLID and MLID) are configured at the ONU at the time of the ONU’s  
32 authorization. Any exchange of information between the OLT and ONU pertaining to the ONU  
33 authorization is carried over the encrypted MLID logical link.

34 Subject to operator’s policies, the ONU authorization procedure maybe modified or bypassed altogether.

#### 35 **11.2.3.1 Authorization procedure**

36 The ONU authorization procedure generally comprises the verification of ONU properties and capabilities  
37 (see 11.2.5.1.1), the verification of user account / service eligibility (see 11.2.5.1.2), and the verification of  
38 ONU’s operational parameters (see 11.2.5.1.3). If an ONU fails the get authorized, its access to the  
39 network shall be restricted, as described in 11.2.4. The authorization procedure is illustrated in Figure 11-6.



1

2 11.2.3.1.1

3

4

#### **Verification of ONU properties and capabilities.**

5

This step ensures that the ONU device is compatible and interoperable with the rest of equipment used in the operator's network. The verification typically involves checking ONU's manufacturer and/or model to confirm that it belongs to the class of ONUs that were pre-qualified to operate on a given network. Various ONU capabilities may also be queried and checked to ensure that the ONU is able to support the necessary services.

10

The ONU properties and capabilities are immutable characteristics of the device. Therefore, this step is generally performed only once, in the process of onboarding of a new ONU (see 11.2.3.4). It is also possible to perform this step offline, prior to ONU deployment, as explained in 11.2.3.2.1.2.

12

1                   **Verification of user account / service eligibility.**

2     This step associates the identity of the ONU device with a user account and verifies the eligibility of the  
3     ONU to be provisioned to service user data. The association between the ONU with the user account needs  
4     to be established only once, in the process of onboarding of a new ONU (see 1.2.3.2). There could be  
5     various operator-specific methods of establishing such association, for example:

6                   **11.2.3.1.2**

7                   — The ONU's serial number / MAC address can be scanned/retrieved from the ONU and  
8                   associated with a known customer account at the time of installation by either a technician or  
9                   customer, facilitated by the use of an external device with Internet access.

10                  — A unique activation/association passphrase can be generated by the NMS and conveyed to the  
11                  customer prior to ONU installation. During the ONU onboarding phase, the user is prompted  
12                  to enter the passphrase via the CPE device connected to the ONU. The passphrase together  
13                  with the ONU identity is delivered via the eOAM channel to NMS, allowing the association  
14                  to be established (see 1.2.3.2). This method may be particularly necessary in situations  
15                  where no Internet connectivity is available, except through the ONU being deployed.

16     It is also possible to associate ONU identity with a user account offline, prior to ONU deployment, as  
17     explained in 1.2.3.2.1.3.

18                   **Verification of ONU's operational parameters.**

19                   **11.2.3.1.3**

20     Optionally, operators may desire to analyze various operational characteristics of the ONU to determine its  
21     eligibility to access the PON and connected resources. For example, OLT may verify that given ONU is  
22     authorized for the particular OLT PON port on which it is discovered (i.e., a port-based access control).  
23     Alternatively, or in addition to the PON port verification, the OLT may compare the ONU's round-trip time  
24     to the previous measurement and deny access to the ONU that has unexpectedly relocated.

25     The verification of ONU's operational parameters is performed every time the ONU is restarted, not just  
26     during the initial onboarding.

27                   **11.2.3.1.4**

28                   **ONU authorization requirements**

29     The OLT shall be able to support port-based access control and should be able to support RTT-based access  
30     control.

31     If port-based access control is enabled, the OLT shall verify that given ONU is authorized for the particular  
32     OLT PON port on which it is discovered.

33     If RTT-based access control is supported and enabled, the OLT shall keep track of ONU's RTT value  
34     measured during previous ONU registration and it shall compare the newly-measured RTT to the previous  
35     RTT value.

36     The OLT shall restrict network access to the following categories of ONUs:

37                   — The ONU whose properties and capabilities were found unacceptable for the given network;

                    — An ONU that could not be associated with a valid user account;

                    — The ONU that has been re-discovered on a PON port different from the one on which it is  
                    authorized (only if PON port-based authorization is enabled);

1           — The ONU that has unexpectedly relocated, i.e., whose measured round-trip time is  
2           significantly different from the previous measurement (only if RTT-based authorization is  
3           supported and is enabled).

4           Several possible methods to restrict ONU's access to the network are described in [11.2.4](#).

5           The OLT may consult NMS/AAA servers to determine the ONU's authorization to access the network  
6           resources. The mechanism of communication with the NMS/AAA servers or the nature of information  
7           passed between the OLT and the servers is outside the scope of this standard.

### 8           **11.2.3.2 Onboarding of a new ONU**

9           Within the context of this standard, the term *onboarding* refers to the process of integrating a newly-  
10          installed ONU into the established operator's network. A number of steps related to ONU authorization to  
11          operate on a given network are performed only once, during the ONU onboarding. However, an operator  
12          has the ability to repeat the onboarding procedure for any ONU, even after the ONU has already been  
13          onboarded.

### 14          **ONU deployment scenarios**

15         ~~11.2.3.2.1~~ The steps required during the ONU onboarding are determined by the ONU deployment scenario adopted  
16         by the network operator. Several typical ONU deployment scenarios are described in sub-clauses  
17         [11.2.3.2.1.1](#) through [11.2.3.2.1.4](#), although other scenarios are also possible.

#### 18         **11.2.3.2.1.1 Unknown/generic ONU**

19         The unknown/generic ONU deployment scenario refers to deployment of an ONU that has not been in the  
20         possession of the network operator prior to deployment, as is the case of a customer-procured ONU. In  
21         other words, this is a scenario where the operator has no apriori information about the ONU properties and  
22         capabilities or about which user account the ONU is associated with.

23         In this scenario, the ONU authorization includes all of the following steps: the verification of ONU  
24         properties and capabilities (see [11.2.3.1.4](#)), the verification of user account / service eligibility (see  
25         [11.2.3.1.3](#)), and the verification of ONU's operational parameters (see [11.2.3.1.3](#)).

#### 26         **11.2.3.2.1.2 Pre-validated ONU**

27         The *pre-validated ONU* deployment scenario refers to deployment of ONUs whose properties and  
28         capabilities have been verified by the operator prior to deployment. An individual ONU is selected for the  
29         installation from a pool of verified ONUs, although it is not known which exact ONU will be deployed to a  
30         specific customer until the installation time.

31         In this scenario, the AAA server maintains a list of identities of the pre-verified ONUs. During the ONU  
32         authorization, the AAA server recognizes the ONU identity as pre-verified. Therefore, the ONU  
33         authorization includes only the following steps: the verification of user account / service eligibility (see  
34         [11.2.3.1.2](#)) and the verification of ONU's operational parameters (see [11.2.3.1.3](#)).

#### 35         **11.2.3.2.1.3 Pre-validated user account**

36         The *pre-validated user account* refers to a deployment scenario where the customer is required to register  
37         its customer-procured ONU before the ONU's installation. The registration would typically involve a  
38         submission of the ONU's serial number (or some other form of ONU identity) along with the customer  
39         account number (or some other form of customer identity, such as a unique activation code or a passphrase  
40         provided by the operator). The information submitted by the customer is used to associate the ONU identity

1 with customer identity and this association is recorded in the AAA server. However, no information is  
2 available to the operator regarding the ONU's properties and capabilities.

3 During the ONU authorization procedure, the AAA server finds the record of a verified user account  
4 associated with the identity of the ONU being authorized. Therefore, the ONU authorization includes only  
5 these steps: the verification of ONU properties and capabilities (see 11.2.3.1.) and the verification of  
6 ONU's operational parameters (see 11.2.3.3).

#### 7 **11.2.3.2.1.4 Pre-authorized ONU**

8 The *pre-authorized ONU* refers to a deployment scenario whereby an operator selects a specific ONU to be  
9 deployed to a specific customer. Before the ONU shipment / deployment, a record is made and stored in the  
10 AAA server reflecting the fact that the ONU is pre-validated (see 11.2.3.2.1.2) and the associated user  
11 account is also pre-validated (see 11.2.3.2.1.3).

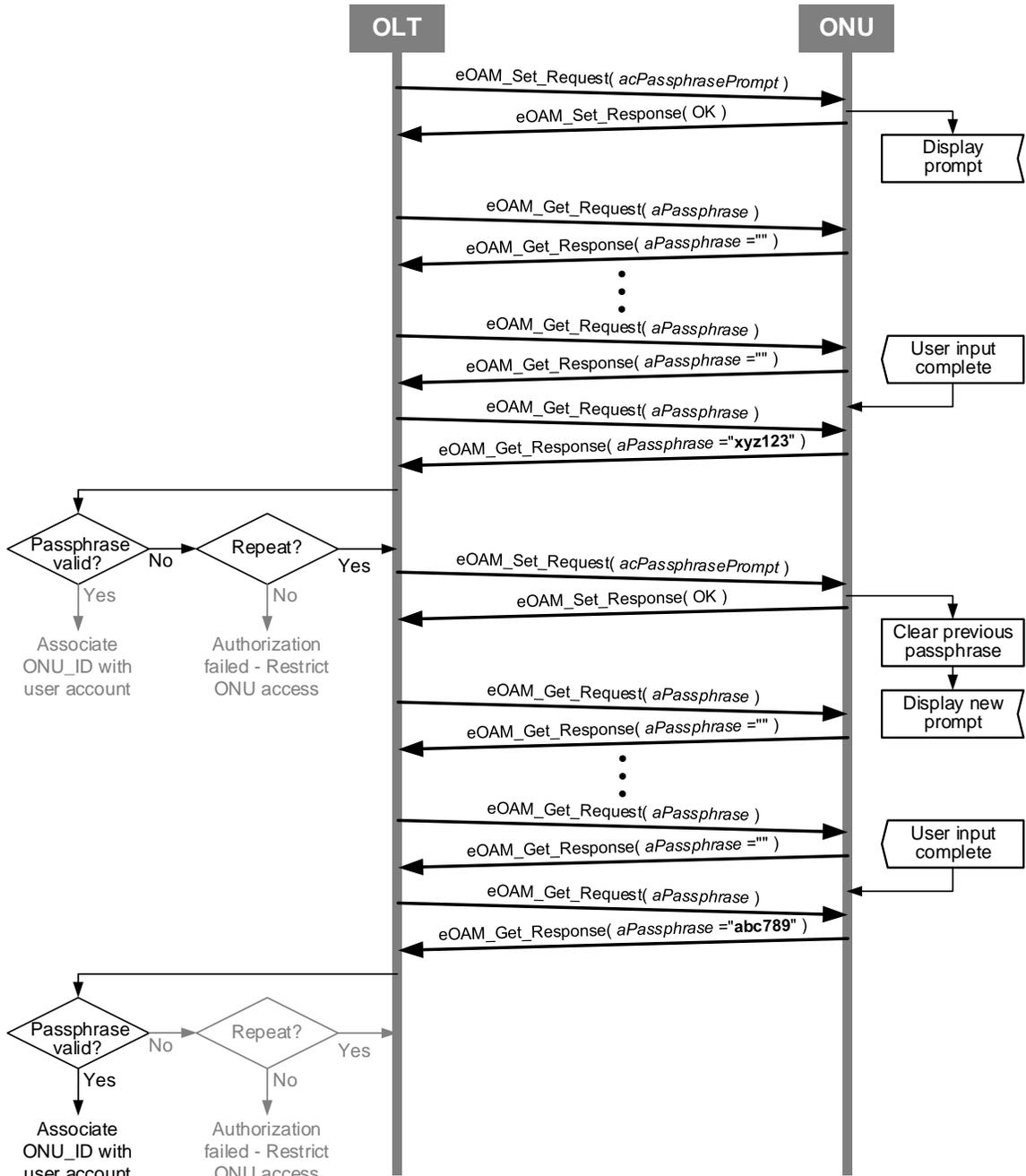
12 During the ONU authorization procedure, the AAA server finds the record matching the identity of the new  
13 ONU, and confirming the verification status of the ONU as well as the verification status of the user  
14 account associated that ONU. Therefore, the ONU authorization procedure includes only the verification of  
15 ONU's operational parameters (see 11.2.3.3).

