USB iTC: Functional Requirements

1. Security Requirements

The individual functional security requirements for a USB Portable Storage Device are specified in the sections below. These will be used to create SFRs in the final cPP.

1.1 Introduction

The USB Portable Storage Device (hereafter referred to as "the device" or "the TOE") is a portable storage device that provides a USB interface for connecting to a host computer. The device employs cryptographic means to provide the necessary protection of user data, and the strength of these cryptographic means lies in the quality of the algorithms, the mode used and the key sizes as well as the entropy of the authorisation factor (e.g., password, passphrase) and cryptographic keys. The device encrypts the user data as it is stored on the device, and decrypts the user data as it leaves the device. All cryptographic functions (encryption/decryption of user data, hashing, random number generation, etc.) are implemented on the device itself. This precludes, for example, encryption/decryption being carried out in a driver on the host computer.

The use of encryption is intended to ensure that data is protected such that even if sophisticated inspection tools were employed to read data from a found or stolen device then any successful access to the data would require a prohibitive cryptanalytic effort.

In addition to protecting the user data, the device is also responsible for ensuring the system data (e.g., software, firmware) cannot be modified by untrusted entities through the logical interface.

The device is dedicated to storing user data, which may or may not include removable memory media in the device. This cPP is not suitable for use with more general USB devices that provide access to other forms of storage such as CDs or DVDs, nor for more complex devices that may include memory media (such as smartphone, cameras or media players).

The device, its boundaries, components and connections are depicted in Figure 1.

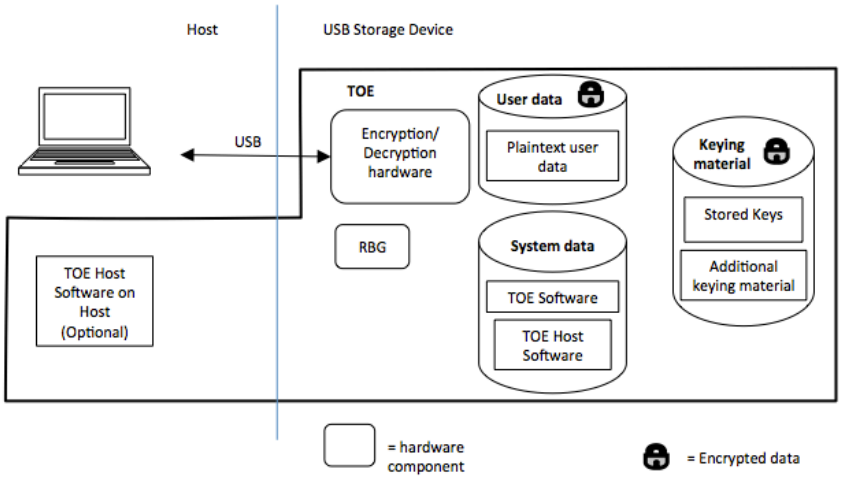

Figure 1 USB device boundaries, components and connections

It is intended to be used for the following scenarios:

- Transfer of sensitive data between two host computers.
- Long term storage of sensitive data.

In this cPP we refer to "authorisation" rather than "authentication" to reflect the fact that the device does not need to hold reference authentication data, or other data that identifies users as individuals. Authorisation data includes passphrases and possibly other data (such as cryptographic salt values), and is used to derive a Key Encryption Key (KEK). The KEK is used to encrypt a Data Encryption Key (DEK) that is generated on the device and used to encrypt and decrypt data stored on the device. To support the intention that a found or stolen device presents a prohibitive cryptanalytic challenge to an attacker, the authorisation data that ultimately give access to the DEK and the user data must not be stored unencrypted on the device.

The host computer plays no role in the encryption/decryption of user data. Software running on the host computer could gather the necessary authorisation data but must not perform any other authorisation functions. The TOE Host software is then considered as part of the TOE and is subject to requirements to provide certain functionality to users of the device.

In addition to user data, the device may also hold data that determines operation of the device itself (including software that may be downloaded to a host computer, such as driver software). This data is referred to as "system data". System data may include (but are not limited to) software or firmware, patches, or configuration data. The TOE provides the integrity protection of the system data. The host computer plays no role in protecting the system data that resides on the device.

All data on the device is either user data, system data, keying material or else unused (unused areas of the device may contain old encrypted user data or old encrypted keying material).

Several optional capabilities for the device are also defined and identified in the relevant requirement sections below.

1.2 Authorisation

1.2.1 Authorisation Model

For the purposes of defining the security requirements for user authorisation, the concept of a "session" is introduced and is defined as the period of operation of the device from the point at which it requests authorisation data to the point at which the authorisation data and further access to the plaintext user data expires. This is illustrated in Figure 2.

