|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SEC 30 |
| Source:\* | François Ennesser, Gemalto (ETSI), francois.ennesser@gemalto.com |
| Date:\* | 2017-06-22 |
| Reason for Change/s:\* | Modification of TS-0003 Annex L to allow greater flexibility in physical and logical implementations, while preserving interoperability at the API level thanks to TS-0016. |
| CR against: Release\* | 3 |
| CR against: WI\* | Active <WI-0067 PKI SE Framework>  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number: (Note to Rapporteur - use latest agreed revision)  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TS-0003 v3.3.1 |
| Clauses \* |  |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only ONE of the above shall be ticked |
| Impacted other TS/TR(s) |  |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES  NO  This CR may break backwards compatibility with the last approved version of the TS? YES  NO |
| Template Version: January 2017 (Do not modify) | |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

GUIDELINES for Change Requests:

Provide an informative introduction containing the problem(s) being solved, and a summary list of proposals.

Each CR should contain changes related to only one particular issue/problem.

In case of a correction, and the change apply to previous releases, a separate “mirror CR” should be posted at the same time of this CR

Mirror CR: applies only when the text, including clause numbering are exactly the same.

Companion CR: applies when the change means the same but the baselines differ in some way (e.g. clause number).

Follow the principle of completeness, where all changes related to the issue or problem within a deliverable are simultaneously proposed to be made E.g. A change impacting 5 tables should not only include a proposal to change only 3 tables. Includes any changes to references, definitions, and acronyms in the same deliverable.

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar to the extent practicable.

Use Change bars for modifications.

The change should include the current and surrounding clauses to clearly show where a change is located and to provide technical context of the proposed change. Additions of complete clauses need not show surrounding clauses as long as the proposed clause number clearly shows where the new clause is proposed to be located.

Multiple changes in a single CR shall be clearly separated by horizontal lines with embedded text such as, start of change 1, end of change 1, start of new clause, end of new clause.

When subsequent changes are made to content of a CR, then the accepted version should not show changes over changes. The accepted version of the CR should only show changes relative to the baseline approved text.

## Introduction

The present CR modifies the CR for Annex L of TS-0003 in SEC-2017-0098R01 as follows:

* .include TLS cipher suites used in TS-0003
* Include features required by TS-0016

### -----------------------Start of change 1-------------------------------------------

Annex L (normative):  
Tamper-resistant secure element framework supporting asymmetric cryptography Services

# L.0 Introduction

### L.0.1 Overview

Secure elements may be integrated in PKI systems to provide secure identification and authentication of devices, tamper-resistant storage areas for sensitive data (especially secure storage of private keys which may be generated on board in the SE and always used within it) managed by defined stakeholders, and digital signature services with management of digital certificates. Secure element supporting asymmetric cryptographic services are termed Asymmetric Secure Element (ASE) in the rest of the present annex, which specifies features that should be exposed by the ASE to its hosting device to enable interoperable application deployments:

* Providing keys (which may be randomly generated data) to the hosting device for encrypting, integrity protecting and authenticating data sent by the hosting device to receiver of the data.
* Negotiation of keys for protecting the communication between hosting device and ASE.
* Calculating signatures for data to provide non repudiation
* Generation of random numbers for the TLS command ClientHello
* Key negotiation of the TLS pre-master secrets
* Signature generation and verification for the TLS authentication.
* Providing generic cryptographic services to Application Entities

The ASE may be a UICC [24] or eUICC [eUICC], in which case the framework proposed in the present annex may coexist with some features specified in Annex D, e.g. by being implemented as a GlobalPlatform applet loaded on a UICC. However, an ASE does not need to be UICC or eUICC compliant in the context of the present annex. The ASE capabilities specified in the present annex may be implemented as a secure element applet as per GlobalPlatform Card Specifications [63], which first needs to be selected in order for the ASE to exhibit the specified behaviour. This implementation provides the possibility to install and provision the asymmetric cryptographic capabilities on secure elements, even after deployment on the field, in a standard manner. It also enables to leverage on the Security Domains structure (SD) of the GlobalPlatform Card specification [63], allowing multiple stakeholders to independently operate and manage their own secure environments on a single secure element.