#### 16 **Passphrase-based user account verification**

17 ~~11.2.3.2.2~~ As explained in 11.2.3.1.2, operator may rely on a customer-unique passphrase as a method to associate  
18 ONU identity with a customer account or service profile. In this method, during the ONU onboarding phase,  
19 the user is prompted to enter the passphrase via the CPE device connected to the ONU. The passphrase  
20 together with the ONU identity is delivered via the eOAM channel to NMS, allowing the association to be  
21 established (see 11.2.3.2.2).

22 The passphrase-based account verification method relies on the extended action *acPassphrasePrompt* (see  
23 14.6.5.3) and the extended read-only attribute *aPassphrase* (see 14.4.5.2), and is performed over an  
24 encrypted MLID logical link.

25 The *acPassphrasePrompt* action displays an NMS-specified prompt to the user and also resets the value of  
26 any stored passphrase to a zero-length string. The *aPassphrase* action allows the retrieval of the user-  
27 entered string. As the time required for a user to manually enter the passphrase is expected to be much  
28 longer than the OAMPDU response timeout (see 13.2.3), it is expected that *aPassphrase* attribute is  
29 queried multiple times before the input is received, as illustrated in Figure 11.7.



1

2

**Figure 11.7 — Passphrase query protocol**

3

**11.2.4 ONU access restriction**

4

The ONU that failed the authentication is unable to derive the initial encryption key, and as a result, is unable to establish the symmetric encryption with the OLT. Correspondingly, the NMS does not initiate the authorization procedure for such ONU. The ONU that has successfully authenticated may fail the subsequent authorization.

5

6

7

8

An operator may resort to different actions in response to ONU’s failure to get authenticated or authorized.

1 One possible action is to withhold service configuration/provisioning to the ONU. Such ONU remains  
2 registered with two active LLIDs: primary PLID and primary MLID. The ONU is able to process GATE  
3 MPCPDUs and generate REPORT MPCPDUs, and it is able to exchange OAMPDUs with the OLT, but is  
4 unable to service any user data. Such ONU does consume a small amount of PON transmission capacity.

5 Alternatively, the operator may suspend the ONU. The suspended ONU is deregistered and does not  
6 attempt to register again, until instructed to do so. The ONU suspension method is described in 11.2.4.1.

#### 7 **11.2.4.1 Suspension and resumption of ONU operation**

8 The ONU registration status is controlled via the REGISTER MPCPDU (see IEEE 802.3, 144.3.6.4). The  
9 specific ONU behavior is determined by the value of the *Flag* field. Per IEEE 802.3, 144.3.7:

10 If the *Flag* field in the REGISTER MPCPDU has the value of ACK, the ONU performs a fast  
11 registration sequence where it simply responds with the REGISTER\_ACK MPCPDU, while  
12 remaining registered at all times.

13 If the *Flag* field in the REGISTER MPCPDU has the value of NACK, the ONU deregisters and  
14 goes through a complete discovery sequence, as outlined above.

15 This standard extends the definition of the *Flag* field by adding an additional value SUSPEND (2) as  
16 defined in Table 11-xx. The Table 11-xx replaces the table 144-5 in IEEE 802.3:

17 **Table 11-xx—REGISTER MPCPDU *Flag* field**

<u>Value</u>	<u>Indication</u>	<u>Notes</u>
<u>0</u>	<u>ACK</u>	<u>The ONU's requested registration is successful or a registered ONU is asked to re-register</u>
<u>1</u>	<u>NACK</u>	<u>The registration request is denied or a registered ONU is asked to deregister and register again.</u>
<u>2</u>	<u>SUSPEND</u>	<u>The ONU is requested to deregister and to not attempt to register again.</u>
<u>3 to 255</u>	<u>Reserved</u>	<u>Ignored on reception</u>

18  
19 The REGISTER MPCPDU with the *Flag* field value equal to SUSPEND shall be transmitted to the  
20 destination ONU in an envelope with unicast PLID assigned to this ONU.

21 The ONU that receives REGISTER MPCPDU with the *Flag* field value equal to SUSPEND shall  
22 deregister (i.e., release its assigned PLID and MLID values). The suspended ONU shall begin to receive  
23 and process the envelopes sent to the DISC PLID logical link, but it shall not attempt to register until either  
24 of the following events has occurred:

25 — The ONU was factory-reset. (The factory-reset procedure is outside the scope of this  
26 standard.)

27 — The ONU received a unicast DISCOVERY MPCPDU (see IEEE 802.3, 144.3.6.6), i.e., a  
28 DISCOVERY MPCPDU sent to DISC\_PLID with a unicast MAC DA matching the identity  
29 of the given ONU (11.2.1).

30 The ONU shall maintain its suspension state across the restart or reboot. The ONU may remain in the  
31 suspended state for an unlimited period of time.

## 11.3 Encryption

### 11.3.1 Overview of encryption architecture

#### 11.3.1.1 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.

#### 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.

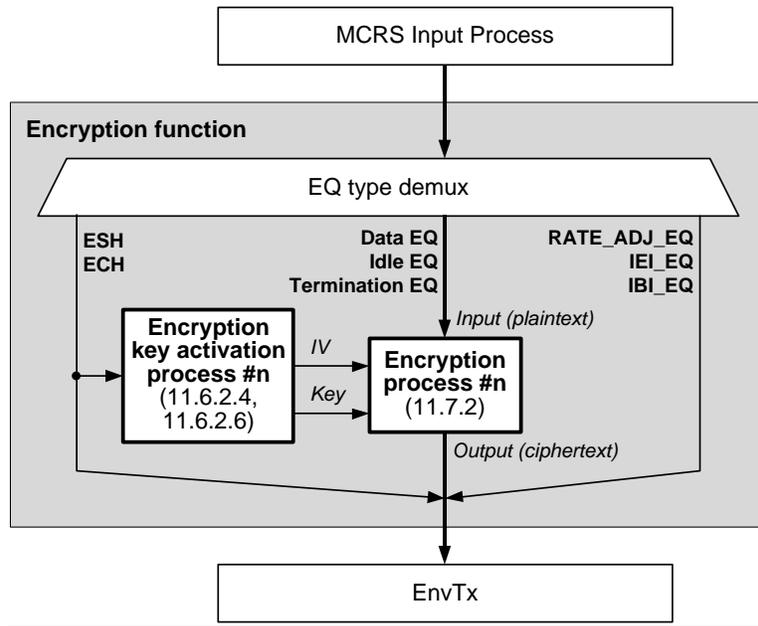
The unidirectional logical links are typically provisioned as point-to-multipoint (P2MP) links and carry downstream multicast traffic. Each envelope transmitted by the OLT is delivered to multiple ONUs. Therefore, the encryption key used to encrypt the multicast traffic needs to be shared among all ONUs 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  $U$  ONUs and is provisioned to use  $M$  multicast LLIDs instantiates  $U + M$  encryption entities.

#### 11.3.1.2 Location of encryption/decryption functions

The Multi-channel Reconciliation Sublayer (MCRS) reconciles  $L$  logical links (i.e., MAC instances) above the sublayer with  $C$  physical layer channels below it. The 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<sub>SEC</sub>) sublayer. The encryption function is located in the transmit path of the MCRS<sub>SEC</sub> sublayer, as illustrated in Figure 11-1(a), and the decryption function is located in the receive path of the MCRS<sub>SEC</sub> sublayer, as illustrated in Figure 11-1(b).





1

2

**Figure 11-2–Encryption function block diagram.**

3

The Encryption Key Activation process detects the transmission of either the envelope start header (ESH) or the envelope continuation header (ECH) and uses the data inside the header to generate an initialization vector IV and the encryption key. These two parameters are passed to the Encryption process, which encrypts the given envelope as one cryptographic message.

6

7

The encryption function encrypts the envelope payload, which consists of data EQs, idle EQs, and termination EQs. The envelope header itself bypasses the Encryption process and is transmitted unencrypted (see 11.7.5.2).

9

10

The inter-envelope control EQs include rate adjustment (RATE ADJ EQ), inter-envelope idle (IEI EQ), and inter-burst idle (IBI EQ) (see 11.7.5.1). These control EQs bypass the Encryption process and are transmitted unencrypted.

12

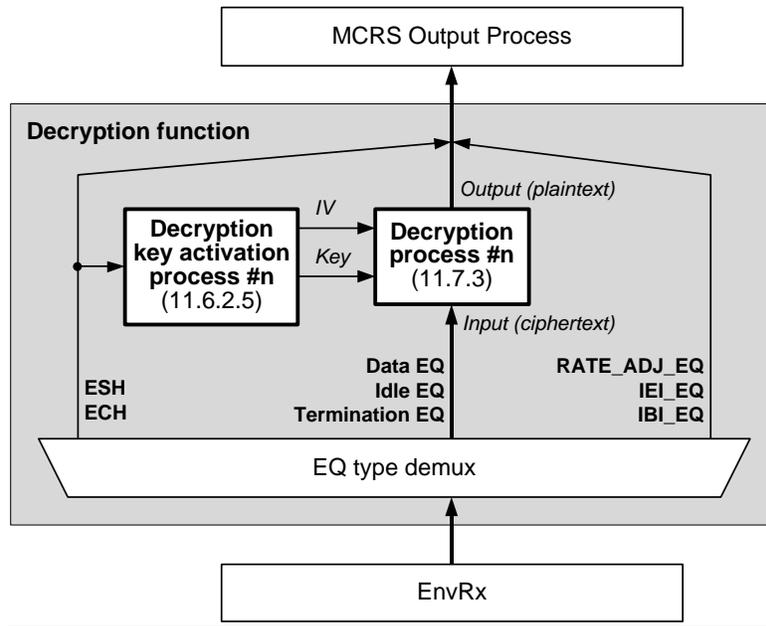
13

#### **11.3.1.4 Decryption function block diagram**

14

Each instance of the decryption function includes the Decryption Key Activation process (see 11.6.2.5) and the Decryption process (see 11.7.3). The block diagram of the decryption function is illustrated in Figure 11-3.