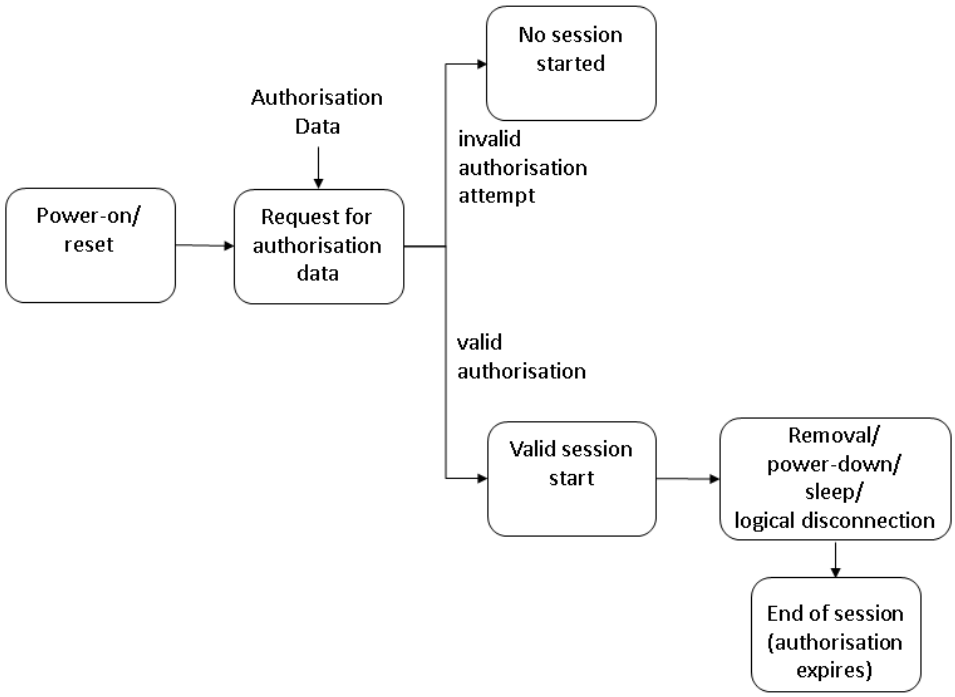

Figure 2 Session Lifecycle

At power-on (or other equivalent logical start, such as a reset) the TOE requests authorisation data, and no access to the plaintext user data and keying material is possible at this stage. If invalid authorisation data is entered then there is no session started, and the plaintext user data and keying material remains inaccessible. If valid authorisation data is received then a session is started and the keying material and plaintext user data is made available until the device is removed from the host, the host enters a power-down or sleep state, or there is a logical disconnection from the host. Any of the preceding disconnection events results in the end of the session and the plaintext user data and keying material is rendered inaccessible.

The user authorisation model includes:

- Input of authorisation data (section 1.2.2)
- Conditioning of authorisation data (section 1.2.3)
- Protection of authorisation data (section 1.2.4)
- Authorisation failures (section 1.2.5)

- Authorisation data recovery (section 1.2.6)
- Authorisation data change (section 1.2.7)
- Validation of authorisation data (section 1.2.8)

In order to allow flexibility in product implementation, some requirements are optional (being claimed in a Security Target only if a product provides that functionality). For example, the requirement that support authorisation data change should be only included in a Security Target if the product allows the authorised users to change the authorisation data (see section 1.2.7).

1.2.2 Input of authorisation data

An authorised user of the device provides authorisation data directly to the device or else via TOE software running on a host computer. Software running on the host computer that gathers the necessary authorisation data is considered part of the TOE. The current version of the cPP supports only passphrases as the authorisation data.

Requirements:

Input of authorisation data:

- The device shall support passphrases with a maximum length of at least 64 characters. The passphrase shall have a minimum length of at least 8 characters.
- The device shall support passphrases that consist of upper case characters, lower case characters, digits and other supported characters.
- The host software (where used in order to enter authorisation data to the device) shall gather authorisation data without displaying the values of the authorisation data.
- The host software (where used in order to enter authorisation data to the device) shall gather authorisation data without persistently storing the values of the authorisation data, or storing this data in non-volatile memory.
- The host software (where used in order to enter authorisation data to the device) shall accept all authorisation data values accepted by the device (this means that the host software does not reduce the authorisation data choices available to a user, and that the device itself is responsible for enforcing any quality metrics on the authorisation data (e.g. rejecting passphrases that are below the minimum length).
- The device must ensure that successful user authorisation has been achieved (i.e. a valid session starts, see Figure 2) before any of the following actions:
    a) accessing, modifying user data on the device
    b) storing user data on the device
    c) modifying system data.