### L.0.2 Naming Conventions

To easily identify whether a key is public or private, whether it exists in the ASE or the hosting device or is a CA key, and also the usage of a key, the following notation is used in this annex:

**KeyType.KeyOwner.KeyUsage**

To easily identify whether a certificate can be verified in the ASE or not, whether it exists in the ASE or the hosting device or belongs to a CA or root CA, and also its usage, the following notation is used in this annex:

**CertType.CertOwner.CertUsage**

The possible values are shown in the following table:

Table L.0-1: Naming convention

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Meaning |
| Key or certificate Type | PuK | Public Key |
| PrK | Private key |
| Owner | ICC | ASE |
| IFD | Hosting device (i.e. interface with M2M application) |
| CA | Certification Authority |
| CAICC | Certification authority that generated the certificate for the ICC public key |
| RCA | Root Certification Authority |
| Usage | AUT | Authentication key |
| DS | Digital signature key |
| KA | Key Agreement |
| CS-AUT | Certificate Signature Authentication |

# L.1 Physical interface and transport protocol

The intention of the present annex is to specify a set of generic security services that shall be supported in oneM2M ASE and should be exposed to oneM2M applications through the Secure Environment Abstraction Layer of TS-0016 [TS16].

The ASE services are described at a high level in order to support implementations that comply with specific regulations, e.g. regional standards such as EN 419 212 [64] in the European Union or FIPS 201-2 [FIPS201] in the USA, or vertical such as BSI TR 03109 [3109] in the German energy sector. The ASE security services described in this annex are commonly supported in secure elements used for certificate-based security deployments, such as governmental or corporate identification cards supporting digital signature as per EN 419 212 [64] or FIPS 201-2 [FIPS 201].

The functionalities described in the present annex imply the presence of a random number generation capability in the ASE. This functionality may be made available to the hosting device. They also imply that the ASE supports asymmetric cryptography based on the RSA or ECC algorithms, and the AES symmetric algorithm.

The Secure Element may interface with the hosting M2M device through various physical communication means., The difference between the multiple communication links (wired or contactless)that may be useddoes not otherwise impact the way applications would interact with the Secure Element.

# L.2 Lifecycle phases

The ASE lifecycle comprises the following phases:

* Personalization, where the ASE maintains the state initialized upon creation to enable its initial provisioning. This phase is supposed to take place in a trusted facility under control of the stakeholder responsible for the ASE (e.g. ASE issuer facility, device assembly line or Point of sale). It ends when the ASE receives a trigger to transition into its operational state.
* Operational phase, where the ASE maintains a state suitable for secure operation in the field, into which a transition is triggered upon completion of the personalization phase.

A secure channel shall first be established to secure data exchange with a host, as described in clause L.3. Depending on the operating environment, the secure channel may only ensure mutual authentication between both entities, or add MIC protection, or add both MIC and encryption.

In case the ASE functions are implemented as a GlobalPlatform applet, the security of the applet during the installation phase is ensured by the Card Manager, which computes the session keys and opens a secure channel with the personalization device to protect loading and installation. Once loaded, the applet becomes selectable, which enables its provisioning.

Operation of the ASE (or ASE applet) during its personalization phase can be subject to specific constraints and can include special commands that are not available in the operational statet. For example, the GlobalPlatform Card Specification [63] specifies low level personalization commands and procedures that may be implemented by ASE supporting ISO 7816-4 APDUs [26] in deployments requiring interoperability in the personalization state.

At the end of initial provisioning/personalization, the ASE (or ASE applet) enters an operational state, in which the functions specified in clause L.4 shall be available.

During operation, the secure element or specific information within it (e.g. keys) may move to a “blocked” state designed as a protection mechanism once it encounters any integrity problem or e.g. if a maximum allowed number of authentication attempts has been reached. .