16



1

2

**Figure 11-3–Decryption function block diagram.**

3

The Decryption Key Activation process detects the reception of either the envelope start header (ESH) or the envelope continuation header (ECH) and uses the data inside the header to generate an initialization vector IV and the decryption key. These two parameters are passed to the Decryption process, which decrypts the given envelope as one cryptographic message.

4

5

6

7

The decryption function decrypts only the envelope payload, which consists of data EQs, idle EQs, and termination EQs. The envelope header itself is received unencrypted and bypasses the Decryption process (see 11.7.5.2).

8

9

10

The inter-envelope control EQs include rate adjustment (RATE ADJ EQ), inter-envelope idle (IEI EQ), and inter-burst idle (IBI EQ) (see 11.7.5.1). These control EQs are received unencrypted and bypass the Decryption process.

11

12

### **11.3.1.5 Latency requirements**

13

14

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.

15

16

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.

17

### **11.3.2 Initial key establishment**

18

19

The initial key is used by the OLT to encrypt the MLID channel for subsequent distribution of the first session key to the ONU (see step 4 in Figure 11-4). The establishment of the initial key involves the derivation of the initial key value (see 11.3.2.1) and the activation of the initial key (see 11.3.2.2).

20

21

### 1 11.3.2.1 Initial key derivation

2 Once the ONU and OLT have completed the authentication exchange, the initial AES-128 encryption key  
3 shall be established by both parties from the least-significant 128 bits (16 octets) of the MSK, which is  
4 derived from the TLS 1.3 ephemeral session key as described in RFC-9190, section 2.3.

5 ~~The initial key is used by the OLT to encrypt the MLID channel for distribution of a new session key to the~~  
6 ~~ONU. See for details on the session key distribution protocol~~

### 7 11.3.2.2 Initial key activation

8 ~~The initial key is activated using the same procedure as defined for the session key activation (see 11.3.4).~~  
9 ~~The ONU stores the derived value of the initial key in the key memory The OLT stores the derived value of~~  
10 ~~the initial key in their local key memories, in the location keys[ee][0] (see 11.6.2.1), where [ee] is~~  
11 ~~an index of the encryption entity associated with the bidirectional logical link(s) between the OLT and the~~  
12 ~~ONU.~~

13 ~~For the ONU to be able to decrypt the encrypted envelopes, its cipher clocks need to be synchronized with~~  
14 ~~that in the OLT, as detailed in 11.3.5.4.1. The cipher clock synchronization relies on an OAMPDU~~  
15 ~~containing the Sync Cipher Clock TLV sent from the OLT to the ONU. Before activating the encryption in~~  
16 ~~the downstream direction, the OLT shall receive a positive acknowledgement from the ONU, indicating~~  
17 ~~that the cipher clocks have been synchronized.~~

18 ~~After the initial key is written into the key memory and a positive acknowledgement of cipher clock~~  
19 ~~synchronization is received, the OLT sets the initialKeyReady[ee] variable to true. The initial~~  
20 ~~key is activated in the downstream direction first, when the OLT transmits an envelope encrypted with this~~  
21 ~~key (see 11.3.4.1).~~

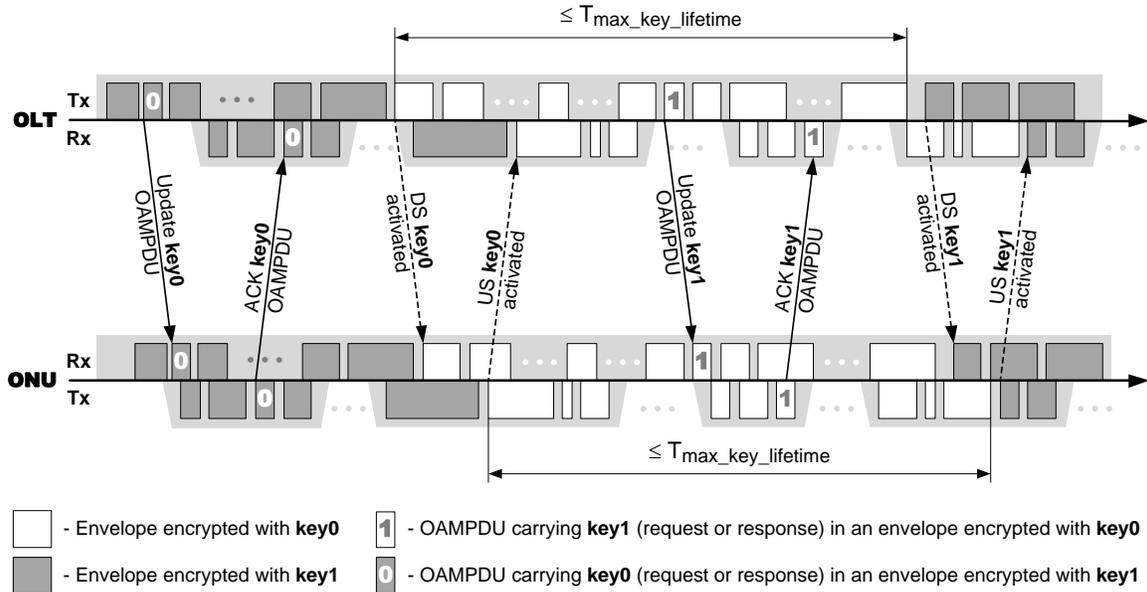
## 22 11.3.3 Session key distribution protocol

### 23 11.3.3.1 Protocol overview

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

26 After the initial key exchange (defined in 11.3), all the subsequent keys are generated by the OLT and are  
27 distributed using the *acConfigEncrKey* action (see 14.6.5.1). The OAMPDU carrying the next key is  
28 transmitted within the MLID envelope encrypted using the current key. The OLT shall not transmit the  
29 OAMPDU carrying the *acConfigEncrKey* action over an unencrypted MLID channel.

30 Figure 11-6 illustrates the process of updating the session key (hereinafter referred as “key”) and the  
31 subsequent activation of that key, first by the OLT and then by the ONU.



**Figure 11-6—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.

#### **41.2.3.211.3.3.2 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 *timeout<sub>OLT</sub>* 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.

### 11.2.3.3.11.3.3.3 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 11-6, 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.

### 11.2.3.3.11.3.3.1

#### 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.

#### 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.

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

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

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

## 12 [11.2.4.11.3.4](#) **Session key activation protocol**

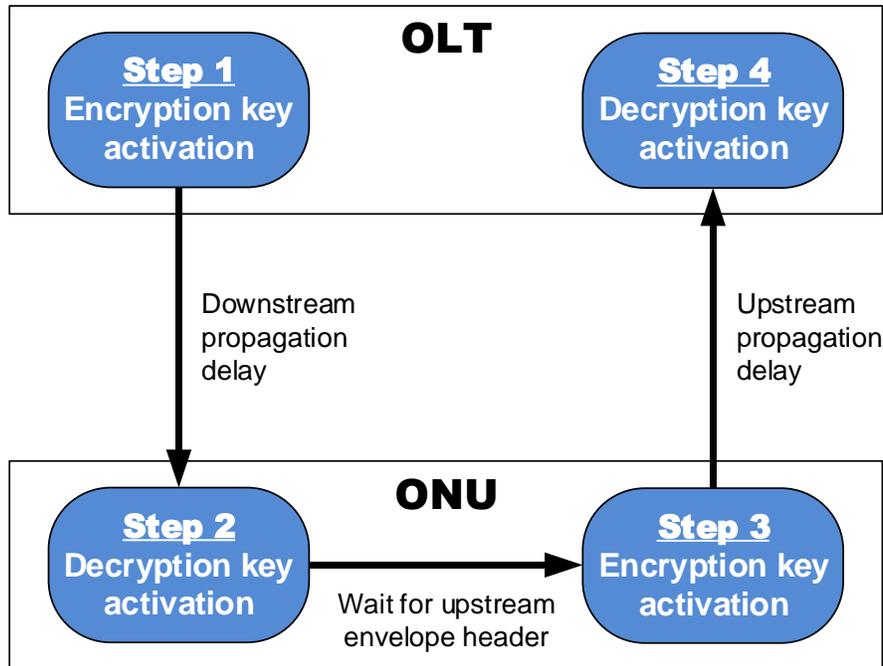
### 13 [11.2.4.11.3.4.1](#) **Protocol overview**

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

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

#### 21 [11.2.4.11.3.4.1.1](#) **Activation of a unicast (bidirectional) key**

22 The unicast (bidirectional) key activation procedure consists of four sequential steps, as illustrated in Figure  
23 11-7.



**Figure 11-7—Four steps comprising the key activation procedure**

Each of the above four steps is represented by an independent process that runs continuously within the secure Multi-Channel Reconciliation Sublayer (MCRS<sub>SEC</sub>).

**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.

**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.

**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

1 of its upstream transmission. The ONU waits for the next envelope header from to the given  
2 encryption entity. The ONU indicates switching to the new key by toggling the value of the  
3 *EncKey* field in the envelope header. The payload following this envelope header is encrypted  
4 using the new key.

#### 5 **Step 4 — Decryption key activation at the OLT:**

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

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

#### 15 **Activation of a multicast (unidirectional) key**

16 ~~11.2.4.4~~ 11.2.4.1 ~~11.3.4.1.3~~ The activation of a multicast (unidirectional) key, i.e., a key associated with a multicast LLIDs, involves  
17 only step 1 and step 2 (see 0) because the multicast LLIDs carry traffic only in the downstream direction.

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

#### 21 ~~11.2.4.4.1~~ 11.3.4.1.3 **Location of key activation processes**

22 The encryption and decryption key activation processes are located within the secure MCRS (MCRS<sub>SEC</sub>)  
23 sublayer, as detailed in 11.3.1.2.