- The device must ensure that authorisation expires (see Figure 2):

    a) on removal of the device from a host

    b) whenever a host to which it is connected enters a power-down or sleep state

    c) on logical termination of the connection to the host.

1.2.3 Conditioning of authorisation data

Authorisation data is used to determine whether the user is authorised to access the plaintext user data on the device. When the user enters valid authorisation data then a valid session is started (see Figure 2) and the plaintext user data is made available to the user. The authorisation data is used to derive a Key Encryption Key (KEK). More information on requirements for user data encryption is provided in section 1.3.1.

Requirements:

Conditioning:

- The device shall perform passphrase conditioning using NIST SP 800-132 where the salt is generated by the device's RBG*, an iteration count greater than or equal to 1000, and HMAC using SHA-1, SHA-256 or SHA-512.*
- The submask resulting from the conditioning function shall be at least the same length (in number of bits) as the DEK.

1.2.4 Protection of authorisation data

Requirements:

Protection:

- Authorisation data shall not be persistently stored in the TOE, including TOE software running on a host computer, and shall not be stored in non-volatile memory.

1.2.5 Authorisation failures

When the user enters invalid authorisation data then there is no session started, and the plaintext user data and keying material remains inaccessible. If the user enters a threshold value of consecutive invalid authorisation attempts then the device permanently deletes the DEK and therefore makes any data encrypted under the current DEK inaccessible. The number of consecutive failed authorisation attempts will be specified in the Security Target for the specific device.

Authorisation failure may be detected as a result of detecting that the KEK or DEK value obtained is incorrect, or as a result of an optional separate authorisation data validation mechanism as specified in section 1.2.8.

Requirements:

Authorisation failures:

- The device shall limit consecutive failed authorisation attempts. The device must make the user data permanently unavailable by deleting the current DEK as soon as a threshold number of consecutive authorisation failures is reached.

1.2.6 Authorisation data recovery

If the device provides a mechanism to enable data recovery in the event of loss of authorisation data (e.g. to allow the DEK to be recovered in case authorisation data is lost), then the device must allow this mechanisms to be disabled, such that it cannot be re-enabled by an unauthorised user, unless this enablement also requires generating a replacement DEK (and hence making any data encrypted under the current DEK inaccessible).

Requirements:

Disable authorisation data recovery:

- The authorised users shall be able to disable data recovery mechanisms if the device provides such a recovery mechanism.
- Once disabled the device shall not allow the recovery mechanism to be enabled, unless this enablement also requires generating a replacement DEK (and hence making any data encrypted under the current DEK inaccessible)

This is an optional requirement, but must be implemented if the device supports a recovery mechanism.

1.2.7 Authorisation data change

The device allows the user to change their passphrase to a new value, but only on correct presentation of the current value. This requirement ensures that authorised users can change the value of the authorisation data without losing access to the user data stored on the device.

Requirements:

Authorisation data change:

- The device enables authorised users to change the value of the authorisation data, but only when supplying correct current authorisation data as part of the change operation.

1.2.8 Validation of authorisation data

The device may provide an additional function that validates the authorisation data separately from using the authorisation data to derive the KEK and decrypt the DEK.

Requirements:

- The device shall perform validation of the authorisation data using the following methods:
    - key wrapping as specified in section 1.3.4
    - comparison of a stored value with a hash of the authorisation data submask using the hash or keyed hash as specified in section 1.3.9
    - comparison of a stored value with the output of a decryption using an element derived from the authorisation data.
- The device shall only decrypt user data within a session after the authorisation data has been successfully validated.

This is an optional requirement (because the device does not need to provide a separate validation mechanism), but must be implemented if the device supports an authorisation data validation mechanism that is separate from the derivation of the KEK.

1.3 Encryption

1.3.1 Reference Key Model

The key architecture used by a product may differ in the number of layers of keys, and the number of keys used at each layer. However, this cPP describes a reference key model that is used to identify certain elements of a key architecture on which the cPP places specific requirements. The Security Target author describes the correspondence of the key architecture of a *specific* product to the cPP reference key model by a combination of information in the TOE Summary Specification of the Security Target and the Key Management Description information specified in Appendix B.

In the figures for the reference key model data stores are indicated by a rectangle. If the rectangle has a single side bar on the left, it signifies an ephemeral or transient storage that may persist up to the conclusion of the session or be zeroized immediately after use. In either case, the single bar data stores are zeroized after use. If there are two bars on the left side of the rectangle, the data store is considered permanent, i.e. it is stored in non-volatile memory and persists between powered-off events. The round-cornered rectangles are processes and arrows indicate data flows between the other components.