# L.3 Device Application / ASE Authentication and Secure Channel Establishment

To prevent execution of commands and access to information by unauthorized entities, communication between the hosting device application and ASE shall rely on the establishment of a secure channel, based on mutual authentication of the communicating entities, both in the personalization and the operational state. This enables encryption of the information exchanged over the Mcs and Mca reference points. This bilateral mechanism ensures that:

* On one side, any entity (such as a clerk) which wants to access the protected data on the ASE, shall authenticate themselves to the ASE. Behind the entity are the system and the hosting device (called IFD). The ASE checks that the entity who is requiring access to the data is allowed to do so.
* On the other side, the ASE authenticates itself to the clerks systems via the IFD, to ensure that it is genuine.

After mutual authentication between an entity and the ASE, the ASE grants the specific access rights related to the entity.

The secure channel authentication required for the ASE and external entity to exchange sensitive information may be based on either symmetric or asymmetric credentials:

* Asymmetric key mutual authentication based on the ASE and IFD verifying the existence of a certified key pair in the other entity. This process can be based on either RSA device authentication, or ECC device authentication. Where needed, common symmetric session keys can then be derived using the Diffie–Hellman key exchange mechanism to ensure integrity and/or confidentiality of the information exchange.
* Symmetric key mutual authentication based on the ASE and IFD verifying the existence of two AES symmetric secret keys, KENC and KMIC, in the other entity. A successful symmetric mutual authentication opens the secure channel.

Establishment of a secure channel, i.e. a secure messaging session, requires a successful mutual authentication between the ASE and hosting device.

The following scenarios shall terminate a secure channel:

* Power off or reset of the ASE
* Reselection of the ASE applet
* A command with an incorrect MIC is received by the ASE
* A command in clear text is received by the ASE in an encrypted secure messaging session.

The present annex does not mandate any specific secure channel mechanism to allow alignment with contextual requirements. Example of relevant secure channel mechanisms include the following:

* Secure Channel Protocols (SCP) specified in the GlobalPlatform Card Specification [63], such as SCP 11 or SCP 03
* Secure channel mechanisms specified in the GSMA eUICC specification [eUICC]
* Secure Channel mechanisms specified in EN 419 212 [64] or FIPS 201-2 [FIPS 201].

# L.4 ASE Supported Functions

### L.4.1 ASE Verifiable Certificates

These are certificates stored in the ASE and used in asymmetric key mutual authentication. The ASE Verifiable Certificate is issued and signed by a trusted certificate authority (CA) and stored in the hosting device to show that it (and so the entity behind it) can be trusted. This certificate is referred as C\_CV.IFD.AUT. The ASE can check that the ASE Verifiable Certificate in the hosting device can be trusted by using the CA’s public key.

Similarly, the ASE may contain a certificate issued and signed by the CA, called the C.ICC.AUT. The hosting device can check that this certificate was genuinely issued and signed by the CA by using the CA’s public key.

In EN 419 212 [64], ASE Verifiable Certificates used in RSA-based device authentication are non self-descriptive (i.e. the tags and lengths of the signature elements are not included in the format), while SE Verifiable Certificates used in Elliptic Curve Device Authentication are self-descriptive. Such SE Verifiable Certificates include a Certificate Holder Authorization (CHA) that may be used as a security condition to access relevant sensitive data.

### L.4.2 ASE Secure Storage

#### L.4.2.1 Overview

An ASE shall support a way to store information in its protected non volatile memory.

File objects comprise Elementary Files (EFs), and DFs used to organize the file structure in a hierarchy. EFs store This can be used for information meant to be exchanged with external entities: This includes permanent storage of stakeholder information, storage of service credentials, and storage of data for service processing. This can be updated dynamically during operation provided that access control conditions are satisfied.

Data objects are meant to store information used during internal processes such as secret keys. The structures for Data objects may need to be reserved during the personalisation phase but their content can be updatable, if desirable, during the operational phase.

#### L.4.2.2 PIN

PINs may be used to identify a user and to protect data. See clause L.4.6 for further details.