#### 24 ~~11.2.4.2.1~~ 11.3.4.2 **Definition of processes comprising the key activation protocol**

##### 25 **Variables**

26 activeKeyIndex[ee]

27 TYPE: 1-bit integer

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

30 decryptionCounter[ch]

31 TYPE: 128-bit sequence

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

36 decryptionKey[ch]

37 TYPE: sequence of 128 or 256 bits

1           The value of the key currently used for decryption on channel `ch`. The `decryptionKey` value  
2           is fetched for every received envelope header and is passed to the decryption function (see Figure  
3           11-1(b)).

4   `encryptionCounter[ch]`  
5           TYPE: 128-bit sequence

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

10 `encryptionEnabled[ee]`  
11          TYPE: boolean

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

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

19          In the ONU, under normal operation, this variable is equal to `receivedEncrypted[ee]`, i.e.,  
20          an encryption entity `ee` encrypts the outgoing envelopes only if the envelopes this entity receives  
21          from the OLT are also encrypted. For testing and troubleshooting purposes, the NMS may force  
22          the ONU's encryption to be on or off (via the attribute *aEncryptionMode*, see 14.4.5.1)

23 `encryptionKey[ch]`  
24          TYPE: sequence of 128 or 256 bits

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

28 `encryptionMode[ee]`  
29          TYPE: boolean

30          This variable is an alias of the extended attribute *aEncryptionMode* (0xDB/0x04-04) defined in  
31          14.4.5.1.

32 `initialKeyDone[ee]`  
33          TYPE: boolean

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

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

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

1    initialKeyReady[ee]  
2        TYPE: boolean  
3        This variable is used only by the OLT encryption key activation process (see 11.6.2.4), where it  
4        causes the encryption entity ee to start encrypting traffic in the downstream direction.  
5        If the encryption entity ee is associated with an ONU object, this variable is set to true by the  
6        OLT OAM client after the initial (ephemeral) key was calculated and stored in the array keys as  
7        element keys[ee][0]. The derivation of the initial key is described in 11.3. This variable is  
8        reset to false on read.  
9        ~~If~~ If the encryption entity ee is associated with a multicast LLID, this variable is set to true by  
10       the OLT OAM client after the first session key was distributed to all ONUs in a multicast group  
11       and its reception and processing was acknowledged by every ONU. This variable is reset to  
12       false on read.

13    keys[E][2]  
14        TYPE: array of encryption keys  
15        keys[E][2] is a two-dimensional array representing the stored encryption keys. The array size  
16        is  $E \times 2$ , where  $E$  represents the total number encryption entities, with two values stored for each  
17        entity - the currently-active key and the key to be activated next. The value of  $E$  for the OLT  
18        ( $E_{OLT}$ ) is defined in 11.8.1 and its value for the ONU ( $E_{ONU}$ ) is defined in 11.8.2.  
19        The key values are written into the keys[E][2] array by the Security function of the OAM  
20        Client as the end result of the Session Key Distribution Protocol (see 11.5). The Session Key  
21        Activation protocol stat diagrams have a read-only access to this array.

22    keyInterval[ee]  
23        TYPE: integer  
24        This variable represents an interval of time between updating the encryption keys (i.e., a key  
25        lifetime) for encryption entity ee. The value of this variable is provisioned to the OLT by the  
26        NMS, subject to constraints listed in 11.8.3.

27    receivedEncrypted[ee]  
28        TYPE: boolean  
29        This variable indicates whether the received envelope, whose LLID value maps to the encryption  
30        entity ee, is encrypted or not. In the OLT and in the ONU, the value of this variable is derived  
31        from the *EncEnabled* field of the received envelope headers.

32    RxEQ[ch]  
33        TYPE: EQ  
34        This variable represents an envelope quantum (EQ) received and stored in the EnvRx buffer of the  
35        MCRS on channel ch (see IEEE Std 802.3, 143.3.4).

36    TxEQ[ch]  
37        TYPE: EQ  
38        This variable represents an envelope quantum (EQ) being transmitted from the EnvTx buffer of  
39        the MCRS on channel ch (see IEEE Std 802.3, 143.3.3).

1 **Functions**

2 calculateIV(ch, eq)

3 This function calculates the value of the initialization vector (IV) used as an input block to the  
4 AES forward cipher in the AES-CTR (see NIST SP 800-38A, 6.5). The argument ch is an index  
5 of the channel on which the IV is to be used. The argument eq is an EQ that represents an  
6 envelope header. The IV construction method is defined in 11.7.4.2.

7 isHeader(eq)

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

10 mapEncrEntity(llid)

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

15 **Timers**

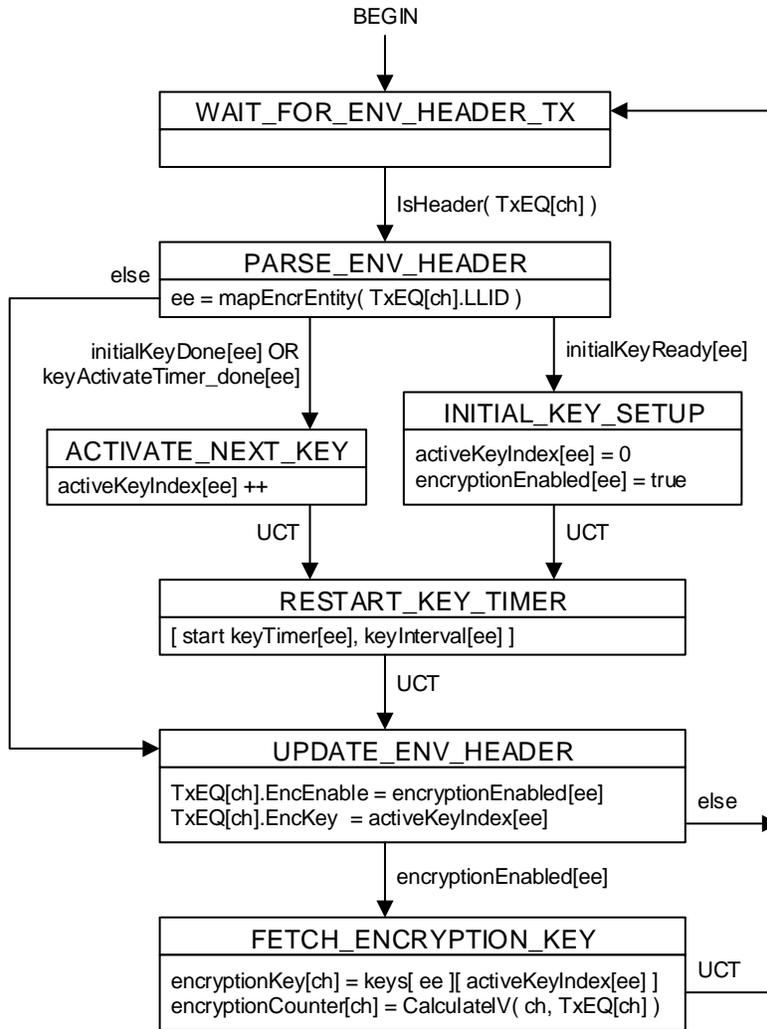
16 ~~11.2.4.2.3~~ keyTimer[ee]

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

20 Each timer instance keyTimer[ee] is associated with an instance of boolean variable  
21 keyTimer\_done[ee]. Upon expiration of the timer, the value of keyTimer\_done[ee]  
22 becomes true (see 3.6.6).

23 11.2.4.2.4 **OLT encryption key activation process state diagram**

24 The OLT shall implement the encryption key activation process as depicted in state diagram in Figure 11-8.  
25 There shall be a separate instance of the encryption key activation process for each transmit channel ch in  
26 the OLT.



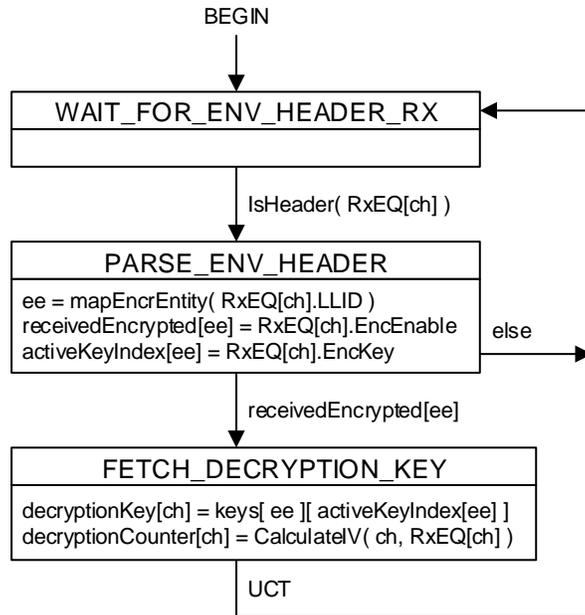
1

2 [11.2.4.2.5](#) [11.3.4.2.5](#) **Figure 11-8—OLT encryption key activation process state diagram**

3

**OLT and ONU decryption key activation process state diagram**

4 The OLT and the ONU shall implement the decryption key activation process as depicted in state diagram  
 5 in Figure 11-9. There shall be a separate instance of the decryption key activation process for each receive  
 6 channel *ch* in the OLT and in the ONU.



1

2

**Figure 11-9—OLT and ONU decryption key activation process state diagram**

3

**ONU encryption key activation process state diagram**

11.2.4.2.6 11.3.4.2.6

4

The ONU shall implement the encryption key activation process as depicted in state diagram in Figure

5

11-10. There shall be a separate instance of the encryption key activation process for each transmit channel

6

ch in the ONU.