The reference key model is based on the use of a Data Encryption Key that is generated on the device (section 1.3.2) and then used to encrypt user data (section 1.3.5). The Data Encryption Key is never stored unencrypted in non-volatile memory on the device, but is stored encrypted under a Key Encryption Key (KEK). There may be additional levels of key

that are used to protect a KEK, but ultimately a specified key wrapping method is used to protect the DEK with the KEK (section 1.3.4).

A device may use more than one DEK, and/or more than one KEK.

The reference key model therefore includes:

- Generation of the DEK (section 1.3.2)
- Derivation of the KEK (section 1.3.3)
- Wrapping of the DEK with the KEK (section 1.3.4)
- Encryption of data under the DEK (section 1.3.5)
- Destruction of keys (section 1.3.6)
- Supporting cryptographic functions used to implement other device functionality (sections 1.3.7- 1.3.9).

In order to allow flexibility in product implementation, some requirements are optional (being required in a Security Target only if a product provides that functionality) and other requirements are determined by other selections of different options within requirements. For example, the requirements that support digital signature verification in support of trusted updates (section 1.3.10) are only included in a Security Target if the product supports updates based on digital signatures (as opposed to supporting updates based on a published hash or providing no support for updates, see section 1.4). Similarly, the key encryption requirements in section 1.3.8 are only included in a Security Target if the product uses this function (typically as part of the product-specific key chain described in section 1.3.3). However, the key wrapping requirements in section 1.3.4 are required in the Security Target of any conformant product because the use of this function is a requirement for the encryption of a DEK under a KEK regardless of the other aspects of the key chain.

Requirements:

All cryptographic operations described in this document are required to take place within the device.

1.3.2 DEK Generation

The key reference model for generation of the Data Encryption Key (DEK) is simply that the DEK is generated on the USB device using a suitable deterministic random bit generator:

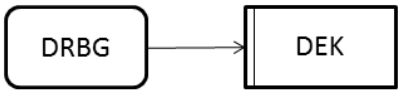

Figure 3 Key Reference Model for DEK generation

The DEK is never stored in plaintext in non-volatile memory on the device: it is always stored wrapped under the KEK.

Requirements:

DEK generation:

- the device shall generate a DEK of either 128 or 256 bits, using a deterministic RBG that is part of the device. The RBG must meet the requirements separately specified for the RBG

RBG:

- The device shall perform all deterministic random bit generation services in accordance with either ISO/IEC 18031:2011 or NIST SP 800-90A, using Hash_DRBG (any), HMAC_DRBG (any), or CTR_DRBG (AES).

- The entropy provided by the RBG is subject to the additional requirements in Appendix A

- The deterministic RBG shall be seeded by an entropy source within the boundary of the device that accumulates entropy from a software or hardware-based noise source with entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 “Security strength table for hash functions”, of the keys and hashes that it will generate.

1.3.3 KEK derivation

The KEK is derived from a chain of operations that starts from authorisation data (such as a passphrase or externally stored submask) obtained from authorisation factors (such as user entry at a keypad, in the case of a passphrase, or loading from a token in the case of an externally stored submask). The number of stages in this sequence is left to the implementation (this is the meaning of the dotted lines in Figure 4), but it must begin with the conditioned authorisation data (see section 1.2.3) and end in the creation of a KEK, in a manner that will preserve a security strength that is sufficient to protect the DEK.

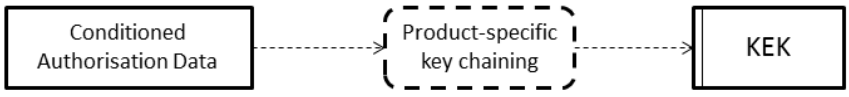

Figure 4 Key Reference Model for KEK derivation

The KEK is never stored in plaintext in non-volatile memory on the device: it is either recreated directly from the authorisation data whenever it is needed, or else is obtained as the last step in the key chaining process.

Requirements:

KEK derivation:

- the device shall derive a KEK that is at least the same length as the DEK using one or more of the following methods in combination:

    a) using the conditioned authorisation data meeting the requirements in section 1.2.3 as the KEK

    b) intermediate key generation as specified in section 1.3.7

    c) key wrapping as specified in section 1.3.4

    d) key encryption as specified in section 1.3.8

  while maintaining the effective strength of the DEK (i.e. 128 bits or 256 bits).

- When generating an intermediate key the device shall combine submasks using the functions specified in the submask combination requirements in section 1.3.9.

Application Note: the Key Management Description (see Appendix B) is required to make clear how the product-specific key chaining is specified in the SFRs (in terms of which operations and keys are covered by which SFRs).

1.3.4 DEK Wrapping

A DEK is always wrapped by a KEK when it is stored in non-volatile memory. (The KEK may itself be wrapped by other keys or may be derived from the authorisation data – see section 1.3.3.) The DEK exists in plaintext on the device only in volatile memory, and only after successful authorisation of a user has caused it to be unwrapped within the current session.