#### L.4.2.3 Symmetric secret keys

Symmetric secret keys are 16-byte, 24-byte or 32-byte AES keys used for symmetric key mutual authentication. Two secret keys, KENC and KMIC, are shared by the secure element and its host, and can be diversified, for example by using the secure element serial number. Mutual authentication consists of each entity proving that it possesses the two keys to the other entity. A symmetric key can optionally be protected by a ratification counter. There may be multiple key pairs (KENC, KMIC) in an ASE. They shall be created together and initialized during the personalization phase.

#### L.4.2.4 Public keys

RSA and ECC public keys are associated with private keys in a key pair sharing a common one byte identifier, KID. These could be used for mutual authentication or to verify a signature or certificate. RSA Public Keys can also be used to encrypt sensitive data, while ECC Public Keys can be used to derive a symmetric shared key (ZZ) to be used to encrypt data.

The typical process to create a key pair in an ASE requires reservation of space for an Asymmetric Key Header during the personalization phase. This initializes a key container with at least a public portion and optionally a private portion.

The following public keys are generally stored in the ASE:

* CA public keys used in asymmetric key mutual authentication
* RSA and ECC public keys used by the application

More than one CA may store its public key PuK.CA.AUT on an ASE.

RSA public keys always contain a modulus, N, and a public exponent, e.The keys may be automatically updated by ASE internal process, or the keys may be generated outside the secure element.

#### L.4.2.5 Private keys

Private keys are used for public key cryptographic operations of M2M applications, such as generation of digital signatures, sensitive data decryption, and asymmetric scheme mutual authentication.

Private keys are always stored in the ASE to be adequately protected. They may be initialized either during the personalization phase or during the operational phase.

#### L.4.2.6 Diffie-Hellman Key Exchange parameters

The Diffie–Hellman key exchange parameters used in asymmetric key mutual authentication may also be stored in the ASE.

## L.4.2.7 Arbitrary Application Data

This provides a service to create, store, update and delete application data in the SE.

## L.4.2.8 ProfileData

This provides a service to store and protect profile data. A profile is the representation of parameters and data for its application, keys, and load files.

## L.4.3 On-Board Key Generation (OBKG)

The On-Board Key Generation functionality enables creation of a public / private key pair within an ASE, so that the private key never leaves the ASE which protects it during storage and usage (e.g. to sign a certificate).

OBKG is initiated when a command is sent to the ASE to initialize or update the value of a key pair when the ASE is in Operational state. This command only generates new values for private key and public key and returns the public key value in its response.

On-Board Key Generation has several advantages:

* The ASE performs the computation of the key values. The key value is not precomputed or imposed by an external entity.
* As the key update takes place within the ASE, the secure element handles the security of the operation instead of the hosting application.
* The command may need to satisfy access conditions to the private key data object in order to update the value of the private key data object.
* The new private key value never leaves the secure element.
* The life span of the key pair can be easily managed within the application, e.g. by regular renewals in order to adapt to the specific risks to which the key pair may be exposed.

## L.4.4 Digital Signature

#### L.4.4.1 Overview

The ASE may be used to generate Digital Signatures, by which a message is authenticated by the receiver to ensure that it is sent by the intended sender and that the message was not altered since it was sent. The signatures are generated using the Digital Signature keys stored in the ASE.

#### L.4.4.2 Digital Signature Generation

The digital signature generation process is the computation of the message signature using the digital signature private key on a pre-computed message hash digest. As the signature is generated using the sender's private key which is securely stored in the ASE, the message can only be sent by authorized sender and not by anybody else.

The digital signature creation process is as follows:

1. **Message Hashing.** The sender (Host Application) computes the hash of the original message using a hash algorithm. The host application calls a command to perform the hashing.
2. **Formatting Hash to Digital Signature Input (DSI)**. The ASE pads the hash to the length and format indicated by the hashing command.
3. **Signature Creation.** The hash is ciphered with the sender's private key. The result is known as the signature
4. **Digitally Signed Message Sending**.The signature is appended to the original message and sent.

#### L.4.4.3 Message Hashing

The generation of the hash may be performed in three ways:

1. performed entirely by the ASE using a dedicated command
2. performed externally
3. partially performed by the ASE and partially performed externally (in this case, the data is split).