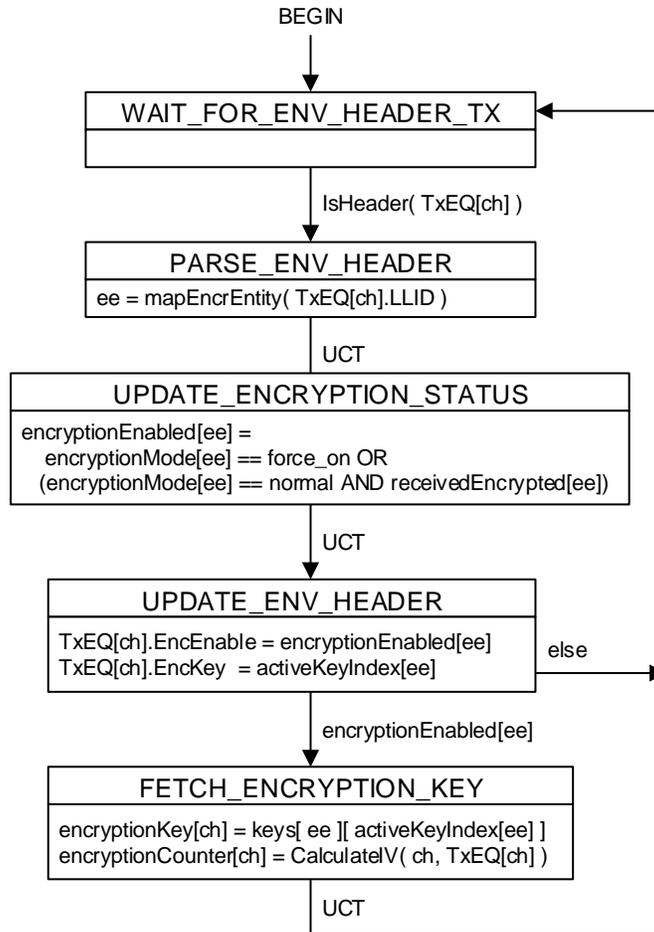


Figure 11-10—ONU encryption key activation process state diagram

### 11.2.5.11.3.5 Cryptographic method

#### 11.2.5.11.3.5.1 Introduction

In SIEPON.4 systems, the OLT and ONUs encrypt data using the AES Counter mode (AES-CTR). The AES-CTR is a confidentiality mode that applies the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are XOR-ed with the plaintext to produce the ciphertext, and vice versa. The AES-CTR mode requires that all counter values be distinct across all of the messages that are encrypted under the given key. For the detailed specification of the AES-CTR refer to NIST SP 800-38A, 6.5.

#### Envelope-based encryption

The concept of transmission envelope is defined in IEEE Std 802.3, 143.2.4.2. An envelope encapsulates continuous transmission by a specific MAC instance (LLID) on one MCRS channel.

In SIEPON.4 cryptographic method, the encryption is based on an envelope structure, i.e., an envelope payload constitutes the plaintext message to be encrypted. The envelope headers themselves are not encrypted. An entire envelope payload is encrypted using the same session key. A new session key may only activate during the reception or transmission of an envelope header (refer to Session Key Activation protocol in 11.6).

1 The cipher block size is 128 bits. Each block of plaintext includes exactly two EQs. Some plaintext blocks  
 2 are only partially-encrypted, i.e., the above-mentioned XOR operation is applied to only a portion of the  
 3 plaintext block. The reason for this is explained in 11.7.2.

4 **Location of the encryption/decryption functional blocks**

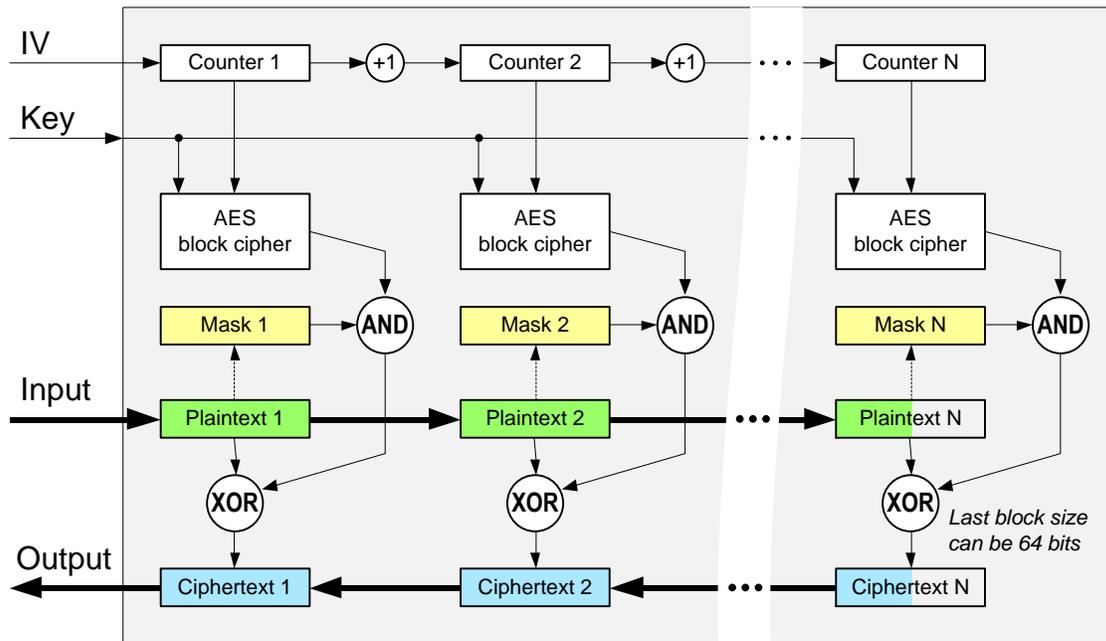
5 The encryption and decryption functional blocks are located within the secure MCRS (MCRS<sub>SEC</sub>) sublayer,  
 6 as detailed in 11.2.2.

7 **Encryption process**

8 The encryption process applies the forward cipher function to each counter block, and the resulting output  
 9 blocks are XOR-ed with the corresponding plaintext blocks to produce the ciphertext blocks (see Figure  
 10 11-11).

11 The first counter block in a message (Counter 1) is initialized to the value called Initialization Vector (IV).  
 12 The IV value used for the encryption is calculated by the OLT encryption key activation process (see  
 13 Figure 11-8) and by the ONU encryption key activation process (see Figure 11-10). Every subsequent  
 14 counter block associated with the given message is constructed by incrementing the value of the previous  
 15 counter block by 1.

16 If the envelope payload length is odd, the last block will only contain one EQ. In such case, the most  
 17 significant 64 bits of the last output block are used for the XOR operation and the remaining 64 least  
 18 significant bits of the last output block are discarded.



19

20 **Figure 11-11—Block diagram of the encryption process**

21 For every block of plaintext, a mask is constructed to block-out the control characters (see 11.7.5.2). This  
 22 mask is AND-ed with the output of the AES block cipher, resulting in the unencrypted control characters  
 23 being placed in the ciphertext blocks.

1 In the encryption process, the forward cipher block operations can be performed in parallel. Moreover, the  
2 forward cipher functions can be applied to the counters prior to the availability of the plaintext data, if the  
3 corresponding counter block values can be determined.

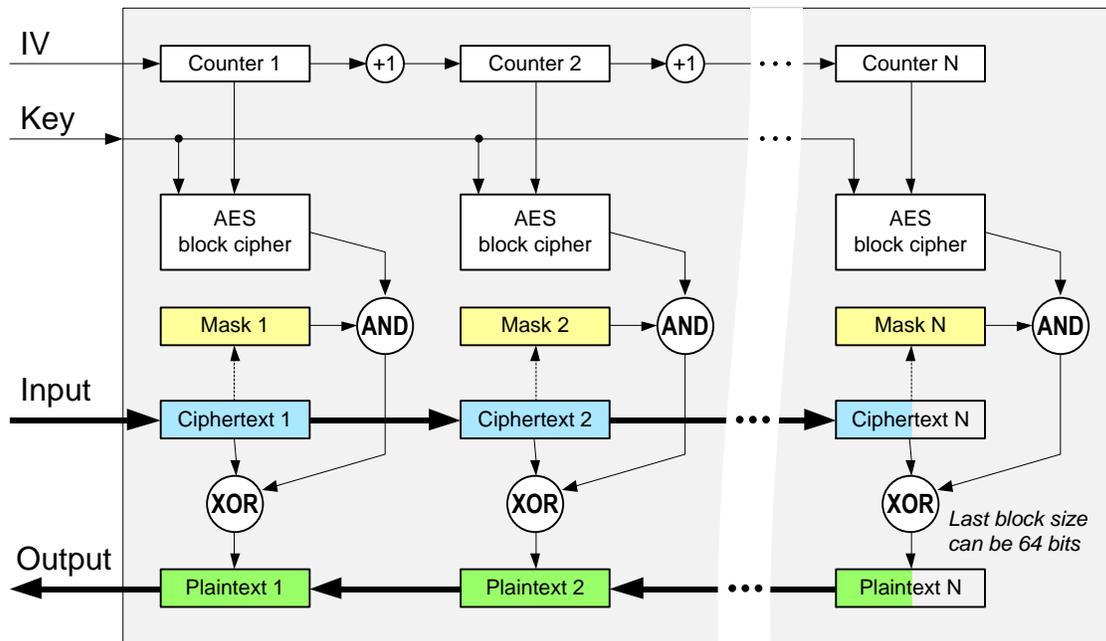
#### 4 11.2.5.3 11.3.5.3 **Decryption process**

5 The decryption process applies the forward cipher function to each counter block, and the resulting output  
6 blocks are XOR-ed with the corresponding ciphertext blocks to recover the plaintext blocks (see Figure  
7 11-12).

8 Similarly to that in the encryption process, the first counter block in a message (Counter 1) is initialized to  
9 the value called Initialization Vector (IV). The IV value used for the decryption is calculated by the OLT  
10 and ONU decryption key activation process (see Figure 11-9). Every subsequent counter block associated  
11 with the given message is constructed by incrementing the value of the previous counter block by 1.

12 For a given encrypted message (i.e., an envelope), the IV and the subsequent counter block values applied  
13 by the decryption process match the IV and the counter values that were previously applied by the  
14 encryption process to encrypt the same message.

15 If the envelope payload length is odd, the last block will only contain one EQ. In such case, the most  
16 significant 64 bits of the last output block are used for the XOR operation and the remaining 64 least  
17 significant bits of the last output block are discarded.



18

19

**Figure 11-12—Block diagram of the decryption process**

20

21

22

For every block of ciphertext, a mask is constructed to block-out the control characters (see 11.7.5.2). This mask is AND-ed with the output of the AES block cipher, resulting in the unencrypted control characters being transferred from the ciphertext blocks into the plaintext blocks.

23

24

25

In the decryption process, the forward cipher block operations can be performed in parallel. Moreover, the forward cipher functions can be applied to the counters prior to the availability of the ciphertext data, if the corresponding counter block values can be determined.

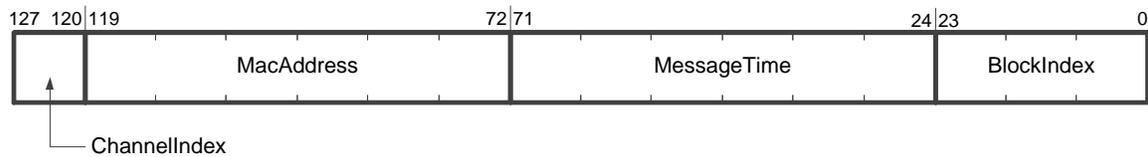
#### 11.2.5.411.3.5.4 Initialization Vector (IV) construction

The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message; across all of the messages that are encrypted under the given key, all of the counters must be distinct. This condition is satisfied by ensuring that IV values calculated for every message are distinct for any message (envelope) encrypted with the same key.