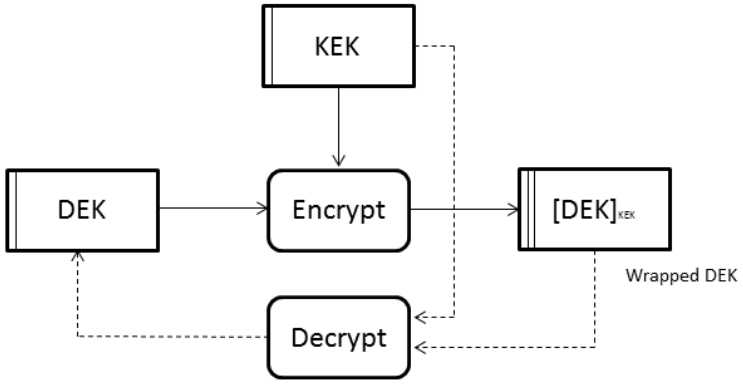

Figure 5 DEK wrapping with KEK

The KEK is an ephemeral value as shown in Figure 5, but in some implementations may itself be stored in an encrypted form (section 1.3.8) or wrapped form (as described in this section).

Requirements:

DEK storage:

- The DEK shall never be stored in plaintext form in non-volatile memory on the device
- The DEK shall only be stored in non-volatile memory on the device when wrapped according to the requirements separately specified for DEK wrapping
- The DEK shall only be unwrapped after successful authorisation of a user
- The device shall not support export of the DEK from the device in any form (plaintext, wrapped, or otherwise encrypted)
- The device shall not support import of the DEK to the device in any form (plaintext, wrapped, or otherwise encrypted).

DEK wrapping:

- The device shall wrap the DEK with a KEK according to the requirements separately specified for key wrapping.

Key wrapping:

- The device shall wrap keys using AES in one of the modes KW, KWP, GCM, or CCM, and meeting ISO/IEC 18033-3 (AES) and either ISO/IEC 19772 or NIST SP 800-38F.

1.3.5 Whole Device Encryption

Requirements:

Whole device encryption:

- The device shall encrypt all user data that is stored on the device, without user intervention; the device shall not contain any plaintext user data in non-volatile memory.
- The device shall perform data encryption and decryption of user data using AES in CBC, GCM or XTS mode according to the following standards: AES as specified in ISO/IEC 18033-3, CBC as specified in ISO/IEC 10116, GCM as specified in ISO/IEC 19772, and XTS as specified in IEEE 1619
- See also requirements for salt, nonce, and initialisation vector (IV) generation in section 1.3.9 for mode-related requirements.

1.3.6 Key Destruction

Requirements:

Key Destruction:

- The device shall destroy all cryptographic keys and keying material (whether in plaintext or encrypted form) using the following methods according to the type of memory where the key has been held:

    a) For volatile memory, the erasure shall be executed by a single direct overwrite using either a pseudo-random pattern generated by the device's RBG or a string of zeroes.
    b) For non-volatile storage, the erasure shall be executed by at least one overwrite of the key data physical storage location using either a pseudo random pattern generated by the device's RBG or a static pattern.

1.3.7 Intermediate Key Generation

Generation of intermediate keys using a key derivation function (KDF) is allowed as one of the optional steps in the product-specific key chaining between conditioned authorisation data and the KEK (cf. section 1.3.3).

Requirements:

Intermediate key generation:

- where intermediate keys are used in a sequence of keys between the conditioned authorisation data and the KEK that ultimately protects the DEK, then the intermediate keys shall be generated based on input to a KDF from a deterministic RBG or generated on the device using a deterministic RBG.

- The KDF shall meet either

    a) NIST SP 800-108 with KDF in Counter Mode, KDF in Feedback Mode, or KDF in Double-Pipeline Iteration Mode; or

    b) NIST SP 800-132, using the keyed-hash functions specified in section 1.3.9,

    such that the output is at least of equivalent security strength (in number of bits) to the DEK.

- The deterministic RBG shall be part of the device and shall meet the deterministic RBG requirements in section 1.3.2 such that the output is at least of equivalent security strength (in number of bits) to the DEK.

1.3.8 Key Encryption

Key encryption is allowed as one of the optional steps in the product-specific key chaining between conditioned authorisation data and the KEK (cf. section 1.3.3).

Requirements:

Key encryption:

- The device shall perform key encryption and decryption using 128-bit or 256-bit AES in CBC or GCM mode, meeting the standards: AES as specified in ISO /IEC 18033-3, CBC as specified in ISO/IEC 10116, and GCM as specified in ISO/IEC 19772.