For RSA Signatures, the ASE may use any of the following secure hash algorithms:

* SHA-256
* SHA-384
* SHA-512

For ECC signatures, the ASE may also use any of these SHA algorithms.

#### L.4.4.4 Formatting Hash to Digital Signature Input (DSI)

The generated hash is typically shorter than the required length of the Digital Signature Input and needs to be padded accordingly. The DSI needs to conform to a particular format, so the hash cannot be simply padded by adding a padding character. For this reason, the ASE is able to perform the necessary padding.

#### L.4.4.5 Signature Creation

The ASE uses the DSI to compute the digital signature upon instruction from its host.

The following algorithms are supported:

* ALG\_ECDSA\_SHA\_256 :Signature algorithm ALG\_ECDSA\_SHA\_256 generates a 32-byte SHA-256 digest and signs/verifies the digest using ECDSA with the curve defined in the ECKey parameters - such as the P-256 curve specified in the Digital Signature Standards specification[FIPS186-2].
* ALG\_ECDSA\_SHA\_384 :Signature algorithm ALG\_ECDSA\_SHA\_384 generates a 48-byte SHA-384 digest and signs/verifies the digest using ECDSA with the curve defined in the ECKey parameters - such as the P-384 curve specified in the Digital Signature Standards specification[FIPS186-2].
* ALG\_ECDSA\_SHA\_512 :Signature algorithm ALG\_ECDSA\_SHA\_512 generates a 64-byte SHA-512 digest and signs/verifies the digest using ECDSA with the curve defined in the ECKey parameters - such as the P-521 curve specified in the Digital Signature Standards specification[FIPS186-2].

#### L.4.4.6 Integrity of the Data to be Signed

The ASE may check integrity of the data to be signed, as required by some signature certification schemes.

#### L.4.4.7 Digital Signature Verification

The digital signature verification typically involves decrypting the signature using the sender's public key and hashing the original message using the hashing algorithm. If the hashes are equal, the signature is valid.

As the signature is created using the sender's private key, it can only be verified by the sender's public key. By verifying the signature, the recipient has proof that the sender's private key was used to encrypt the message hash and that the message has not been altered.

Since this does not require a high level of security, this process is typically performed externally and the ASE is not involved in this operation.

The principle of digital signature verification is shown for informational purposes only:

1. The receiver uses the sender's public key to decrypt the signature and retrieve the message hash.
2. The receiver hashes the original message and compares it with the result obtained in step 1. If the two hashes match, then the sender is authentic.

### L.4.5 Encryption and Decryption

#### L.4.5.1 Overview

Public key pairs may be used for encryption and decryption of sensitive data, typically symmetric session keys.

In the case of RSA, the public key of the receiver’s RSA key pair is used to encrypt messages and the private key of the key pair stored in the ASE is used to decrypt the message. The external entity uses the ASE’s public key to encrypt the message, which is not a sensitive operation, while the ASE uses the corresponding private key to decrypt the message internally using the PSO- Decipher (RSA use) decryption function. This process ensures that only the intended recipients can decrypt and read the message. Upon successful completion of the command, the ASE returns the deciphered message in the response.

In the case of ECC, the public key of the receiver (the ASE) is used to derive a shared key ZZ, which is used to encrypt and decrypt data. The key is generated by the ASE.

*Editor’s note: Assess and specify applicability of ASE based encryption/decryption to M2M Primitives and parameters.*

For security reason, it is strongly recommended to never use the same private key for deciphering and signing.

The following subclauses provide an example of a message encryption and decryption process wherein the encrypted data is a one-time session key that has been used to encrypt another message.

#### L.4.5.2 RSA Message Encryption and Decryption

The message encryption process is performed by the message sender (external entity). The process includes the following steps:

1. Message Encryption. The message sender encrypts the document with a one-time session key. Typically, this is an AES session key.
2. Symmetric Key Encryption. The message sender encrypts the symmetric session key with the host application RSA public key with a specified padding, e.g. PKCS #1.
3. Message Sending. The message sender sends the encrypted session key and the encrypted message to the host application.

The message decryption occurs in the host application. The process includes the following steps:

1. Symmetric Key Decryption. Upon receiving the message, the host application instructs the ASE to decrypt the symmetric key. The ASE returns the decrypted symmetric key in the response.
2. Message Decryption. The host application decrypts the message using the symmetric key retrieved in Step 1. This step is performed by the host application.

For security reasons, it is strongly recommended not to use the same private key for decryption and signing.

The messages to be decrypted may be protected by e.g. RSASSA PKCS#1 v1.5 algorithm or RSAES OAEP algorithms.

#### L.4.5.3 ECC Message Encryption and Decryption

**Encrypting a Message (ECC):**

The steps are as follows:

1. The sender derives a shared key, ZZ, from the ASE certified public key (yb) and the hosting device ephemeral private key (ra). This process involves generation of a random challenge.
2. The sender encrypts a message using ZZ.

**Decrypting a Message (ECC):**

1. The sender sends both the encrypted message and his/her public key (ya) to the ASE acting as the receiver.
2. The receiver uses ZZ to decrypt the messsage.

#### L.4.5.4 AES Message Encryption and Decryption

The following methods may be supported according to TS-0016 [TS16]:

* ALG\_AEAD\_AES\_128\_GCM: The AEAD\_AES\_128\_GCM authenticated encryption algorithm works as specified in RFC 5116 [RFC5116], using AES-128 as the block cipher, by providing the key, nonce, and plaintext, and associated data to that mode of operation.
* ALG\_AEAD\_AES\_256\_GCM: This algorithm is identical to AEAD\_AES\_128\_GCM, but with the following differences: K\_LEN is 32 octets, instead of 16 octets, and AES-256 GCM is used instead of AES-128 GCM.
* ALG\_AEAD\_AES\_128\_CCM: The AEAD\_AES\_128\_CCM authenticated encryption algorithm works as specified in RFC 5116 [RFC5116], using AES-128 as the block cipher, by providing the key, nonce, associated data, and plaintext to that mode of operation.
* ALG\_AEAD\_AES\_256\_CCM: This algorithm is identical to AEAD\_AES\_128\_CCM, but with the following differences: K\_LEN is 32 octets, instead of 16, and AES-256 CCM is used instead of AES-128 CCM.
* ALG\_AEAD\_AES\_128\_CCM\_8: The AEAD\_AES\_128\_CCM\_8 authenticated encryption algorithm is identical to the AEAD\_AES\_128\_CCM algorithm (see Section 5.3 of RFC 5116 [RFC5116]), except that it uses 8 octets for authentication, instead of the full 16 octets used by AEAD\_AES\_128\_CCM (see Section 6.1 of [RFC 6655]).
* ALG\_AEAD\_AES\_256\_CCM\_8: The AEAD\_AES\_256\_CCM\_8 authenticated encryption algorithm is identical to the AEAD\_AES\_256\_CCM algorithm (see Section 5.4 of RFC 5116 [[RFC5116](http://www.rfc-archive.org/getrfc.php?rfc=5116)]), except that it uses 8 octets for authentication, instead of the full 16 octets used by AEAD\_AES\_256\_CCM (see Section 6.2 of RFC 6655 [31]).
* ALG\_AES\_BLOCK\_128\_CBC\_NOPAD : Cipher algorithm ALG\_AES\_BLOCK\_128\_CBC\_NOPAD provides a cipher using AES with block size 128 in CBC mode and does not pad input data.
* ALG\_AES\_CBC\_ISO9797\_M1: Cipher algorithm ALG\_AES\_CBC\_ISO9797\_M1 provides a cipher using AES with block size 128 in CBC mode, and pads input data according to the ISO 9797 [ISO9797] method 1 scheme.
* ALG\_AES\_CBC\_ISO9797\_M2: Cipher algorithm ALG\_AES\_CBC\_ISO9797\_M2 provides a cipher using AES with block size 128 in CBC mode, and pads input data according to the ISO 9797 [ISO9797] method 2 (ISO 7816-4, EMV'96) scheme.
* ALG\_AES\_CBC\_PKCS5: Cipher algorithm ALG\_AES\_CBC\_PKCS5 provides a cipher using AES with block size 128 in CBC mode, and pads input data according to the PKCS#5 scheme.

### L.4.6 User Authentication through PIN

PINs are used to identify the owner of an ASE and to protect its data.

A data object in the ASE may be protected by a PIN. In this case, access to the object shall only be allowed upon successful verification of the PIN.

An ASE may also support an Activation PIN verification mechanism to prevent unauthorized use of the ASE before verification that the ASE is provided to the authorized owner.

The Activation PIN needs to be presented once only during the Operational Phase.

The ASE may also support a "Force PIN Change Before Signature" mechanism.

If the feature is activated after personalization and if the Digital Signature key is protected by a PIN, the PIN shall be changed at least once after personalization to make the signature functionality available.

## L.4.7 TLS-Handshake

The ASE may provide services for the establishment of TLS channels (Handshake), including:

* Generation of random numbers for the TLS command ClientHello
* Key negotiation of the TLS pre-master secrets
* Signature generation and verification for the TLS authentication
* Securing the data sent via the negotiated TLS channel

The applicable cipher suites are listed in clause 10.2.

## L.4.8 getSEFunctions

## This service provides a list of available sensitive functions provided by the secure elementL.4.9 Random numbers

## This service provides random numbers to the hosting deviceL.4.10 Calculating MICs

This service calculates MICs. The following algorithm may be supported:

* ALG\_AES\_CMAC\_128 : Signature algorithm ALG\_AES\_CMAC\_128 generates a 16-byte Cipher-based MAC (CMAC) using AES with blocksize 128 in CBC mode with ISO9797\_[ISO9797] M2 padding scheme.
* ALG\_AES\_MAC\_128\_NOPAD :Signature algorithm ALG\_AES\_MAC\_128\_NOPAD generates a 16-byte MAC using AES with blocksize 128 in CBC mode and does not pad input data.
* ALG\_HMAC\_SHA\_256 :HMAC message authentication algorithm ALG\_HMAC\_SHA\_256 This algorithm generates an HMAC following the steps found in RFC 2104 [33] using SHA-256 as the hashing algorithm.
* ALG\_HMAC\_SHA\_384 :HMAC message authentication algorithm ALG\_HMAC\_SHA\_384 This algorithm generates an HMAC following the steps found in RFC 2104 [33]using SHA-384 as the hashing algorithm.
* ALG\_HMAC\_SHA\_512 :HMAC message authentication algorithm ALG\_HMAC\_SHA\_512 This algorithm generates an HMAC following the steps found in RFC 2104 [33] using SHA-512 as the hashing algorithm.

## L.4.11 Device Authentication

This service provides authentication of the hosting device, verifying the authenticity of remote entities and negotiating session keys for protecting the communication between the mutual authenticated entities.

### .-----------------------Start of change to REFERENCES----------------------------------

# 2 References

## 2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non‑specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

The following referenced documents are necessary for the application of the present document.

[1] oneM2M TS-0001: "Functional Architecture".

[…]

[TS16] oneM2M TS-0016 “Secure Environment Abstraction Layer”

[3109] BSI TR 03109 Smart Meter Gateway specification

[FIPS201] NIST Federal Information Processing Standard 201-2, Personal Identity Verification (PIV) of Federal Employees and Contractors, August 2013

[eUICC] GSMA: “SGP.01 - Embedded SIM Remote Provisioning Architecture”

[FIPS186-2] NIST Federal Information Processing Standard 186-2, Digital Signature Standard (DSS)

[RFC5116] IRTF RFC 5116, “An interface and algorithms for authenticated Encryption”, 2008-01

[ISO9797] ISO 9797 “Information Technology – Security Techniques – Message Authentication Codes (MACs)”, 2011

### -----------------------End of change to REFERENCES---------------------------------