To encrypt a message, the IV is calculated by the encryption key activation processes at the OLT (see Figure 11-8) and at the ONU (see Figure 11-10) when an envelope header (ESH or ECH) is observed in the transmit path of the MCRS<sub>SEC</sub> sublayer.

To decrypt a message, the IV is calculated by the decryption key activation processes at the OLT and the ONU (see Figure 11-9) when an envelope header (ESH or ECH) is observed in the receive path of the MCRS<sub>SEC</sub> sublayer.

The data within the envelope header together with the index of the channel on which this envelope header was transmitted or received comprise the input parameters to the `CalculateIV(...)` function that derives the IV values in the above mentioned processes (see 11.6.2.2). It is critical that for any given encrypted message (envelope), the IV calculated by the decryption key activation process matched the IV calculated by the encryption key activation process.



**Figure 11-13—Structure of the Initialization Vector**

The structure of the IV is illustrated in Figure 11-13. The IV consists of the following four fields:

*ChannelIndex* – Index of the channel on which the encrypted message is being transmitted or received. The most significant bit (bit 127) represents the direction (0 – downstream; 1-upstream), and bits [126:120] represent the channel number. For example, the value 0x01 represents the downstream channel 1 (DC1) and the value 0x80 represents the upstream channel 0 (UC0). The MCRS channels are explained in IEEE Std 802.3, 143.4.1.1.

The inclusion of this field ensures that in a situation when multiple ESHs to/from the same ONU are transmitted at the same time (i.e., the IVs have the same *MessageTime* field value) on different channels, their associated IV values would still be distinct.

*MacAddress* – This is the MAC address of the device that encrypted the given envelope. In the downstream direction, this is the MAC address associated with the PON port of the OLT. In the upstream direction, this is the MAC address associated with the PON port of the transmitting ONU.

To calculate the IV for the decryption, the ONU uses the OLT's MAC address known to it from the MPCP registration step. The OLT also has a prior knowledge of MAC addresses of all connected ONUs, but it needs to determine which specific ONU sourced the given envelope. It does that by first extracting the LLID value from the ESH and then looking up the MAC address associated with this LLID.

*MessageTime* – This field represents a timestamp of *cipher clock* captured at the moment when the encryption key activation process observes the ESH in the MCRS<sub>SEC</sub> transmit path or

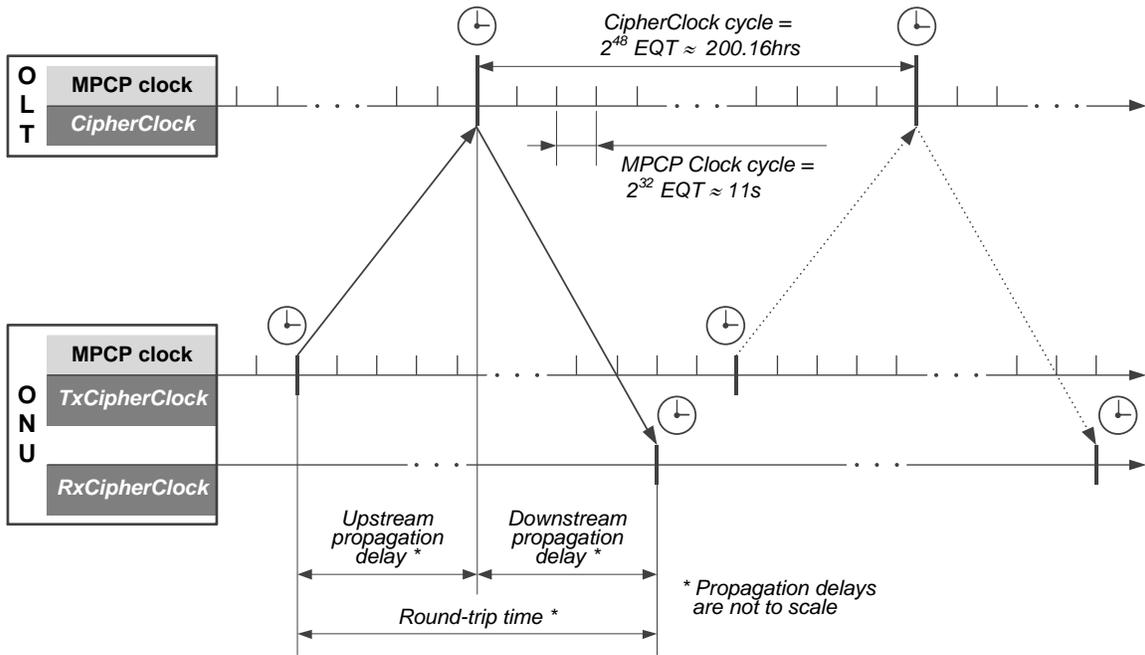
1 the decryption key activation process observes the ESH in the  $MCRS_{SEC}$  receive path.  
 2 The cipher clock runs synchronously with the MPCP clock, but otherwise is a distinct  
 3 clock, as explained in 11.7.4.1.

4 This field ensures that all the counter block values used on the same channel (i.e., the  
 5 IVs have the same *ChannelIndex* values) and with the same session key are distinct.

6 *BlockIndex* – The block index field is set to zero when an envelope header is detected. This field is  
 7 incremented by 1 for every subsequent counter block in the same envelope.

8 **Cipher clock**

9 The cipher clock is a 48-bit counter that runs synchronously with the MPCP clock (*LocalTime*), but is a  
 10 distinct clock. The OLT and the ONUs contain versions of this clock that is used as a timestamp source for  
 11 the IV field *MessageTime*. At the OLT, a single clock, referred to as *CipherClock* is used for IV  
 12 construction in both the encryption and the decryption functions. At the ONU, there are two instances of  
 13 the cipher clock: the *TxCipherClock* that is used to construct the IV for the encryption function, and the  
 14 *RxCipherClock* that is used to construct the IV for the decryption function. The relationship between the  
 15 OLT's *CipherClock* and the ONU's *RxCipherClock* and *TxCipherClock* is illustrated in Figure 11-14.



16

17 **Figure 11-14—Relationship of OLT's *CipherClock* and**  
 18 **ONU's *TxCipherClock* and *RxCipherClock***

19 The MPCP clock is a 32-bit counter that increments by one every EQT. The initial synchronization of the  
 20 MPCP clock takes place during ONU's MPCP discovery and registration and is described in  
 21 IEEE Std 802.3, 144.3.1.1.

22 The origin point of the MPCP clock (counter) at the ONU is advanced relative to the origin point of the  
 23 MPCP clock at the OLT by the upstream propagation time. The desired effect of such shift is that an  
 24 envelope header transmitted by the ONU at its local MPCP time  $T_i$  is received by the OLT also at its local  
 25 MPCP time  $T_i$ .

1 The *CipherClock* in the OLT is an extension of the OLT MPCP clock constructed by prepending 16 most-  
2 significant bits to the MPCP clock, i.e., `LocalTime` counter (see IEEE Std 802.3, 144.2.1.1). The carry-  
3 over bit from the `LocalTime` counter increments the first bit of the 16-bit extension portion (i.e., the bit  
4 32 of the 48-bit *CipherClock* counter).

5 The *TxCipherClock* in the ONU is an extension of the ONU MPCP clock and is constructed in a manner  
6 similar to the OLT *CipherClock* construction. Because the OLT *CipherClock* and the ONU *TxCipherClock*  
7 extend their respective MPCP clocks, they preserve their relative shift, ensuring that the *MessageTime*  
8 value used to construct the IV for the encryption at the ONU matches the *MessageTime* value used to  
9 construct the IV for the decryption at the OLT.

10 The *RxCipherClock* at the ONU is a separate 48-bit clock increments synchronously with the ONU MPCP  
11 clock, but is not an extension of the MPCP clock (i.e., the low 32 bits of the *RxCipherClock* are not equal to  
12 ONU's MPCP `LocalTime` value). The origin point of the *RxCipherClock* at the ONU is delayed relative  
13 to the origin point of the *CipherClock* at the OLT by the downstream propagation time. The desired effect  
14 of such shift is that an envelope header transmitted by the OLT when its *CipherClock* value is  $T_i$  is received  
15 by the ONU at its *RxCipherClock* value is also  $T_i$ .

#### 16 [11.2.5.4.1.1](#)[11.3.5.4.1.1](#) Cipher clock alignment in the upstream

17 The MCRS defined in IEEE Std 802.3, Clause 143 ensures that an envelope header (EH) transmitted by the  
18 ONU at a specific local time value is received at that exact local time at the OLT. To achieve that, the ONU  
19 sets the EPAM field in the EH to equal 6 least significant bits of its MPCP time ( $\text{EH.EPAM} =$   
20  $\text{LocalTime}[5:0]$ ).

21 At the OLT, this EH is received (i.e., is written) into the `EnvRx` buffer into row with index equal to  
22 `EH.EPAM`. This EH is then read from the `EnvRx` buffer at the exact time when the OLT's  
23 `LocalTime[5:0]` are equal to the row index (i.e., when  $\text{LocalTime}[5:0] = \text{EH.EPAM}$ ). As the 6  
24 LSB are aligned, so are the entire extended MPCP clock values at the ONU and OLT are equal.

25 Since the *CipherClock* at the OLT and the *TxCipherClock* at the ONU are the extensions of their respective  
26 MPCP clocks, it follows that the value of ONU's *TxCipherClock* latched at the moment when the EH is  
27 written into `EnvTx` buffer at the ONU matches the value of OLT's *CipherClock* latched at the moment  
28 when the EH is read from the `EnvRx` buffer at the OLT.

#### 29 [11.2.5.4.1.2](#)[11.3.5.4.1.2](#) Cipher clock alignment in the downstream

30 In the downstream direction, the OLT sets the EPAM field in the EH to equal 6 least significant bits of its  
31 MPCP time ( $\text{EH.EPAM} = \text{LocalTime}[5:0]$ ).

32 At the ONU, this EH is received (i.e., is written) into the `EnvRx` buffer into row with index equal to  
33 `EH.EPAM`. This EH is then read from the `EnvRx` buffer at the exact time when the 6 LSB of the ONU's  
34 *RxCipherClock* are equal to the row index. (Note however, that  $\text{ONU's LocalTime}[5:0] \neq \text{EH.EPAM}$   
35 because the ONU's MPCP clock is advanced by the upstream propagation time, see Figure 11-14.)

36 As the 6 LSB are aligned, it follows that the value of OLT's *CipherClock* latched at the moment when the  
37 EH was written into `EnvTx` buffer at the OLT matches the value of ONU's *RxCipherClock* latched at the  
38 moment when the EH was read from the `EnvRx` buffer at the ONU.