1.3.9 Supporting Cryptographic Functions

Requirements:

Salt, nonce, and initialisation vector (IV) generation:

- The device shall only use salts that are generated by the device's RBG
- The TSF shall only use unique nonces with a minimum size of 64 bits
- The TSF shall create IVs in the following manner:
  - CBC: IVs shall be non-repeating and unpredictable,
  - XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer,
  - GCM: IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^32 for a given secret key.

***Application Note**: This requirement covers several important factors – the salt must be random, but the nonces (for CCM) only have to be unique (a nonce is called a Starting Variable (SV) in ISO/IEC 19772). For tweak values, "assigned consecutively" could mean using a one-up counter.*

Submask combination:

- When generating an intermediate key the device shall combine submasks using:
  a) exclusive OR (XOR),
  b) SHA-256, or
  c) SHA-512.

Hash:

- The device shall perform cryptographic hashing using either SHA-256, SHA-386 or SHA-512 in accordance with ISO/IEC 10118-3:2004.

Keyed hash:

- The device shall perform keyed-hash message authentication using either HMAC-SHA-256 or HMAC-SHA-512 in accordance with ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2".

  Application Note: the key size to be used in the HMAC falls into a range between L1 and L2 defined in ISO/IEC 10118 for the appropriate hash function for example for SHA-256 L1 = 512, L2 =256) where L2 ≤ k ≤ L1.

1.3.10 Protection of updates

Requirements:

Protection of updates:

- The device shall use one of the following cryptographic methods to verify the integrity of the updates:

  a) RSA Digital Signature Algorithm with a key size (modulus) of 2048 bits or greater, meeting

     - FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS2v1_5; or

     - ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3

  b) Elliptic Curve Digital Signature Algorithm and cryptographic key sizes of 256 bits or greater, meeting

     - FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, implementing "NIST curves" P-256, P-384, and (optionally) P-521; or

     - ISO/IEC 14888-3, Section 6.4

  c) A published hash using SHA-256 or SHA-512 meeting ISO/IEC 10118-3:2004.

  d) Keyed hash using either HMAC-SHA-256 or HMAC-SHA-512 in accordance with ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2".

Application Note: the key size to be used in the HMAC falls into a range between L1 and L2 defined in ISO/IEC 10118 for the appropriate hash function for example for SHA-256 L1 = 512, L2 =256) where $L2 \leq k \leq L1$.

1.4 Secure Updates

1.4.1 Trusted Firmware Updates

The device may provide authorised users with the ability to initiate signed firmware updates using a digital signatures mechanism, a published hash or keyed hash mechanism. (Requiring authorisation before allowing updates to be initiated gives the authorised users the ability to control the firmware version in use and prevents updates being accidentally or maliciously applied.)

Requirements:

Trusted firmware updates:

- If the device provides the capability to update the firmware, then before applying any update to the device it must verify the associated digital signature, a published hash or keyed hash using a mechanism that meets the requirements in section 1.3.10 so as to demonstrate that the update has been issued by a source that is authorised to provide such updates.

- Optionally the device may prevent rollback of previously applied updates.

This is an optional requirement, but must be met if the device supports updates to firmware. Prevention of rollback is optional even when the device supports updates to firmware.

1.4.2 Host Software Updates

The device may provide authorised users with the ability to initiate signed host software updates using a digital signatures mechanism, a published hash or keyed hash mechanism. (Requiring authorisation before allowing updates to be initiated gives the authorised users the ability to control the host software version in use and prevents updates being accidentally or maliciously applied.)

Requirements:

Trusted host software updates:

- If the device provides the capability to update the device host software, then before applying any update, it must verify the associated digital signature, published hash or keyed hash using a mechanism that meets the requirements in section 1.3.10 so as to demonstrate that the update has been issued by a source that is authorised to provide such updates.

- The area where the device host software resides (if applicable) shall be read-only (if updates to host software are not supported) or else shall be writeable only via the trusted update mechanism.

- Optionally the device may prevent rollback of previously applied updates.

This is an optional requirement, but must be met if the device supports updates to host software. Prevention of rollback is optional even when the device supports updates to host software.

1.5 System integrity

Requirements:

- The device shall perform self-tests to ensure that the device is functioning properly when powered on and when reset (and optionally at other times). If the device fails a self-test, the device shall enter an error state or reset. The error state shall be a "mute" state in which the device does not respond or communicate with the host; in particular, the device shall not decrypt any user data stored on the device and shall not allow modification of the system data.

- The device shall run Known Answer Tests of the cryptographic algorithms supported by the device when the device is powered on and when reset (and optionally at other times).

- The device shall run firmware integrity tests when the device is powered on and when reset (and optionally at other times).

1.6 Management Functions

Management functions support other functionality provided by the device by enabling configuration or providing administrative functions.