#### 39 [11.2.5.4.1.3](#)[11.3.5.4.1.3](#) Initial cipher clock synchronization

40 While the OLT and ONU MPCP clocks are synchronized as part of the MPCP Discovery and Registration  
41 process (see IEEE Std 802.3, 144.3.1.1), the *RxCipherClock* and the *TxCipherClock* require an additional  
42 synchronization procedure to synchronize the 16 most-significant bits.

1 To initiate the cipher clock synchronization at the ONU, the OLT issues a *Set\_Request* OAMPDU  
2 containing the *Sync Cipher Clock* TLV (see 14.6.5.2). The OLT shall not activate the initial encryption key  
3 for an ONU until it receives a positive acknowledgement from that ONU that the *Sync Cipher Clock* TLV  
4 was processed successfully.

5 The OLT shall form the *Sync Cipher Clock* TLV by setting its *RxCipherTimestamp* and  
6 *TxCipherTimestamp* fields as follows:

```
7     RxCipherTimestamp = CipherClock;  
8     TxCipherTimestamp = RxCipherTimestamp + RTT[PLID];
```

9 The  $RTT[PLID]$  is the round-trip time value measured for the given ONU (PLID) at the time of its MPCP  
10 discovery. Note that adding  $RTT[PLID]$  may cause the value *TxCipherTimestamp* to wrap around.

11 It may not be possible to tightly control the transmission time of OAMPDUs, unlike that of MPCPDUs. It  
12 is acceptable for the OAMPDU containing the *Sync Cipher Clock* TLV to be transmitted after the time  
13 epoch corresponding to the captured value of the OLT's *CipherClock*, but the transmit time (referenced to  
14 the ESH of the envelope containing this OAMPDU) shall not lag behind the said time epoch by more than  
15 1 second.

16 When the ONU receives the *Sync Cipher Clock* TLV, it increments both the *TxCipherTimestamp* and  
17 the *RxCipherTimestamp* until the bottom 32 bits of the *TxCipherTimestamp* match the current  
18 value of its local MPCP clock (i.e., *LocalTime* variable). The amount of such increment depends on how  
19 much the transmission time of *Sync Cipher Clock* TLV lagged behind the captured value of OLT's  
20 *CipherClock*. Note that incrementing the *TxCipherTimestamp* or the *RxCipherTimestamp* may  
21 cause the values to wrap around.

```
22 while( TxCipherTimestamp[31:0] != LocalTime)  
23 {  
24     TxCipherTimestamp ++;  
25     RxCipherTimestamp ++;  
26 }
```

27 Once the bottom 32 bits of *TxCipherTimestamp* match the *LocalTime*, the values of *TxCipherClock*  
28 and *RxCipherClock* are set by writing the corresponding adjusted timestamps into the *acSyncCipherClock*  
29 attribute (see 14.6.5.2):

```
30 acSyncCipherClock.sTxCipherClock = TxCipherTimestamp;  
31 acSyncCipherClock.sRxCipherClock = RxCipherTimestamp;
```

32 From this moment on, the *TxCipherClock* and the *RxCipherClock* increment synchronously with the  
33 ONU's MPCP clock.

34 If the *TxCipherClock* and the *RxCipherClock* are synchronized properly, the following holds true:

- 35 — The bottom 32 bits of *TxCipherClock* match the local MPCP time at all times  
36 ( $TxCipherClock[31:0] == LocalTime$ ).
- 37 — The bottom 6 bits of *RxCipherClock* match the value of the EPAM field in any received  
38 envelope header.

39 Implementations may choose to verify the above conditions in order to ensure proper *RxCipherClock* and  
40 *TxCipherClock* alignment.

1 **11.2.5.4.1.4 11.3.5.4.1.4 Implementation options (informative)**

2 At the OLT, the *CipherClock* and MPCP clock can share the same 48-bit variable (register), with the  
3 MPCP clock occupying the 32 least-significant bits (i.e., `LocalTime[31:0] =`  
4 `CipherClock[31:0]`).

5 At the ONU, an observation can be made that the time frame reference of *RxCipherClock* lags behind the  
6 time frame reference of the *TxCipherClock* by a fixed interval equal to ONU's round trip time. Thus, it is  
7 possible to represent the MPCP clock, the *TxCipherClock* and the *RxCipherClock* by the same 48-bit  
8 variable (register). The *TxCipherClock* is represented by the full register value, while the ONU MPCP  
9 clock is represented by the 32 least-significant bits (i.e., `LocalTime[31:0] =`  
10 `TxCipherClock[31:0]`). The *RxCipherClock* can be derived by subtracting the round-trip time (a  
11 fixed constant) from the value of the *TxCipherClock*: `RxCipherClock[47:0] =`  
12 `TxCipherClock[47:0] - RTT`.

13 **CalculateIV(...) function**

14 The function `CalculateIV(ch, eh)` is used by the encryption and decryption key activation  
15 processes in the OLT and ONUs. In each of these processes, the behavior of this function is similar at the  
16 high level, but differs in specific minor details, as explained below. This function executes within one  
17 MPCP clock cycle (in under one EQT), therefore the 6 least-significant bits of the relevant cipher clock  
18 counter match the value of the EPAM field of the EH (see IEEE Std 802.3, 143.3.2).

19 In the OLT encryption key activation process, the function `CalculateIV(ch, eh)` is called at the  
20 moment when an envelope header (EH) `eh` is observed in the MCRS transmit path on channel `ch` (see  
21 Figure 11-8). The following is the definition of the function `CalculateIV(ch, eh)` for the OLT  
22 encryption key activation process:

```
23 int128 CalculateIV( ch, eh )  
24 {  
25     iv.ChannelIndex = ch;           // Channel index  
26     iv.MacAddress   = OLT_MAC_ADDRESS; // Known constant  
27     iv.MessageTime  = CipherClock;   // Latch OLT's cipher clock  
28     iv.BlockIndex   = 0;             // Reset block index  
29     return iv;  
30 }
```

31 In the OLT decryption key activation process, the function `CalculateIV(ch, eh)` is called at the  
32 moment when an envelope header (EH) `eh` is observed in the MCRS receive path on channel `ch` (see  
33 Figure 11-9). The following is the definition of the function `CalculateIV(ch, eh)` for the OLT  
34 decryption key activation process:

```
35 int128 CalculateIV( ch, eh )  
36 {  
37     iv.ChannelIndex = ch;           // Channel index  
38     iv.MacAddress   = MacAddr[eh.llid]; // MAC address table lookup  
39     iv.MessageTime  = CipherClock;   // Latch OLT's cipher clock  
40     iv.BlockIndex   = 0;             // Reset block index  
41     return iv;  
42 }
```

43 Note that, in this function, the OLT needs to perform a table lookup to retrieve the MAC address associated  
44 with a given LLID value.

1 In the ONU encryption key activation process, the function CalculateIV(ch, eh) is called at the  
2 moment when an envelope header (EH) eh is observed in the MCRS transmit path on channel ch (see  
3 Figure 11-10). The following is the definition of the function CalculateIV(ch, eh) for the ONU  
4 encryption key activation process:

```
5 int128 CalculateIV( ch, eh )
6 {
7     iv.ChannelIndex = ch;           // Channel index
8     iv.MacAddress   = ONU_MAC_ADDRESS; // Known constant
9     iv.MessageTime  = TxCipherClock; // Latch ONU's tx. cipher clock
10    iv.BlockIndex   = 0;           // Reset block counter
11    return iv;
12 }
```

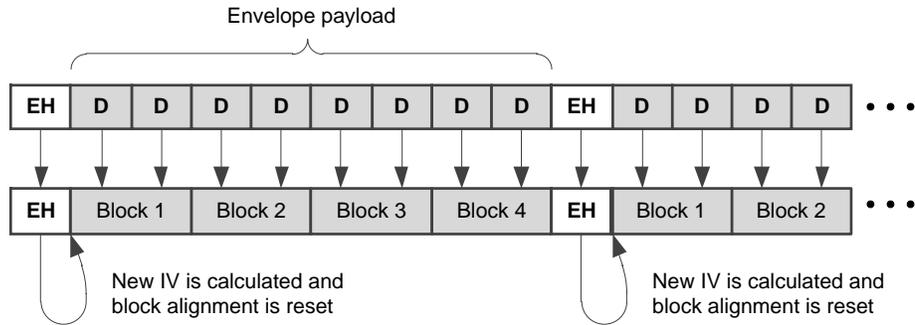
13 In the ONU decryption key activation process, the function CalculateIV(ch, eh) is called at the  
14 moment when an envelope header (EH) eh is observed in the MCRS receive path on channel ch (see  
15 Figure 11-9). The following is the definition of the function CalculateIV(ch, eh) for the ONU  
16 decryption key activation process:

```
17 int128 CalculateIV( ch, eh )
18 {
19     iv.ChannelIndex = ch;           // Channel index
20     iv.MacAddress   = OLT_MAC_ADDRESS; // Learned at registration
21     iv.MessageTime  = RxCipherClock; // Latch ONU's rx. cipher clock
22     iv.BlockIndex   = 0;           // Reset block index
23     return iv;
24 }
```

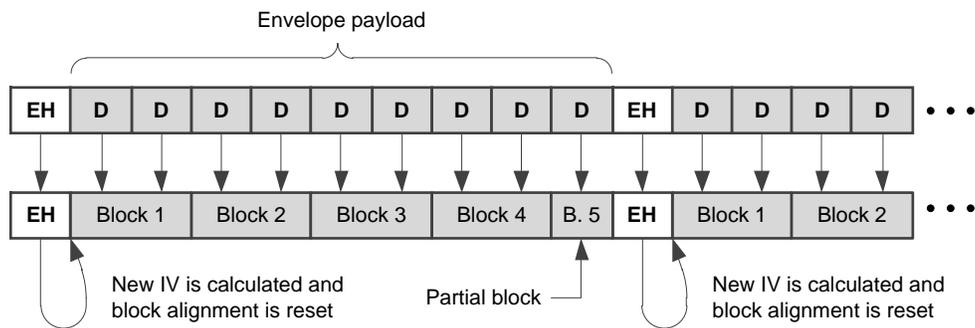
## 25 **41.2.5-511.3.5.5 Encrypted envelope format**

26 To encrypt a message, an envelope payload is divided in 128-bit blocks of plaintext and the encryption  
27 operation is performed as described in 11.7.2. To decrypt a message, an envelope payload is divided in  
28 128-bit blocks of ciphertext and the decryption operation is performed as described in 11.7.3.

29 Exactly two EQs form a plaintext or ciphertext block, except in the case of odd payload length, the last  
30 block contains a single EQ (see Figure 11-15). In every envelope, the first payload block is aligned to the  
31 end of envelope header. The envelope header itself is not encrypted.



a) Block alignment in an envelope with the payload of even length



b) Block alignment in an envelope with the payload of odd length



1

2 [11.2.5.5-11.3.5.1](#)

**Figure 11-15—EQ-to-block conversion**

3

**EQ types that bypass the encryption and decryption processes**

4

As was shown in 11.2.3 and 11.2.4, there are several types of EQs that can appear in the MCRS data path. Some of the EQ types represent control sequences used to signal frame, envelope, or burst delineation (see IEEE Std 802.3, 143.3.3.6.2). These EQs require special treatment by the encryption and the decryption functions, as illustrated in Figure 11-16 and detailed below:

8

**Rate Adjustment (RATE\_ADJUST\_EQ):**

9

The MCRS (MCRS<sub>SEC</sub>) periodically inserts a series of 33 RATE\_ADJUST\_EQs to pace the MAC data rate in order to allow the FEC parity data insertion by the PCS. The position of RATE\_ADJUST\_EQ insertion is determined by the Input process of the MCRS Transmit function and by the Output process of the MCRS Receive function. The RATE\_ADJUST\_EQ insertion by the Input and the Output processes may happen at different positions within an envelope.

14

The RATE\_ADJUST\_EQs are not considered part of envelope (i.e., they are not accounted in envelope length value). As described in 11.2.3 and 11.2.4, these EQs bypass the Encryption/Decryption processes, i.e., they are not encrypted and they do not affect the plaintext

15

16

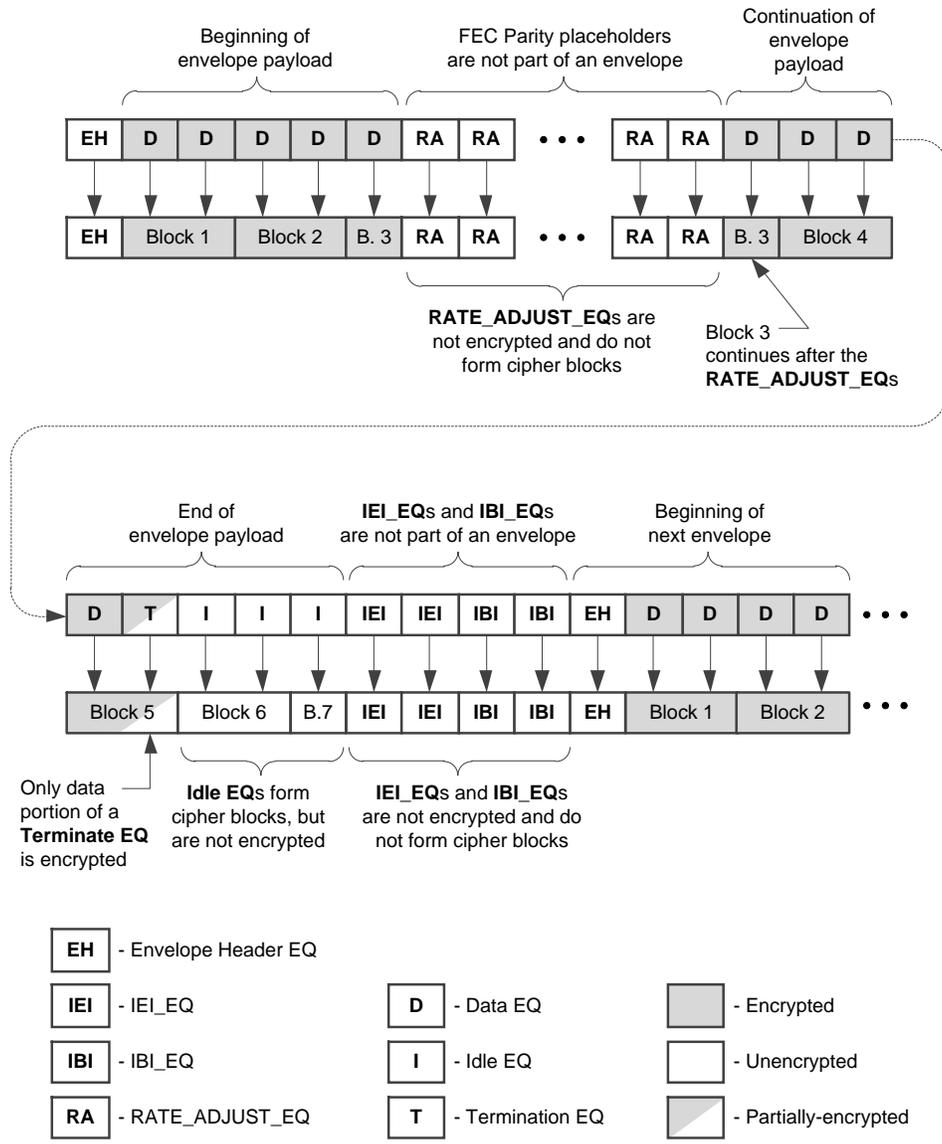
1 or the ciphertext block alignment. The sequence of RATE\_ADJUST\_EQs may be inserted in the  
 2 middle of single plaintext or ciphertext block, as illustrated in Figure 11-16.

3 **Inter-Envelope Idle (IEI\_EQ):**

4 The IEI\_EQs are inserted when there is no envelope available for transmission, while the  
 5 transmission channel itself is active. The IEI\_EQs are not part of an envelope. However, unlike the  
 6 RATE\_ADJUST\_EQs, they cannot appear in the middle of an envelope. Within the encryption  
 7 and decryption functions, the IEI\_EQs bypass the Encryption/Decryption processes, i.e., they are  
 8 not encrypted and they do not affect the plaintext or the ciphertext block alignment.

9 **Inter-Burst Idle (IBI\_EQ):**

10 The IBI\_EQs are inserted when the transmission channel is not active, such as between  
 11 transmission bursts. The IBI\_EQs can only appear in the upstream and are not considered part of  
 12 an envelope. Within the encryption and decryption functions, the /IBI/ characters bypass the  
 13 Encryption/Decryption processes, i.e., they are not encrypted and they do not affect the plaintext  
 14 or the ciphertext block alignment.



15

**Figure 11-16—Handling of special EQ types by encryption/decryption function**

**Handling of the control characters in envelope payloads**

There are several EQ types that can appear in the payload portion of an envelope: data EQ, Termination EQ, and (regular) Idle EQ. Some of these EQ types may include control characters /T/ and /I/. In order to support 64b/66b encoding in the PCS, these control characters are passed from the input to the output of the encryption or the decryption process unmodified.

As explained in 11.7.2 and 11.7.3, the control characters are left unencrypted by applying a mask to the output of the AES block cipher, before that output is XOR-ed with the plaintext or the ciphertext blocks.

Within the MCRS, an EQ is represented by a 72-bit structure, consisting of 8 control bits `Ctrl[0:7]` and 8 data octets `Data [0:7]` (see IEEE Std 802.3, 143.2.4.1). If the `Ctrl[i]` bit is 1, then the corresponding `Data[i]` octet represents a control character, which shall be left unencrypted. Otherwise, the `Data[i]` is a data octet, which shall be encrypted. Table 11-1 shows all EQ types that may be encountered in the envelope payload and the associated EQ mask. The masks associated with two EQs that form a plaintext or a ciphertext block are combined to form a 128-bit mask that is to be applied to the output of the AES block cipher.

**Table 11-1—EQ types and the associated encryption EQ masks**

EQ type	Ctrl[0:7] (bin)	Data[0:7] (hex) <sup>a</sup>	Mask (hex)
Data	00000000	xx-xx-xx-xx-xx-xx-xx-xx	FF-FF-FF-FF-FF-FF-FF-FF
Terminate	00000001	xx-xx-xx-xx-xx-xx-xx-FD	FF-FF-FF-FF-FF-FF-FF-00
	00000011	xx-xx-xx-xx-xx-xx-FD-07	FF-FF-FF-FF-FF-FF-00-00
	00000111	xx-xx-xx-xx-xx-FD-07-07	FF-FF-FF-FF-FF-00-00-00
	00001111	xx-xx-xx-xx-FD-07-07-07	FF-FF-FF-FF-00-00-00-00
	00011111	xx-xx-xx-FD-07-07-07-07	FF-FF-FF-00-00-00-00-00
	00111111	xx-xx-FD-07-07-07-07-07	FF-FF-00-00-00-00-00-00
	01111111	xx-FD-07-07-07-07-07-07	FF-00-00-00-00-00-00-00
	11111111	FD-07-07-07-07-07-07-07	00-00-00-00-00-00-00-00
Idle	11111111	07-07-07-07-07-07-07-07	00-00-00-00-00-00-00-00

<sup>a</sup> xx indicates ‘any value’

**11.3.6 Encryption key management**

**11.3.6.1 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 *U* ONUs and *M* multicast LLIDs, the OLT shall be able to support *E<sub>OLT</sub>* encryption entities, where  $E_{OLT} = U + M$  (refer to use of constant *E<sub>OLT</sub>* 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 *U* keys are active for decryption and *U + M* keys are active for encryption.

1 **11.2.6-211.3.6.2 Key storage in the ONU**

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

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

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

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

12 **11.2.6-311.3.6.3 Key lifetime**

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

16 The IV construction is defined in 11.7.4. It relies on the extended 48-bit MPCP clock counter. This counter  
17 increments every envelope quantum time (EQT), which equals 2.56 ns (see IEEE Std 802.3, 1.4.245c).  
18 Thus, the MPCP counter rolls-over every 200.16 hours. The OLT shall maintain the key lifetime of less  
19 than or equal to 200 hours ( $T_{max\_key\_lifetime}$ ). Different encryption entities may optionally have different key  
20 lifetimes not exceeding the  $T_{max\_key\_lifetime}$ .

21 **11.2.7-11.3.7 Encryption testing and troubleshooting modes of operation**

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

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

27 **11.2.7-411.3.7.1 Encryption entities associated with multicast LLID objects**

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

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

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

1 **11.2.7.211.3.7.2 Encryption entities associated with ONU objects**

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

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

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

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

14 At the ONU, the upstream encryption is controlled via the attribute *aEncryptionMode* (see 14.4.5.1). Under  
15 the normal operation, the upstream encryption is enabled if the downstream unicast traffic to this ONU is  
16 encrypted. For testing and troubleshooting purposes, the *aEncryptionMode* allows forcing the upstream  
17 encryption on or off, regardless of the status of the downstream encryption.

18 Disabling the upstream encryption is a less disruptive operation than disabling the downstream encryption,  
19 since the encryption keys are never transmitted in the upstream direction. Upon conclusion of the  
20 troubleshooting analysis, the upstream encryption can be re-enabled by setting the *aEncryptionMode*  
21 attribute to `normal`.

22 **11.2.7.311.3.7.3 Encryption entities associated with multicast MLID objects**

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

26