Requirements:

The device shall support the following management functions for cryptographic aspects:

- The authorised user has the ability to generate a replacement DEK (including initial generation of the DEK) – this shall render any data encrypted under the old DEK inaccessible

- If the device provides a mechanism for recovering user data in the event of the loss of authorisation factors, then the device must also allow the authorised user to disable this mechanism so that data cannot be recovered in this way. Once disabled the device shall not allow the recovery mechanism to be enabled, unless this enablement also requires generating a replacement DEK (and hence making any data encrypted under the current DEK inaccessible)

- The device shall enable authorised users to change the value of some or all of the authorisation data, but only when supplying the current authorisation data as part of the change operation

- If the device supports updates then it shall provide a management function for the authorised user to initiate the update

- If the device supports updates based on digital signature then it shall not provide a management function for replacement of the public key certificate on which the digital signature verification is based

- Optionally the device may provide a management function to enable specification of constraints on authorisation data (e.g. see password length or composition constraints as in section 1.2.2, and the threshold for consecutive authorisation failures as in section 1.2.5)

- If the device supports different cryptographic options (e.g. algorithms or key lengths) to encrypt the user data, i.e. different algorithms or key lengths of the DEK, then it shall provide a management function to enable specification of these cryptographic options. Once new cryptographic options are set, the device shall generate a replacement DEK (and hence making any data encrypted under the current DEK inaccessible). Please note that some cryptographic options might not be supported by the USB cPP and therefore would lead to the device configuration that is not consistent with the evaluated configuration.

A. Entropy Documentation And Assessment

This appendix describes the required supplementary information for the entropy source used by the TOE.

The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide sufficient entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS in the public facing ST.

A. 1 Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. Any information that can be shared regarding the design should also be included for any third-party entropy sources that are included in the product.

The documentation will describe the operation of the entropy source to include, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the entropy comes from, where the entropy output is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

If implemented, the design description shall include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on shall be included.

A. 2 Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source delivering sufficient entropy for the uses made of the RBG output (by this particular TOE). . This argument will include a description of the expected min-entropy rate (i.e. the minimum entropy (in bits) per bit or byte of source data) and explain that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

The amount of information necessary to justify the expected min-entropy rate depends on the type of entropy source included in the product.

For developer provided entropy sources, in order to justify the min-entropy rate, it is expected that a large number of raw source bits will be collected, statistical tests will be performed, and the min-entropy rate determined from the statistical tests. While no particular statistical

tests are required at this time, it is expected that some testing is necessary in order to determine the amount of min-entropy in each output.

For third party provided entropy sources, in which the TOE vendor has limited access to the design and raw entropy data of the source, the documentation will indicate an estimate of the amount of min-entropy obtained from this third-party source. It is acceptable for the vendor to "assume" an amount of min-entropy, however, this assumption must be clearly stated in the documentation provided. In particular, the min-entropy estimate must be specified and the assumption included in the ST.

Regardless of type of entropy source, the justification will also include how the DRBG is initialized with the entropy stated in the ST, for example by verifying that the min-entropy rate is multiplied by the amount of source data used to seed the DRBG or that the rate of entropy expected based on the amount of source data is explicitly stated and compared to the statistical rate. If the amount of source data used to seed the DRBG is not clear or the calculated rate is not explicitly related to the seed, the documentation will not be considered complete.

The entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

A. 3 Operating Conditions

The entropy rate may be affected by conditions outside the control of the entropy source itself. For example, voltage, frequency, temperature, and elapsed time after power-on are just a few of the factors that may affect the operation of the entropy source. As such, documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent - it may be acceptable to simply state that if the device is operating outside of its operating conditions a sufficient level of entropy cannot be guaranteed. Methods used to detect failure or degradation of the source shall be included.

A. 4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

B. Key Management Description

In order to support the evaluation of the SFRs, some additional information is required to describe the keys that implement the SFRs and the ways that these keys relate to each other and to the data that they protect. In particular this information is important as part of a justification that a product Security Target includes an appropriate set of cryptographic SFRs and appropriate selections and assignments within those SFRs. This required supplementary information is referred to in this cPP as the 'Key Management Description', and is specified below.

The Key Management Description should be detailed enough that, after reading, the evaluator will thoroughly understand the keys involved in implementing the SFRs, the data that each key is used to protect and the ways in which each key is used to provide protection. This documentation is not required to be part of the TSS - it can be submitted as a separate document and marked as developer proprietary.

The following topics may not apply to all products, but where a requirement does not apply then the Key Management Description should include an explanation as to why the details do not apply.

The Key Management Description will provide the following information for all keys in the key chain:

- The purpose of the key (indicating its relationship to the Key Reference Model in section 1.3, and the SFRs in the ST that describe the use of the key)
- How and when the key is generated/derived (indicating SFRs in the ST that cover this)
- Whether the key is stored in non-volatile memory
- How and when the key is protected (indicating SFRs in the ST that cover this)
- The strength of the key
- Method of destruction of the key (indicating SFRs in the ST that cover this).

The Key Management Description will also describe the following topics:

- The process for validation of user authorisation shall be described, noting what value(s) is used for validation and the process used to perform the validation. It shall describe how this process ensures no keys in the key chain are weakened or exposed by this process. It shall describe the method used to limit the number of consecutively failed authorisation attempts.
- The authorisation process that leads to the ultimate release of the DEK. This section shall detail the key chain used by the product. It shall describe which keys are used in the protection of the DEK and how they meet the derivation or key wrap requirements. It shall also include any values that add into that key chain or interact with the key chain and the protections that ensure those values do not weaken or expose the overall strength of the key chain.
- The diagram and essay will clearly illustrate the key hierarchy to ensure that at no point could the chain be broken without a cryptographic exhaust or knowledge of the authorisation data, and that the effective strength of the DEK is maintained throughout the Key Chain.

- A description of the data encryption engine, its components, and details about its implementation (e.g. for hardware: integrated within the device's main SOC or separate co-processor, for software: drivers, libraries (if applicable), logical interfaces for encryption/decryption) and identification of any areas that are not encrypted (e.g. partition tables, etc.). The description should also include the data flow from the device's host interface to the device's non-volatile memory storing the data, information on those conditions in which the data bypasses the data encryption engine (e.g. read-write operations to an unencrypted Master Boot Record area). The description should also describe the device's initialisation, the encryption initialisation process, and at what moment the device enables the encryption.
- The process for destroying keys when they are no longer needed by including the type of storage location of all keys and the destruction method for that storage.

The Key Management Description shall include a diagram that provides the following:

- The diagram will include all of keys starting at the conditioning of authorisation data and continuing until the unwrapped DEK is reached. It shall also include any keys or values that contribute to the creation of elements in the chain but are not themselves elements in the chain. It must list the cryptographic strength of each key and illustrate how each key along the chain is protected with either Key Derivation or Key Wrapping (from the allowed options). The diagram should indicate the input used to derive or unwrap each key in the chain.
- A functional (block) diagram showing the main components (such as memories and processors) and the data path between, for hardware, the device's host interface and the device's persistent media storing the data, or for software, the initial steps needed to the activities the TOE performs to ensure it encrypts the storage device entirely when a user or administrator first provisions the product. The hardware encryption diagram shall show the location of the data encryption engine within the data path.
- The hardware encryption diagram shall show the location of the data encryption engine within the data path. The evaluator shall validate that the hardware encryption diagram contains enough detail showing the main components within the data path and that it clearly identifies the data encryption engine.

C. Glossary

Term	Meaning
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
Data Encryption Key (DEK)	A key used to encrypt data-at-rest.
Error State	The device has failed a self-test and could not reset
Key Chaining	The method of using multiple layers of encryption keys to protect data. A top layer key encrypts a lower layer key which encrypts the data; this method can have any number of layers.
Key Encryption Key (KEK)	A key used to encrypt other keys, such as DEKs or storage that contains keys.
Keying Material	A data item that is used in combination with other data in order to derive a cryptographic key (e.g. a passphrase, seed, or each of the values used in an xor combination).
Passphrase Authorisation Factor	A type of authorisation factor requiring the user to provide a secret set of characters to gain access.
Powered-Off State	The device has been shutdown.
Submask	A submask is a bit string used in a cryptographic key chaining process.
Target of Evaluation	A set of software, firmware and/or hardware possibly accompanied by guidance. [CC1]
TOE Security Functionality (TSF)	A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs. [CC1]
TSF Data	Data for the operation of the TSF upon which the enforcement of the requirements relies.

See [CC] for other Common Criteria abbreviations and terminology.

D.Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
AF	Authorisation factor
CA	Certificate Authority
CBC	Cipher Block Chaining
CCM	Counter with CBC-Message Authentication Code
cPP	Collaborative protection Profile
DEK	Data Encryption Key
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
HMAC	Keyed-Hash Message Authentication Code
IEEE	Institute of Electrical and Electronics Engineers
KDF	Key Derivation Function
KEK	Key Encryption Key
NIST	National Institute of Standards and Technology
MBR	Master Boot Record
PBKDF	Passphrase-Based Key Derivation Function
PP	Protection Profile
RBG	Random Bit Generator
RSA	Rivest Shamir Adleman Algorithm
SHA	Secure Hash Algorithm
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
USB	Universal Serial Bus
XTS	XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing