collaborative Protection Profile for USB Portable Storage Devices

Version 0.10

Acknowledgements

This collaborative Protection Profile (cPP) was developed by the USB international Technical Community (iTC) with representatives from industry, Government agencies, and Common Criteria Test Laboratories.

0. Preface

0.1 Objectives of Document

This document presents the Common Criteria (CC) collaborative Protection Profile (cPP) to express the security functional requirements (SFRs) and security assurance requirements (SARs) for a USB Portable Storage Device. The Evaluation Activities that specify the actions the evaluator performs to determine if a product satisfies the SFRs captured within this cPP are described in [SD].

0.2 Scope of Document

The scope of the cPP within the development and evaluation process is described in the Common Criteria for Information Technology Security Evaluation [CC]. In particular, a cPP defines the IT security requirements of a generic type of TOE and specifies the functional security measures to be offered by the TOE and the assurance measures to be applied during evaluation of the TOE to meet stated requirements [CC1, Appendix B].

0.3 Intended Readership

The target audiences of this cPP are developers, consumers, system integrators, evaluators and schemes.

Although the cPPs and Supporting Documents (SDs) may contain editorial errors, cPPs are recognized living documents and the iTCs are dedicated to ongoing updates and revisions. Please report any issues to the USB iTC.

0.4 Related Documents

Common Criteria[1]

[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017.

[CC2] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017.

[CC3] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.

[CEM] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, CCMB-2017-04-004, Version 3.1, Revision 5, April 2017.

[1] For details see http://www.commoncriteriaportal.org/

Other documents

[SD] Evaluation Activities for USB Portable Storage Device cPP, Version 1.0, ***[**Reference detail to be added when available]***

0.5 Revision History

Version	Date	Description
0.3	2015-07-30	Draft published for Public Review
0.4	2015-09-25	Addressed comments received during Public Review
0.5	2015-10-16	Addressed comments received during Public Review
0.6	2015-11-02	Addressed comments received during Public Review
0.7	2015-12-11	Added the FCS_RBG_EXT SFRs
0.8	2016-01-20	Addressed comments received from IPA (Japan).
0.9	2016-09-23	Updated Crypto SFRs based on draft CCDB Crypto WG work, plus other amendments for comments received. Remaining editorial notes marked as: ***[**(editorial note…)]***
0.10	2018-10-19	Updated DRBG SFRs based on input from CCDB Crypto WG, claim conformance to CC v1.3 Revision 5 (instead of Revision 4)

Contents

Figures / Tables

1. PP Introduction

1.1 PP Reference Identification

PP Reference: collaborative Protection Profile for USB Portable Storage Devices

PP Version: 0.10

PP Date: 2018-10-19

1.2 TOE Overview

This is a collaborative Protection Profile (cPP) whose Target of Evaluation (TOE) is a USB Portable Storage Device. It provides a minimal set of security requirements expected by all USB portable storage devices that target the mitigation of a set of defined threats. This baseline set of requirements will be built upon by future cPPs to provide an overall set of security requirements for USB portable storage devices. A USB Portable Storage Device is a portable storage device (hereafter referred to as "the device" or "the TOE") that provides a USB interface for connecting to a host computer.

The device employs cryptographic means to provide the necessary protection of user data. The strength of these cryptographic means lies in the quality of the algorithms, the modes chosen, and the key sizes used 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 to which the device is connected.

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 cryptanalytic effort which is beyond the scope of this cPP.

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).

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

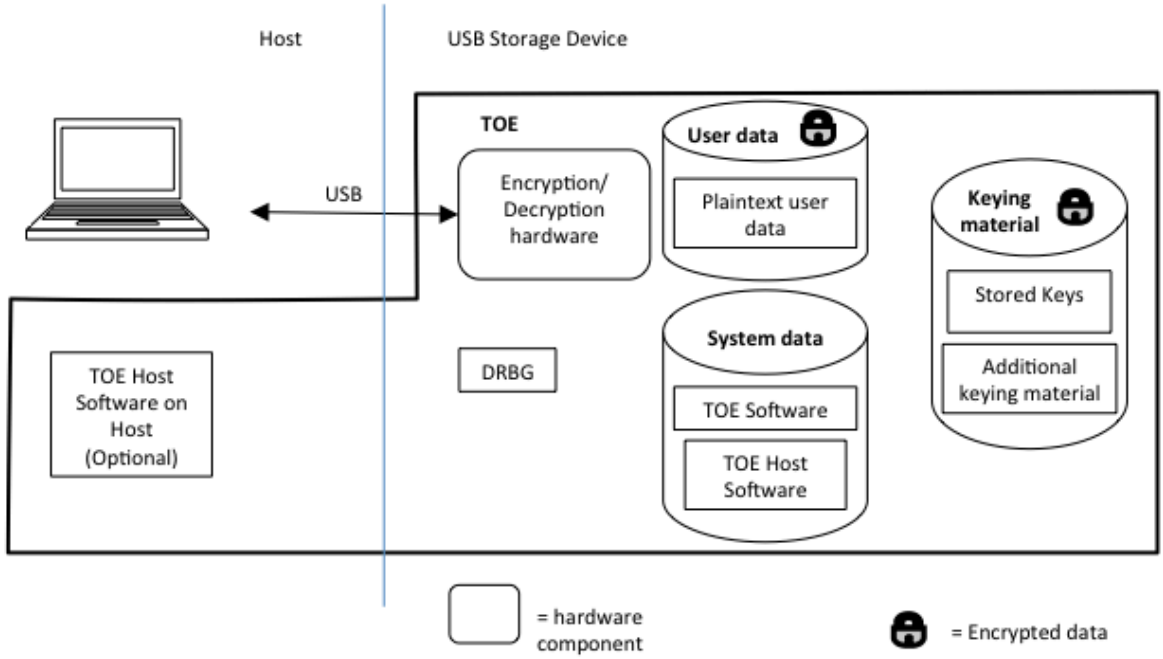


Figure 1: USB device boundaries, components and connections

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 using a suitable deterministic random bit generator (DRBG). The DEK is used to encrypt and decrypt user data stored on the device. To support the intention that a found or stolen device would present a cryptanalytic challenge to an attacker, the authorisation data (that ultimately gives access to the DEK and the user data) must be stored encrypted on the device.

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, firmware, patches, or configuration data. The TOE provides integrity protection of the system data and is responsible for ensuring this data cannot be modified by untrusted entities through the logical interface.

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

1.3 TOE Usage

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. It is also not suitable for more complex devices that may include memory media (such as smartphone, cameras or media players).

It is intended to be used for the following scenarios:

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

2. CC Conformance

As defined by the references [CC1], [CC2] and [CC3], this cPP:

- conforms to the requirements of Common Criteria v3.1, Revision 5.
- is Part 2 extended, Part 3 conformant
- does not claim conformance to any other PP.

The methodology applied for the PP evaluation is defined in [CEM]. This cPP satisfies the following Assurance Families: APE_CCL.1, APE_ECD.1, APE_INT.1, APE_OBJ.1, APE_REQ.1 and APE_SPD.1.

In order to be conformant to this cPP, a TOE must demonstrate Exact Conformance. Exact Conformance, as a subset of Strict Conformance as defined by the CC, is defined as the ST containing all of the requirements in section 5 (these are the mandatory requirements) of the this cPP, and potentially requirements from Appendix A (these are optional SFRs) or Appendix B (these are selection-based SFRs, some of which will be mandatory according to the selections made in other SFRs) of this cPP. While iteration is allowed, no additional requirements (from the CC parts 2 or 3, or definitions of extended components not already included in this cPP) are allowed to be included in the ST. Further, no requirements in section 5 of this cPP are allowed to be omitted.

3. Security Problem Definition

3.1 Threats

A threat consists of a threat agent, an asset and an adverse action of that threat agent on that asset.

Threat agents are unauthorised users, i.e. they do not possess valid authorisation factors. They are referred to here as "attackers". Attackers are typically characterised by a number of factors, such as expertise, available resources, and motivation, with motivation being linked directly to the value of the assets at stake. The device is resistant to an attacker possessing a basic attack potential.

The assets to be protected by the device are:

- the plaintext user data and keying material requiring confidentiality protection, and
- system data requiring integrity protection.

In general, some individual data items stored on the device will not be sensitive, but the cPP does not require distinction of security requirements at this more granular level: all user data and keying material is to be protected for confidentiality, and all system data is to be protected for integrity.

The underlying consequences of all threats considered in this cPP are that an attacker could retrieve plaintext user data or keying material, or could modify system data.

In the subsections below, the description of each threat is followed by a list of the names of the Security Functional Requirements that are intended to counter it, and a rationale paragraph that explains the way in which the SFRs (and objectives for the operational environment) combine to mitigate the threat. The SFR names refer to the requirements listed in section 5, and the objectives for the operational environment can be found in section 4.1.

3.1.1 T.UNAUTHORISED_USER_DATA_ACCESS

The primary threat to be addressed is the unauthorised disclosure of user data stored on a device. Attackers may attempt to use the logical interface to access plaintext data, or may connect the device to a host that provides raw access to the device content (e.g. to specified disk sectors or blocks).

Attackers may gain physical access to the device, and thus bypass the logical interface to obtain access to user data (perhaps by making physical modifications to the device to access its memory via their own physical connections, or the attacker might use equipment appropriate for the attack potential to read the contents of memory locations from their physical state).

Attackers may access user data that remains unprotected due to a failure that interrupts correct operation of the device. Attackers may look for unencrypted keying material giving them unauthorised access to user data.

[FPT_UDD_EXT.1, FCS_COP.1/UDE, FCS_CKM.1/DEK, FCS_CKM.1/Asymm, FTA_USB_EXT.1, FCS_KYC_EXT.1, FCS_CKM_EXT.5/KEK, FCS_SLT_EXT.1, FCS_RBG_EXT.1, FCS_RBG_EXT.2, FCS_RBG_EXT.3, FCS_RBG_EXT.4, FCS_CKM.4, FPT_TST.1, FPT_FLS.1]

Rationale:

- Protection against the unauthorised disclosure of user data stored on the device is provided by user data encryption:
  - FDP_UDD_EXT.1 ensures that all user data is encrypted without user intervention in accordance with FCS_COP.1/UDE. The DEK generation during provisioning is specified in FCS_CKM.1/DEK.
  - FTA_USB_EXT.1 ensures that the TSF requires the user to provide a valid passphrase before it allows access to user data on the TOE. FCS_KYC_EXT.1 ensures that the TSF maintains a chain of intermediary keys originating from the authorisation data submask, the output resulting from the conditioning function specified in FCS_CKM_EXT.5/KEK, and ending in the DEK. (See also the use of FCS_CKM.1/Asymm, FCS_SLT_EXT.1, FCS_RBG_EXT.1, FCS_RBG_EXT.2, FCS_RBG_EXT.3, FCS_RBG_EXT.4, and FCS_CKM.4 in the mitigation of T.KEYING_MATERIAL_COMPROMISE in section 3.1.3 for more description of how keys are protected on the device).
- Failures interrupting correct operation of the device are detected through FPT_TST.1, which enforces known answer tests of the cryptographic algorithms and firmware integrity tests during initial start-up. FPT_FLS.1 ensures that the TSF preserves a secure state when the start-up tests fail.

In addition, OE.LOST_DEVICE makes sure that the device is discarded if it may be suspected that attacker has tampered with the device (since such tampering might be used to give an attacker unauthorised access to data stored on the device after it is returned to the user).

3.1.2 T.UNAUTHORISED_SYSTEM_DATA_MODIFICATION

Attackers may modify system data stored on a device. Attackers may attempt to use the logical interface to modify system data, or may connect the device to a host that provides raw access to the device content (e.g. to specified disk sectors or blocks).

Note that since system data may include software that runs on the host, the threat of unauthorised modification of this software would also make the device a delivery mechanism for spreading malware[2].

[FDP_SDD_EXT.1, FPT_TUD_EXT.1, FPT_TUR_EXT.1]

Rationale:

- As required in FDP_SDD_EXT.1, system data is only updated by the results of normal TOE operation and the actions of authorised users, or via a trusted update mechanism

[2] Note that it is not primarily a responsibility of the TOE to prevent the spread of malware, and it provides no protective measures for user data (only for system data).

as described for T.UNAUTHORISED_UPDATE in section 3.1.5 below, based on FPT_TUD_EXT.1 and (optionally) FPT_TUR_EXT.1.

- OE.LOST_DEVICE makes sure that the device is discarded if it may be suspected that attacker has tampered with the device.
- OE.TRUSTED_HOST makes sure that the host computer is trusted, free of malware that could interfere with the correct operation of the device. Only authorised users have access to the host computer. This mitigates the risk of modification to system data when it is stored on the host.
- OE.TRUSTED_CONNECTION makes sure that communication between the host computer and the device is sufficiently protected to prevent disclosure of, or tampering with, user data, keying material or system data. This mitigates the risk of modification to system data while it is being transferred to the host.

3.1.3 T.KEYING_MATERIAL_COMPROMISE

Possession of any of the keys, authorisation data, random numbers or any other values that contribute to the creation of keys or authorisation data could allow an attacker to defeat the encryption. As part of a conservative approach to security, gaining access to keying material is considered to be of equal importance to gaining access to plaintext user data or system data itself. Attackers may look for keying material in unencrypted sectors of the drive, including in memory used to support power saving modes (in the TOE).

Alternatively an attacker might determine a key because of insufficient entropy used in its generation.

[FCS_CKM.1/Asymm, FPT_KYP_EXT.1, FCS_CKM.3[3], FCS_COP.1/KeyEnc, FCS_KYC_EXT.1, FCS_CKM_EXT.5/KEK, FCS_CKM_EXT.5/Chain, FCS_SLT_EXT.1, FCS_RBG_EXT.1, FCS_RBG_EXT.2, FCS_RBG_EXT.3, FCS_RBG_EXT.4, FCS_CKM.4]

Rationale:

- Protection against the unauthorised disclosure of any keys, authorisation data, random numbers or any other values that contribute to the creation of keys or authorisation data is provided by encryption:
  - FPT_KYP_EXT.1 ensures that keys and keying material are stored in non-volatile memory only when wrapped, as specified in FCS_CKM.3 or encrypted, as specified in FCS_COP.1/KeyEnc.
  - FCS_KYC_EXT.1 ensures that the TSF maintains a chain of intermediary keys originating from the authorisation data submask, the output resulting from the conditioning function specified in FCS_CKM_EXT.5/KEK, and ending in the DEK. Since the beginning of the chain is the authorisation data, and this is not stored in plaintext on the device (Application Note 19 makes clear that FPT_KYP_EXT.1 applies to plaintext authorisation data), an attacker who has

[3] FCS_CKM.3 is mandatory for the wrapping of the DEK with the KEK, but may also be used at other points in a product's key chain (see the reference key model in section 5.3). The different uses are not distinguished for the purposes of the rationale, as both are part of making sure the relevant keys are adequately protected.

possession of the device cannot access the keys without separately obtaining the authorisation data. FCS_KYC_EXT.1 allows multiple methods to be used for the ultimate wrapping or derivation of the DEK: FCS_CKM.1/Asymm, FCS_CKM_EXT.5/KEK, FCS_CKM_EXT.5/Chain, FCS_CKM.3 and FCS_COP.1/KeyEnc.

- FCS_SLT_EXT.1 ensures that keying material is generated with sufficient and effective strength. FCS_RBG_EXT.1, FCS_RBG_EXT.2, FCS_RBG_EXT.3, FCS_RBG_EXT.4, specify requirements on the random number generator.
- FCS_CKM.4 ensures proper deletion of cryptographic keys and keying material.

3.1.4 T.AUTHORISATION_GUESSING

Attackers may mount an exhaustive search (brute force) attack against the device to determine authorisation factors to gain unauthorised access to the user data stored on the device.

[FIA_AFL.1, FCS_CKM.4, FPT_VAL_EXT.1, FIA_PPS_EXT.1, FIA_PPS_EXT.2/num, FIA_PPS_EXT.2/alph, FIA_UAU.7]

Rationale:

- FIA_PPS_EXT.1 defines the method of supplying passphrases to the TOE, and identifies the passphrase composition (including minimum and maximum length requirements) in either FIA_PPS_EXT.2/num or FIA_PPS_EXT.2/alph according to the nature of the interface.
- Where the TOE itself provides an interface for entering the authorisation data, then the feedback to the user as characters are entered must not expose the authorisation data, as required by FIA_UAU.7.
- FIA_AFL.1 ensures that the user data is made permanently unavailable by deleting the current DEK, as specified in FCS_CKM.4, as soon as a threshold number of consecutive authorisation failures is reached. An authorisation attempt failure is defined as a failed submask validation attempts as in FPT_VAL_EXT.1.

3.1.5 T.UNAUTHORISED_UPDATE

Attackers may attempt to perform an unauthorised update of the product, which compromises the security features of the device. Poorly chosen update protocols, signature generation/verification algorithms and parameters may allow attackers to install software and/or firmware that bypasses the intended security features and provides them unauthorised access to user data and keying material or allows modification of system data.

This threat includes attempts to make an unauthorised rollback of updates so that they are no longer applied, or to replay an old, valid update message containing a superseded update (which might allow a known vulnerability to be exploited).

[FPT_TUD_EXT.1, FCS_COP.1/SigVer, FCS_COP.1/Hash, FCS_COP.1/HMAC, FPT_TUR_EXT.1, FMT_SMF.1]

Rationale:

This threat is applicable only if the device provides secure firware/software update functionality. Any device that provides such functionality must comply with the following requirements:

- FPT_TUD_EXT.1 provides authorised users with the ability to query the current version of the TOE firmware/software, initiate updates, and verify the updates using a digital signature mechanism (as specified in FCS_COP.1/SigVer), published hash (as specified in FCS_COP.1/Hash) or keyed hash (as specified in FCS_COP.1/HMAC) prior to installing those updates.
- A further level of protection may optionally be provided if the TOE chooses to implement protection against rollback of updates as in FPT_TUR_EXT.1. If the TOE does not provide the ability to update system data then neither of these SFRs is required.
- FMT_SMF.1 ensures that the TSF provides management functionality to query the current version of the TOE firmware/software and initiate updates of the TOE firmware/software.

3.2 Assumptions

3.2.1 A.USER_GUIDANCE

Users will be instructed in the secure use of the device.

[OE.USER_GUIDANCE]

Rationale:

- OE.USER_GUIDANCE ensures that users take the appropriate procedural measures to safeguard the device and the data that it contains.

3.2.2 A.LOST_DEVICE

The device shall be discarded in the event that the device is left unattended, lost or stolen and later recovered, if it may be suspected that an attacker has tampered with the device.

[OE.LOST_DEVICE]

Rationale:

- OE.LOST_DEVICE identifies this assumption as an obligation on the operational environment.

3.2.3 A.TRUSTED_HOST

The host computer is trusted, free of malware that could interfere with the correct operation of the device. Only authorised users have access to the host computers.

[OE.TRUSTED_HOST]

Rationale:

- OE.TRUSTED_HOST identifies this assumption as an obligation on the operational environment.

3.2.4 A.TRUSTED_CONNECTION

Communication between the host computer and the device is sufficiently protected to prevent disclosure of, or tampering with, user data, keying material or system data.

[OE.TRUSTED_CONNECTION]

Rationale:

- OE.TRUSTED_CONNECTION identifies this assumption as an obligation on the operational environment.

3.3 Organizational Security Policy

3.3.1 P.NO_STORE

It is not possible to reconstruct the keys that protect user data from data persistently stored on the TOE. (This means that complete reference authorisation data is not persistently stored on the TOE.)

[FCS_KYC_EXT.1, FCS_CKM_EXT.5/KEK, FPT_KYP_EXT.1]

Rationale:

- FCS_KYC_EXT.1 ensures that the TSF maintains a chain of intermediary keys originating from the authorisation data submask, the output resulting from the conditioning function specified in FCS_CKM_EXT.5/KEK, and ending in the DEK.
- FPT_KYP_EXT.1 requires that keys and keying material are not stored in plaintext form in non-volatile memory (except possibly for items that are not part of the key chain for FCS_KYC_EXT.1, or where the plaintext items are present only for initial provisioning of the device and therefore will not be used to protect user data (and where OE.USER_GUIDANCE will advise users on the procedural requirements for secure provisioning)).

3.3.2 P.RECOVERY (optional[4])

If the TOE provides any mechanism to enable data recovery in the event of loss of authorisation data then it must allow this mechanism to be disabled, such that it cannot be re-enabled by an unauthorised user.

[FMT_SMF.1]

Rationale:

- FMT_SMF.1 ensures that (if the device provides a recovery mechanism) the TSF provides management functionality to disable data recovery, and ensures that a new DEK is generated if the mechanism is subesequently enabled.

3.3.3 P.CRYPTO

The cryptographic algorithms, key lengths and modes used shall be in conformance with the requirements of the national authorised cryptographic authority.

[FCS_CKM.1/DEK, FCS_CKM.1/Asymm, FCS_CKM.3, FCS_CKM.4, FCS_CKM_EXT.5/KEK, FCS_CKM_EXT.5/Chain, FCS_RBG_EXT.1, FCS_COP.1/UDE, FCS_COP.1/KeyEnc, FCS_COP.1/Hash, FCS_COP.1/HMAC, FCS_COP.1/SigVer, FCS_SLT_EXT.1]

Rationale:

- The cryptographic SFRs identified above are adopted from an internationally harmonised set of cryptographic SFRs. ***[**Currently the SFRs are based on draft output from the CCDB Crypto WG. This text will be updated to reference specific output from the CCDB Crypto WG when available.]***

3.3.4 P.AUTH_CHANGE

The TOE 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.

[FTA_USB_EXT.1, FMT_SMF.1]

Rationale:

- FTA_USB_EXT.1 ensures that users need to re-authorise when they change the value of the authorisation data.

[4] This policy is optional in the sense that if the TOE does not provide a data recovery mechanism then the policy, and the corresponding SFR is not needed. In this case the ST author should include the policy and explain that the relevant SFR is not required. However, if the TOE provides a data recovery mechanism then the policy and its corresponding SFR must be included in the ST.

- FMT_SMF.1 ensures that the TSF provides management functionality to change the value of the authorisation data.

3.3.5 P.FULL_ENCRYPTION

Only encrypted data storage shall be available to store user data (i.e. no unencrypted storage is available to users).

[FDP_UDD_EXT.1, FCS_COP.1/UDE]

Rationale:

- FDP_UDD_EXT.1 ensures that the TSF encrypts all user data without user intervention, as specified in FCS_COP.1/UDE.

3.3.6 P.NO_BOOT

It shall not be possible to boot from the TOE in its evaluated configuration.

[OE.NO_BOOT]

Rationale:

- The operational environment ensures that it is not possible to boot from the TOE in its evaluated configuration (OE.NO_BOOT).

4. Security Objectives

4.1 Security Objectives for the Operational Environment

4.1.1 OE.USER_GUIDANCE

Users will be instructed in the secure use of the device. This guidance shall include a reminder to generate keys in a suitably secure environment that mitigates the risks of any sort of interference with the key generation process, and that ensures that any provisioning keys used by the device do not threaten the security of user data.

4.1.2 OE.LOST_DEVICE

The device is discarded in the event that the device is left unattended, lost or stolen and later recovered, if it may be suspected that an attacker has tampered with the device.

4.1.3 OE.TRUSTED_HOST

The host computer is trusted, free of malware that could interfere with the correct operation of the device. Only authorized users have access to the host computers.

4.1.4 OE.TRUSTED_CONNECTION

Communication between the host computer and the device is sufficiently protected to prevent disclosure of, or tampering with, user data, keying material or system data.

4.1.5 OE.NO_BOOT

The operational environment ensures that the TOE shall not be used as a boot device.

5. Security Functional Requirements

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. The baseline requirements (those that must be performed by the TOE) are contained in this chapter of the cPP. Additionally, there are two other types of requirements specified in Appendix A and Appendix B.

The first type (in Appendix A) is requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this cPP. The second type (in Appendix B) is requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements in that appendix will need to be included in the body of the ST (e.g., cryptographic protocols selected in a trusted channel requirement).

For example, the requirements that support digital signature verification in support of trusted updates (section B.1.3.4) 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). Similarly, the key encryption requirements in section B.1.3.1 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 5.3.3.1). However, the key wrapping requirement in section 5.3.1.2 is 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.

5.1 Conventions

The conventions used in description of the SFRs are as follows:

- Assignments, refinement and selections made by cPP author: Indicated with **bold text** and ~~strikethroughs~~, if necessary;
- Assignments and selections that need to be made by the ST writer: Indicated with ***bold, italic text***;
- Iteration: Indicated by appending an identifier at the end of the SFR name, e.g. FCS_COP.1/SigVer.

Extended SFRs are identified by having a label ‘EXT’ at the end of the SFR name (before any iteration identifier).

In some of the SFRs below, various assignments and/or selections have dependent choices: i.e. the choice in one selection determines or constrains the choices in other assignments/selections. In some of these cases the SFR is followed by a table in which the rows define the allowed sets of choices for completing the SFR. This means that for a Security Target to be judged conformant to the Protection Profile it must include the SFR with selections defined as one or more complete rows from the table. The preferred approach to including multiple rows of a table in an ST is to iterate the SFR for each row.

5.2 SFR Architecture

Figure 2, Figure 3, Figure 4 and Figure 5 give a graphical presentation of the connections between the Security Functional Requirements in sections 5.3-5.8, Appendix A and Appendix B, and the underlying functional areas and operations that the TOE provides[5]. The diagrams provide a context for SFRs that relates to their use in the TOE, whereas other sections define the SFRs grouped by the abstract class and family groupings in [CC2].

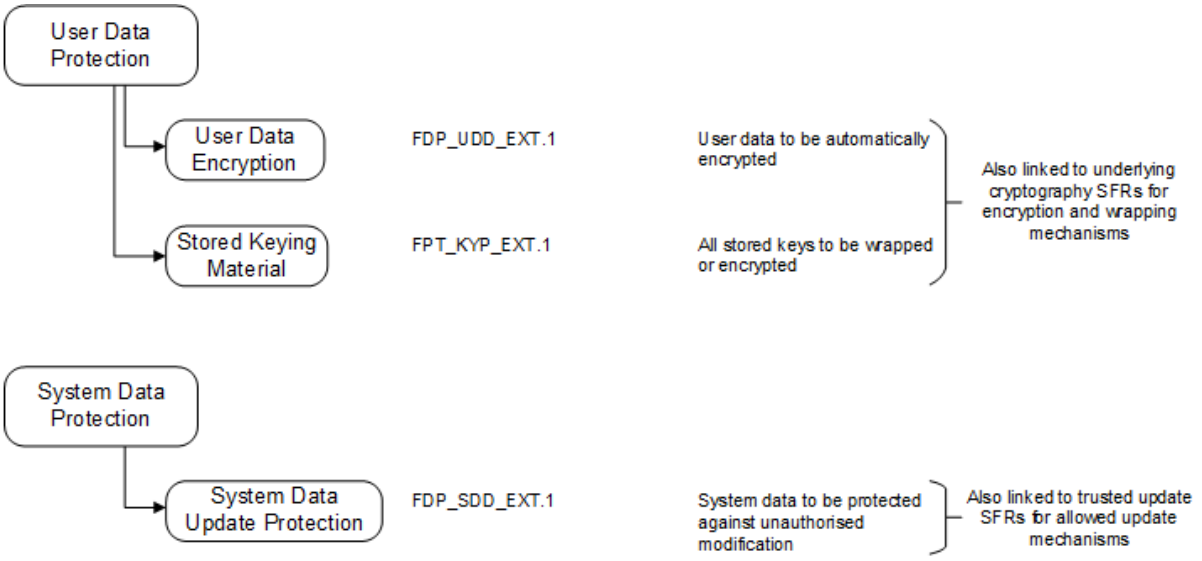


Figure 2: User Data Protection and System Data Protection SFR Architecture

[5] Note that SFRs in Appendix A are marked as "(optional)" in the figures; SFRs in Appendix B are marked "(selection)" (short for "selection-based")

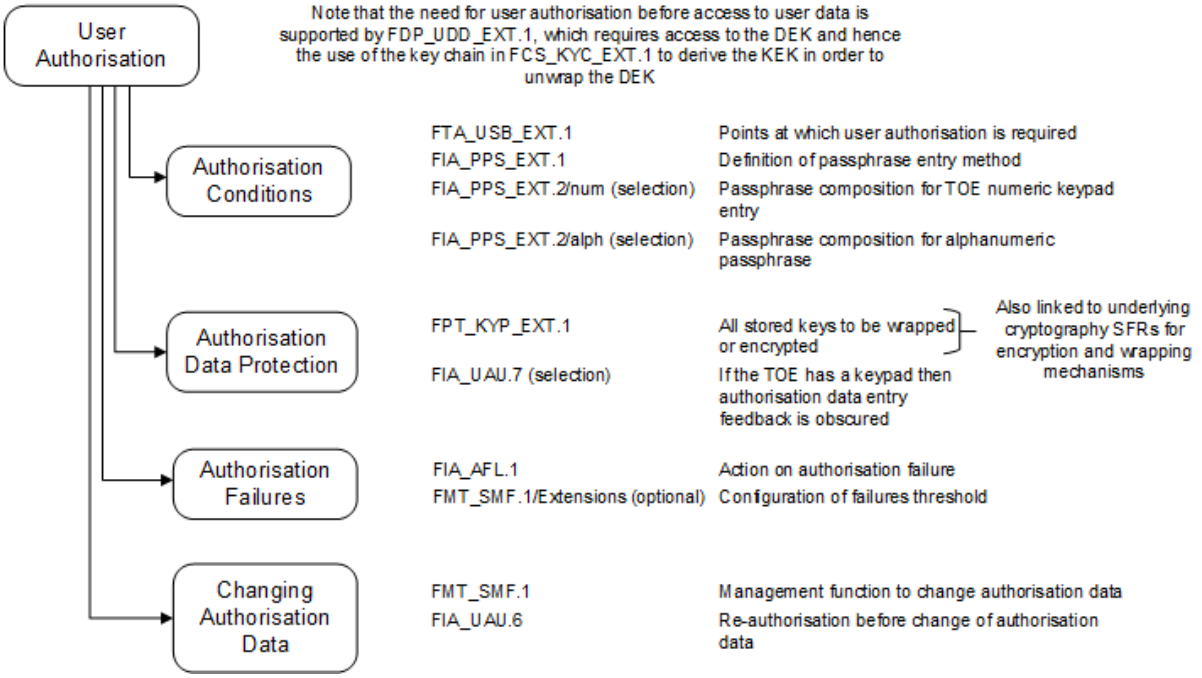


Figure 3: User Authorisation SFR Architecture

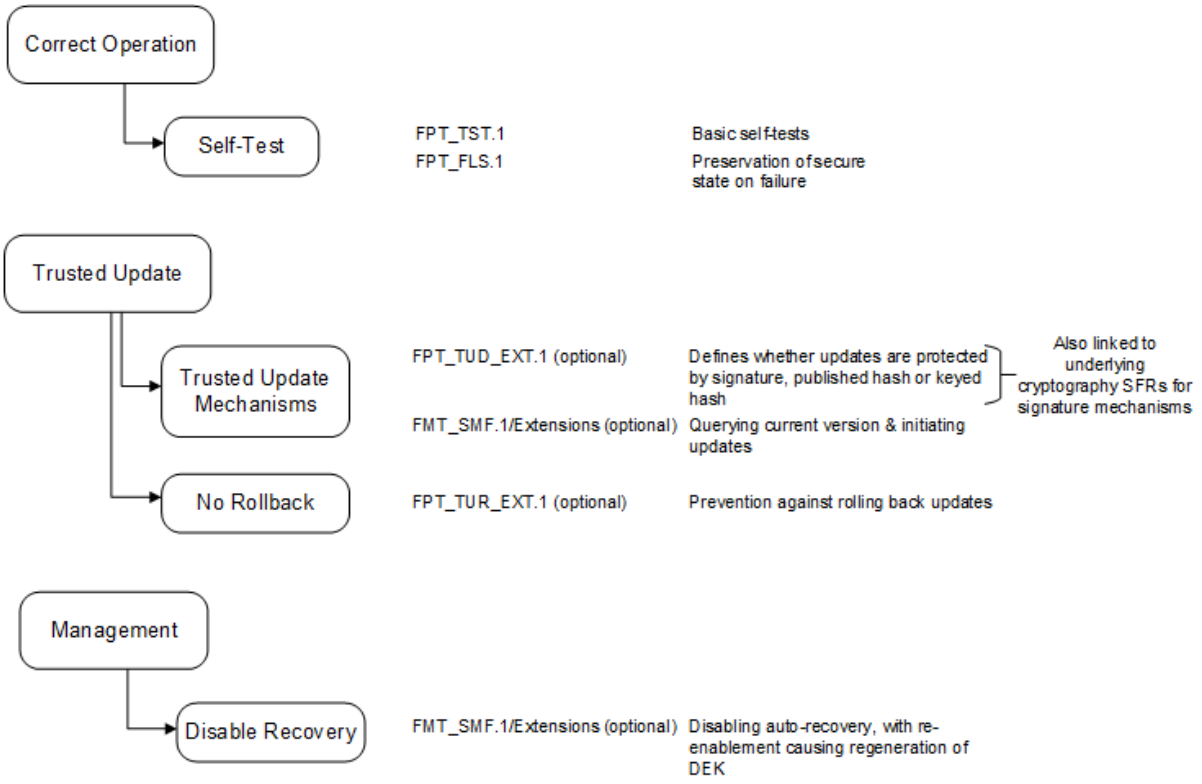


Figure 4: Correct Operation, Trusted Update, and Management SFR Architecture

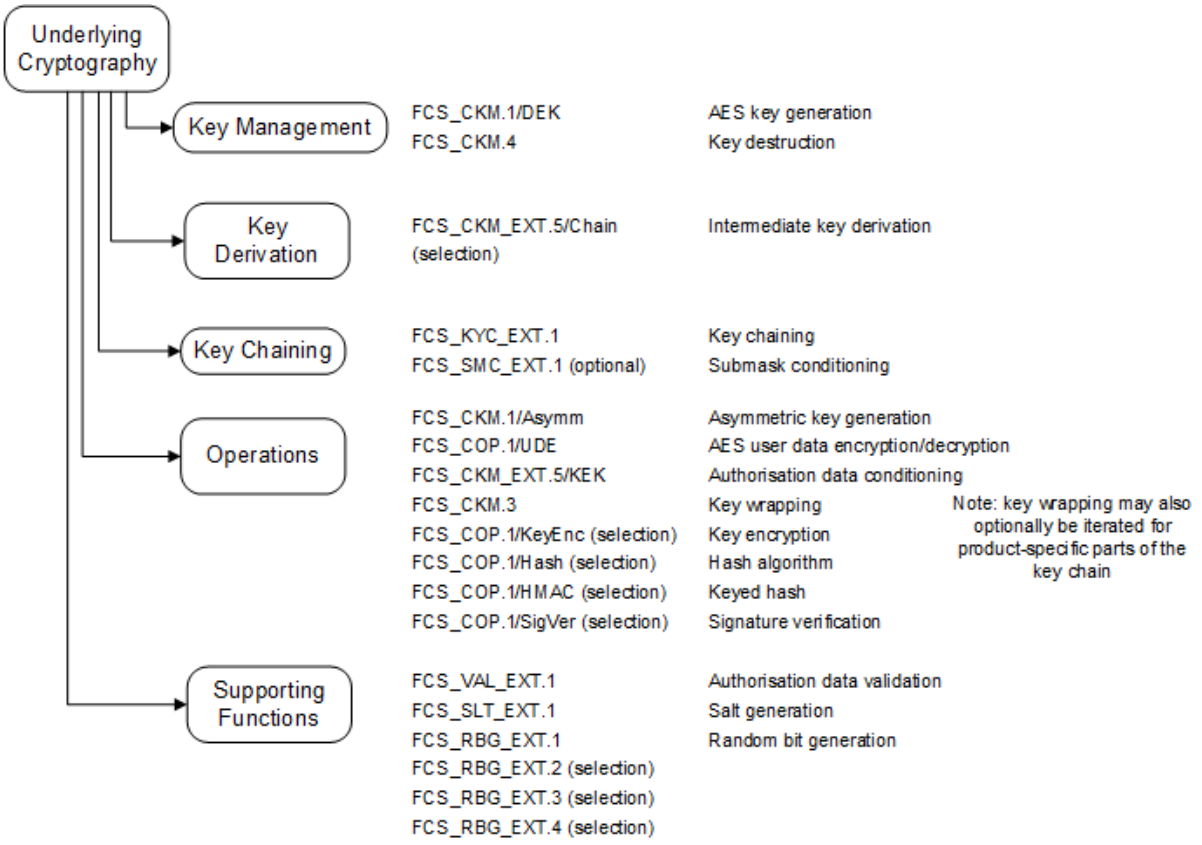


Figure 5: Underlying Cryptography SFR Architecture

5.3 Cryptographic Support (FCS)

This class of SFRs describes the cryptographic requirements for the TOE. As shown in section 5.2, there are cryptographic requirements underlying and supporting a number of the main requirement areas for the TOE. The main area in which the cryptographic mechanisms are involved is the encryption of user data using a Data Encryption Key (DEK) that is accessed by using a Key Encryption Key (KEK) that is derived from the authorisation data entered by a user. Obtaining the correct DEK in this way provides the user authorisation method: if the wrong authorisation data is entered then the chain of operations to obtain the KEK will not return the correct value, and hence the DEK will not be obtained. The chaining from the authorisation data to the DEK is therefore of great importance for the TOE and is described further below in the 'Reference Key Model'.

The key architecture used by different products 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 D.2.

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 (5.3.1.1) and then used to encrypt user data (section 5.3.2.1). 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 5.3.1.2). A device may use more than one DEK, and/or more than one KEK.

The reference key model is as follows:

- Generation of the DEK

  The reference key model for generation of the Data Encryption Key (DEK) described in section 5.3.1.1 is simply that the DEK is generated on the USB device using a suitable deterministic random bit generator as described in 5.3.5.1 that is seeded according to requirements in B.1.4.1, B.1.4.2 and B.1.4.3.

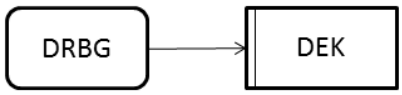


Figure 6: Reference Key Model for DEK generation

- Derivation of the KEK

  The KEK is derived from a chain of operations that starts from authorisation data (such as a passphrase) obtained from authorisation factors. 5.5.2.1 defines the method of supplying passphrases to the TOE. As the chain is followed, submasks may be validated at various stages as described in section 5.6.4.1. The number of stages in the chain is left to the implementation (this is the meaning of the dotted lines in Figure 7), but it must begin with the conditioned authorisation data (see section 5.3.1.4) and end in the creation of the KEK that wraps the DEK (as in section 5.3.1.2), in a manner that will preserve a security strength that is sufficient to protect the DEK.

Figure 7: Reference Key Model for KEK derivation

- Wrapping of the DEK with the KEK

A DEK is always wrapped by a KEK when it is stored in non-volatile memory as described in 5.3.1.2. The KEK may itself be wrapped by other keys or may be derived from the authorisation data – see section 5.3.1.4). 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.

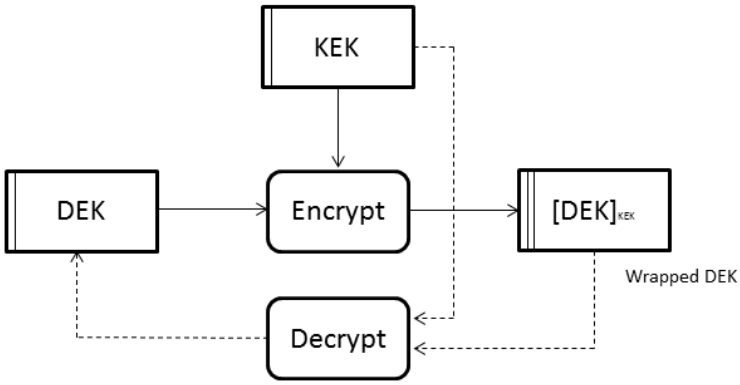


Figure 8: DEK wrapping with KEK

The KEK is an ephemeral value as shown in Figure 8, but in some implementations may itself be stored in an encrypted form (section B.1.3.1) or wrapped form (as described in section B.1.1.2).

- Encryption of data under the DEK

The device encrypts all user data that is stored on the device under the DEK as described in section 5.3.2.1, without user intervention (section 5.4.1.1).

- Destruction of keys

The device destroys all cryptographic keys and keying material (whether in plaintext or encrypted form) using methods according to the type of memory where the key has been held as specified in section 5.3.1.3.

- Supporting cryptographic functions used to implement other device functionality

The device fulfils cryptographic salt generation requirements as described in section 5.3.4.1. The device performs cryptographic hashing as described in section B.1.3.2 as well as keyed-message authentication as described in B.1.3.3.

5.3.1 Cryptographic Key Management (FCS_CKM)

5.3.1.1 **FCS_CKM.1/DEK Key generation (AES Data Encryption Key)**

FCS_CKM.1.1/DEK The TSF shall generate cryptographic keys **[selection: *key name*]**[6] in accordance with a specified cryptographic key generation algorithm **[selection: *cryptographic key generation algorithm*]** and specified cryptographic key sizes **[selection: *key sizes*]** that meet the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_CKM.1.

Identifier	key name	cryptographic key generation algorithm	key sizes	list of standards
DEK1	DEK	Direct[7] Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	[selection: 128 bit, 256 bit, 512 bit]	NIST SP 800-133 (Section 7.1) with ISO 18031 as an approved RBG in addition to those in NIST SP 800-133 (Section 5).

Application Note 1

The purpose of this requirement is to explain DEK generation during provisioning. There is no option to import a DEK. The DEK must be generated on the USB device using a suitable random bit generator as specified in FCS_RBG_EXT.1.

Application Note 2

Key Identifier #1 may be used for instantiations using Camillia as it has the same requirements as AES for key generation algorithm, key size and standards.

Application Note 3

The selection of key size 512 bits is for the case of XTS-AES using AES-256. In the case of XTS-AES for both AES-128 and AES-256, the developer is expected to ensure that the full key is generated using direct generation from the RBG as in NIST SP800-133 section 5.

[6] Refinement of "cryptographic keys" as an assignment of key name which stands for the algorithm type and purpose if any.

[7] Here, "direct generation" is as described in NIST SP 800-133 section 7.1 where the output of the RBG is XORed with V, where V is a bit string of the same length as the RBG output and the value of V is determined in a manner that is independent of the value of the RBG output (and vice-versa).

5.3.1.2 **FCS_CKM.3/DEK Cryptographic Key Access (Key Wrapping)**

FCS_CKM.3.1/DEK The TSF shall perform **[selection: *type of cryptographic key access*]** in accordance with a specified cryptographic key access method **[selection: *cryptographic key access method*]** that meets the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_CKM.3/DEK.

Identifier	type of cryptographic key access	cryptographic key access method	list of standards
KW1	Key wrapping	KWP[8] based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), NIST SP 800-38F, sec. 6.3 (KWP) ISO/IEC 19772, clause 7 (Key wrap)
KW2	Key wrapping	KW based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), NIST SP 800-38F, sec 6.2 (KW)
KW3	Key wrapping	GCM based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), ISO/IEC 19772, clause 11 (GCM)
KW4	Key wrapping	CCM based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), ISO/IEC 19772, clause 8 (CCM)

Application Note 4

This requirement shall be used to specify how the KEK is used to wrap the DEK.This SFR will always be needed in an ST in order to capture the last stage of the key chain in which the DEK is wrapped by the KEK (see Figure 8), but the SFR may also be iterated using the version

[8] Key Wrap with Padding

FCS_CKM.3/Chain in Appendix B.1.1.2 if needed to cover other use in the product-specific part of the key chain (see Figure 7 and FCS_KYC_EXT.1 in section 5.3.3.1).

Application Note 5

Key wrapping mechanisms use always authenticated-encryption modes.

5.3.1.3 **FCS_CKM.4 Cryptographic key destruction**

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method:

- **For volatile memory, the destruction shall be executed by a [selection: *single overwrite consisting of [selection: a pseudo-random pattern using the TSF's RBG, zeroes, ones, a new value of the key, [assignment: a value that does not contain any sensitive data]], removal of power to the memory, destruction of reference to the key directly followed by a request for garbage collection*];**

- **For non-volatile memory [selection:**

  - ***that employs a wear-leveling algorithm: the destruction shall be executed by a [selection: single overwrite consisting of zeroes, single overwrite consisting of ones, overwrite with a new value of a key, single overwrite consisting of [assignment: some value that does not contain any sensitive data], block erase];***

  - ***that does not employ a wear-leveling algorithm: the destruction shall be executed by a [selection: [selection: single, [assignment: number of passes]-pass] overwrite consisting of zeros followed by a read-verify, [selection: single, [assignment: number of passes]-pass] overwrite consisting of ones followed by a read-verify, overwrite with a new value of a key followed by a read-verify, [selection: single, [assignment: number of passes]-pass] overwrite consisting of [assignment: a value that does not contain any sensitive data] followed by a read-verify, block erase]***

    If the read-verification of the overwritten data fails, then the process shall be repeated again up to [assignment: number of times to attempt overwrite] times, whereupon an error is returned.

that meets the following: **No Standard**.

Application Note 6

The term "cryptographic keys" in this SFR includes the authorisation data that is the entry point to the key chain and all other cryptographic keys and keying material (whether in plaintext or encrypted form) – cf. 'Destruction of Keys' in the reference key model in section 5.3.

In the case of volatile memory, the selection "destruction of reference to the key directly followed by a request for garbage collection" is used in a situation where the TSF cannot address the specific physical memory locations holding the data to be erased and therefore

relies on addressing logical addresses (which frees the relevant physical addresses holding the old data) and then requesting the platform to ensure that the data in the physical addresses is no longer available for reading (i.e. the "garbage collection" referred to in the SFR text).

Guidance documentation for the TOE requires users not to allow the TOE to leave the user's control while a session is active (and hence while the DEK is likely to be in plaintext in volatile memory).

The selection for destruction of data in non-volatile memory includes block erase as an option, and this option applies only to flash memory. A block erase does not require a read verify, since the mappings of logical addresses to the erased memory locations are erased as well as the data itself.

Where different destruction methods are used for different data and/or different destruction situations then the different methods and the data/situations they apply to (e.g. different points in time, or power-loss situations) are described in the TSS (and the ST may use separate iterations of the SFR to aid clarity). The TSS includes a table describing all relevant keys and keying material (including authorisation data) used in the implementation of the SFRs, stating the source of the data, all memory types in which the data is stored (covering storage both during and outside of a session, and both plaintext and non-plaintext forms of the data), and the applicable destruction method in each case.

Some selections allow assignment of "a value that does not contain any sensitive data". This means that the TOE uses some specified data not drawn from an RBG meeting FCS_RBG_EXT requirements, and not being any of the particular values listed as other selection options. The point of the phrase "does not contain any sensitive data" is to ensure that the overwritten data is carefully selected, and not taken from a general pool that might contain current or residual data (e.g. user data or intermediate key chain values) that itself requires confidentiality protection.

This SFR does not apply to the public component of asymmetric key pairs.

Application Note 7

Cryptographic keys, including intermediate keys and keying material that are no longer needed are destroyed in volatile memory by using one of these approved methods. In these cases, the destruction method conforms to one of methods specified in this requirement. Cryptographic Erase is considered a well defined term for the destruction of key information. Some solutions support write access to media locations where keys are stored, thus allow for destruction of cryptographic keys via direct overwrites of key and keying material data. In other cases storage virtualization techniques on system and/or device level could result in multiple copies of key data and/or the underlying media technology does not support direct overwrites of locations where key data are stored. Note that one time programmable memories are excluded.

5.3.1.4 **FCS_CKM_EXT.5/KEK Cryptographic Key Derivation (Cryptographic authorisation data conditioning)**

FCS_CKM_EXT.5.1/KEK The TSF shall derive cryptographic keys **[assignment: *key type*]** from **[selection: *input parameters*]** in accordance with a specified cryptographic key derivation algorithm **[selection: *key derivation algorithm*]** and specified cryptographic key sizes **[selection: *key sizes*]** that meet the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_CKM_EXT.5/KEK.

Identifier	key type	input parameters	key derivation algorithm	key sizes	list of standards
1	Authorization Factor Submask	Password	HMAC-[selection: SHA-256, SHA-512], with [assignment: positive integer of 1000 or more] iterations	[selection: 128, 256] bits	NIST SP 800-132

Application Note 8

For identifier 1, 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.

Application Note 9

The output resulting from the key derivation function shall be at least the same length (in number of bits) as the DEK.

5.3.2 Cryptographic Operation (FCS_COP)

5.3.2.1 **FCS_COP.1/UDE Cryptographic operation (AES User Data Encryption/ Decryption)**

FCS_COP.1.1/UDE The TSF shall perform **user data encryption/decryption** in accordance with a specified cryptographic algorithm **[selection: *cryptographic algorithm*]** and cryptographic key sizes **[selection: *cryptographic key sizes*]** that meet the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/UDE.

Identifier	cryptographic algorithm	key sizes	list of standards
UDE1	AES in CBC mode with non-repeating and unpredictable IVs	[selection: 128 bits, 256 bits]	ISO 18033-3 (AES) ISO 10116 (CBC)
UDE2	AES in CCM mode with unpredictable, non-repeating nonce, minimum size of 64 bits	[selection: 128 bits, 256 bits]	ISO 18033-3 (AES) ISO 19772, sec. 8 (CCM) NIST SP800-38C
UDE3	AES in GCM mode with non-repeating IVs IV length must be equal to 96 bits; the deterministic IV construction method [SP800-38D, Section 8.2.1] must be used; the MAC length must be one of the values 96, 104, 112, 120, and 128 bits.	[selection: 128 bits, 256 bits]	ISO 18033-3 (AES) ISO 19772, sec.11 (GCM) NIST SP800-38D
UDE4	AES in XTS mode with unique [selection: consecutive non-negative integers starting at an arbitrary non-negative integer, data unit sequence numbers] tweak values	[selection: 256 bits, 512 bits]	ISO 18033-3 (AES) [selection: IEEE 1619, NIST SP800-38E] (XTS)
UDE5	Camellia in CBC mode with non-repeating and unpredictable IVs	[selection: 128 bits, 256 bits]	ISO 18033-3 (Camellia) ISO 10116 (CBC)
UDE6	Camellia in CCM mode with unpredictable, non-repeating nonce, minimum size of 64 bits	[selection: 128 bits, 256 bits]	ISO 18033-3 (Camellia) ISO 19772, sec. 8 (CCM) SP800-38C
UDE7	Camellia in GCM mode with non-repeating IVs	[selection: 128 bits, 256 bits]	ISO 18033-3 (Camellia) ISO 19772, sec.11 (GCM)

Identifier	cryptographic algorithm	key sizes	list of standards
	the IV length must be equal to 96 bits; the deterministic IV construction method [SP800-38D, Section 8.2.1] must be used; the MAC length t must be one of the values 96, 104, 112, 120, and 128 bits.		NIST SP800-38D
UDE8	Camellia in XTS mode with unique [selection: consecutive non-negative integers starting at an arbitrary non-negative integer, data unit sequence numbers] tweak values	[selection: 256 bits, 512 bits]	ISO 18033-3 (Camellia) [selection: IEEE 1619, SP800-38E] (XTS)

5.3.3 Cryptographic Key Chaining (FCS_KYC)

5.3.3.1 **FCS_KYC_EXT.1 Key Chaining**

FCS_KYC_EXT.1.1 The TSF shall maintain a chain of intermediary keys originating from the authorisation data submask and ending in the DEK using the following method(s):

- **authorisation factor submask as specified in FCS_CKM_EXT.5/KEK (identifier 1)**
- **key wrapping as specified in FCS_CKM.3/DEK**

[selection:

- ***intermediate key generation as specified in FCS_CKM_EXT.5/Chain (identifiers 2-5)***
- ***key encryption as specified in FCS_COP.1/KeyEnc***
- ***key wrapping as specified in FCS_CKM.3/Chain***
- ***no others*****]**

resulting in a key size of **[selection:** ***128 bits or 256 bits*****].**

Application Note 10

Key chaining is the method of using multiple layers of encryption keys to ultimately secure the protected user data on the TOE. The number of intermediate keys will vary. This applies to all keys that contribute to the ultimate wrapping or derivation of the DEK.

The ST Authors select appropriate iterations of other FCS requirements to specify each step in the key chain: it is allowable for an implementation to use multiple methods. One instance of the key wrapping function FCS_CKM.3/DEK will always be mandatory in order to wrap the DEK with the KEK as in Figure 8. However the first selection in FCS_KYC_EXT.1.1 also allows further iterations to be specified in case different key wrapping functions are used at other points in the product-specific key chaining shown in Figure 7.

FCS_CKM.1./Asymm is also included in Appendix B and is included if the key chain requires use of asymmetric encryption/decryption keys generated on the TOE.

The method the TOE uses to chain keys and manage/protect them shall be described in the Key Management and Data Storage Description; see Appendix D.2 for more information.

If 'no others' is applicable in the first selection then the ST author may simply omit the 3rd bullet in this list.

5.3.4 Salt Generation (FCS_SLT_EXT)

5.3.4.1 FCS_SLT_EXT.1 Cryptographic Salt Generation

FCS_SLT_EXT.1.1 The TSF shall only use salts that are generated **by an RBG in the TSF as specified in FCS_RBG_EXT.1**.

5.3.5 Random Bit Generation (FCS_RBG_EXT)

5.3.5.1 FCS_RBG_EXT.1 Random Bit Generation (RBG)

FCS_RBG_EXT.1.1 The TSF shall perform deterministic random bit generation services using **[selection:**

- ***Hash_DRBG [selection: SHA-1, SHA-224, SHA-512/224, SHA-256, SHA-512/256, SHA-384, SHA-512];***
- ***HMAC_DRBG [selection: SHA-1, SHA-224, SHA-512/224, SHA-256, SHA-512/256, SHA-384, SHA-512],***
- ***CTR_DRBG [selection: AES-128, AES-192, AES-256]*****]**

in accordance with **[selection:**

- ***ISO/IEC 18031:2011, Section [selection: C.2.2 Hash_DRBG, C.2.3 HMAC_DRBG, C.3.2 CTR_DRBG];***
- ***NIST SP 800-90A, Section [selection: 10.1.1 Hash_DRBG, 10.1.2 HMAC_DRBG,10.2.1 CTR_DRBG]*****]**

after initialization with a seed.

FCS_RBG_EXT.1.2 The TSF shall use a **[selection:** ***TSF noise source*** **[assignment:** ***name of noise source*****],** ***TSF interface for seeding*****]** for initialized seeding.

FCS_RBG_EXT.1.3 The TSF shall update the RBG state using a noise source in the following situations: **[selection:**

- ***the TSF shall reseed the RBG on the condition: before generation of the DEK in accordance with***

    - ***ISO/IEC 18031:2011, Section [selection: C.2.2.2.3 Reseeding Hash_DRBG (…) Instantiation, C.2.3.2.4 Reseeding HMAC_DRBG (…) Instantiation, C.3.2.2.6 Reseeding CTR_DRBG (…) Instantiation];***

    - ***NIST SP 800-90A Section [selection: 10.1.1.3 Reseeding a Hash_DRBG Instantiation, 10.1.2.4 Reseeding an HMAC_DRBG Instantiation, 10.2.1.4 Reseeding a CTR_DRBG Instantiation];***

  using a [selection: TSF noise source [assignment: name of noise source], TSF interface for seeding];

- ***the TSF shall uninstantiate and instantiate the RBG on the condition: before generation of the DEK in accordance with***

    - ***ISO/IEC 18031:2011(ISO/IEC 18031:2011 is silent on uninstantiate), Section [selection: C.2.2.2.2 Instantiation of Hash_DRBG (…), C.2.3.2.3 Instantiation, of HMAC_DRBG, C.3.2.2.2 Instantiation of CTR_DRBG (…)];***

    - ***NIST SP 800-90A Section 9.4 Removing a DRBG Instantiation, and Section [selection: 10.1.1.2 Instantiation of Hash_DRBG, 10.1.2.3 Instantiation of HMAC_DRBG, 10.2.1.3 Instantiation of CTR_DRBG];***

  using a [selection: TSF noise source [assignment: name of noise source], TSF interface for seeding].

]

Application Note 11

If reseeding is not feasible, the TSF will uninstantiate RBGs, rather than produce output that is of insufficient quality. The listed standards should specify the reseed interval, and procedure for uninstantiating and reseeding. The remaining selection allows the PP Author to require application-specific conditions for reseeding.

"Uninstantiate" means that the internal state of the DRBG is no longer available for use.

Application Note 12

If the ST Author chooses any of the hash models of the DRBG, then the relevant hash function is also included in an FCS_COP.1 iteration covering hash algorithms. Choosing CTR_DRBG in the first selection in FCS_RBG_EXT.1.1 requires the use of AES as the block cipher, also included in an FCS_COP.1 iteration, – triple DES is not allowed.

Application Note 13

FCS_RBG_EXT.1.3 requires the internal state of the RBG to change by reseeding, or stopping and instantiating the RBG, which creates a new state. While the standard specifies how and

when the RBG is reseeded, the standard does not address the specific use case of this TOE, and therefore, the additional condition of doing so before generating the DEK is added to the requirement.

Health tests for the RBG are specified in FPT_TST.1.

5.4 User Data Protection (FDP)

5.4.1 Protection of User Data on Device (FDP_UDD_EXT)

User data is automatically protected by the TOE once the user has been authorised according to the requirements in section 5.5. Before authorisation no user data can be written to the device.

5.4.1.1 **FDP_UDD_EXT.1 Protection of User Data on Device**

FDP_UDD_EXT.1.1 The TSF shall encrypt all user data under the DEK without user intervention, in accordance with FCS_COP.1/UDE.

Application Note 14

The intent of this requirement is to specify that encryption of any protected data will not depend on a user electing to protect that data, and to show that all user data encryption depends on access to the DEK and therefore requires prior successful authorisation (in order to obtain the DEK via the key chain) to start the session. The drive encryption specified in FDP_UDD_EXT.1 occurs transparently to the user and the decision to protect the data is outside the discretion of the user, which is a characteristic that distinguishes it from file encryption. The definition of user data can be found in the glossary.

The TOE provides the cryptographic functions to encrypt/decrypt the user data as specified in FCS_COP.1/UDE.

5.4.2 Protection of System Data on Device (FDP_SDD_EXT)

System data is protected by the TOE to ensure that only authorised changes can be made to it.

5.4.2.1 **FDP_SDD_EXT.1 Protection of System Data on Device**

FDP_SDD_EXT.1.1 The TSF shall allow changes to system data only in the form of changes to configuration data resulting from TSF actions or actions by authorised users **[selection:** ***or a trusted update in accordance with FPT_TUD_EXT.1, none*****]**.

Application Note 15

The intent of this requirement is to specify that system data can only change in ways that reflect the legitimate use of the deveice by authorised users, or the application of a trusted update according to the FPT_TUD_EXT.1 requirements in section A.1.1. Since system data is product-specific, it is recognised that normal operation of the TOE will result in updates to some of the data (e.g. if the DEK is regenerated, or if a data recovery mechanism is disabled as in FMT_SMF.1/Extensions, or to update DRBG state), and this is what is meant by the phrase

"changes to system data only in the form of changes to configuration data resulting from TSF actions or actions by authorised users".

Note that if FPT_TUD_EXT.1 is chosen in the selection then this SFR has a dependency on FPT_TUD_EXT.1.

If 'none' is chosen in the selection then the selection may be left blank by the ST author.

5.5 Identification and Authentication (FIA)

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 9.

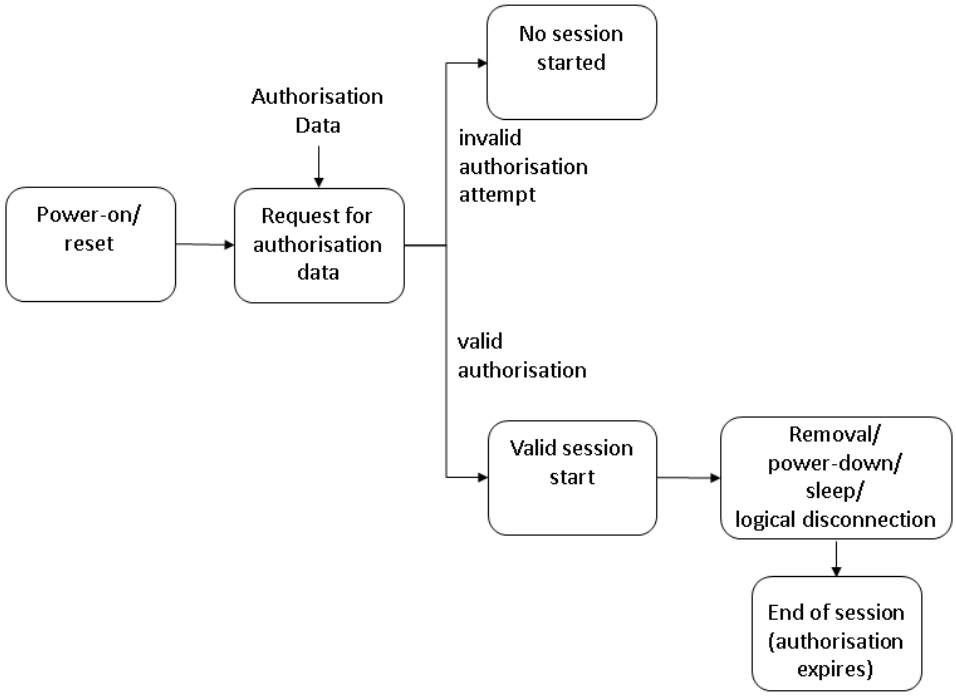


Figure 9: Session Lifecycle

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 are made available until either 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 is as follows:

- Input of authorisation data

The TOE does not provide access to user data until user authorisation has been successfully completed (section 5.7.1.1). An authorised user of the device provides authorisation data directly to the device or else via TOE software running on a host computer (section 5.5.2.1) that meets requirements stated in sections B.2.1.1 or B.2.1.2. 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.

If device allows the users to enter the passphrase on the TOE on the device then an obscured feedback must be provided as specified in B.2.2.1.1.

User authorisation is also required whenever the TOE itself or a host to which the TOE is connected recovers from a power-down or sleep state, or after a host has initiated termination of the current session (section 5.7.1.1).

- Conditioning of authorisation data (section 5.3.1.4)

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 9) and the plaintext user data is made available to the user. The authorisation data is used to derive a Key Encryption Key (KEK).

- Protection of authorisation data (section 5.6.2.1)

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.

- Authorisation failures (section 5.5.1.1)

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 will be detected as a result of a separate authorisation data validation mechanism as specified in section 5.6.4.1.

- Authorisation data recovery (section B.3.1.1)

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 mechanism 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).

- Authorisation data change (section 5.7.1.1 and 5.8.1.1)

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.

- Validation of authorisation data (section 5.6.4.1)

The device provides a function that validates the authorisation data separately from using the authorisation data to derive the KEK and decrypt the DEK.

5.5.1 Authentication Failures (FIA_AFL)

5.5.1.1 **FIA_AFL.1 Authentication failure handling**

FIA_AFL.1.1 The TSF shall detect when **[selection: *[assignment: positive integer number], a user configurable positive integer within [assignment: range of acceptable values]*]** unsuccessful **authorisation** attempts occur related to **user authorisation**.

FIA_AFL.1.2 When the defined number of unsuccessful **authorisation** attempts has been **met**, the TSF shall **destroy the current stored DEK in accordance with FCS_CKM.4**.

Application Note 16

The device makes the user data permanently unavailable by deleting the current DEK as soon as a threshold number of consecutive authorisation failures is reached. The device recognises both recovery of an incorrect key in the key chain for FCS_KYC_EXT.1 and failure of submask validation as described in FPT_VAL_EXT.1 as authorisation attempt failures.

The developer shall enter the number of unsuccessful authorisation attempts or provide a range of acceptable values that could be configured by the authorised user.

When the threshold of unsuccessful attempts in FIA_AFL.1.2 has been exceeed then the current stored (wrapped) DEK is deleted, meaning that the data encrypted under it is no longer accessible. (Note also that if the product implements any data recovery mechanisms then these are required to be disabled as in P.Recovery (section 3.3.2) and FMT_SMF.1/Extensions.

Please note that "authentication" refers to "authorisation" to reflect the fact that the device does not need to hold reference authentication data, or other data that identifies users as individuals. The term "authentication" is used here to comply with CC Part 2 wording.

5.5.2 Passphrase support (FIA_PPS_EXT)

5.5.2.1 **FIA_PPS_EXT.1 Passphrase entry interface**

FIA_PPS_EXT.1.1 The TSF shall support **[selection:**

- ***Entry of passphrase via a keypad on the TOE according to FIA_PPS_EXT.2/num,***
- ***Submission of passphrase according to FIA_PPS_EXT.2/alph*]**.

Application Note 17

The TSF shall support passphrases set by the user that include characters specified in FIA_PPS_EXT.2/num or FIA_PPS_EXT.2/alph as appropriate to the method of passphrase entry for the TOE. Note that FIA_PPS_EXT.2/num only applies to the case where the TOE includes a numeric keypad: if the passphrase is entered via host software, or if the TOE includes a passphrase-entry interface with characters other than those specific in FIA_PPS_EXT.2/num then FIA_PPS_EXT.2/alph should be selected.

5.6 Protection of the TSF (FPT)

The SFRs in this section are concerned with the two particular aspects of the ways in which the TOE protects itself:

- Self-testing on start-up

  The TOE is required to carry out known answer tests for cryptographic functions, to confirm that they are still working correctly and therefore to detect any failures that might put user data at risk by storing it unencrypted, or with the encryption weakened in some way (section 5.6.3.1). It is also required to check its fimrware integrity on start-up, to check for any corruptions or attempts to replace the authentic firmware (section 5.6.3.1). If a failure is detected then the TOE needs to respond in a way that does not put user data at risk, which will typically require the device to enter an irreversible 'mute' state (section 5.6.1.1).

- Protection of keys and keying material

  The TOE is required to protect keys and keying material (including authorisation data entered by the user) by never storing these items in plaintext in non-volatile memory (section 5.6.2.1).

5.6.1 Fail secure (FPT_FLS)

5.6.1.1 FPT_FLS.1 Failure with preservation of secure state

FPT_FLS.1.1 The TSF shall **enter an irreversible mute** state when the following types of failures occur:

- **known answer test failures during initial start-up**
- **firmware integrity test failures during initial start-up**
- **[assignment: *other failures leading to irreversible mute state*]**.

Application Note 18

If the device fails the self-tests specified in FPT_TST.1 then the device enters an irreversible 'mute' state in which the device does not respond or communicate with the host; in particular, the device does not then decrypt any user data stored on the device and does not allow modification of the system data.

5.6.2 Protection of Keys and Keying Material (FPT_KYP_EXT)

5.6.2.1 FPT_KYP_EXT.1 Protection of Keys and Keying Material

FPT_KYP_EXT.1.1 The TSF shall only store keys and keying material in non-volatile memory when wrapped, as specified in FCS_CKM.3, or encrypted, as specified in FCS_COP.1/KeyEnc unless the keys or keying material meets any one of following criteria:

- The plaintext keys or keying material is not part of the key chain as specified in FCS_KYC_EXT.1.
- The plaintext keys or keying material will no longer provide access to the encrypted data after initial provisioning.

Application Note 19

The ST author needs to use FCS_CKM.3/Chain and/or FCS_COP.1/KeyEnc from Appendix B (Selection-based security requirements) for keys and keying material that are stored in non-volatile memory (in addition to the use of FCS_CKM.3/DEK for the wrapping of the DEK with the KEK). The ST Author may need to iterate this requirement if different methods are used to wrap and/or encrypt various keys and keying material.

Note that keying material includes plaintext authorisation data.

5.6.3 TSF self test (FPT_TST)

5.6.3.1 FPT_TST.1 TSF testing

FPT_TST.1.1 The TSF shall run a suite of self tests:

- **known answer tests of the cryptographic algorithms**
- **random bit generator known answer tests that meet the requirements of NIST SP 800-90A Revision 1, section 11.3 Health Testing**
- **firmware integrity tests**

during initial start-up to demonstrate the correct operation of **the TSF**.

Application Note 20

The known answer health tests in NIST SP 800-90A Revision 1 cover the instantiation, generation and reseeding functions of the RBG. Health tests are only required to be generated on initial start-up of the device; the capability to run health tests of the random bit generator on demand (identified as 'should' requirements in NIST SP 800-90A Revision 1) is not required in this cPP.

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of **[selection: *[assignment: parts of TSF data], TSF data*]**.

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of **the [selection: *[assignment: parts of TSF], TSF*] firmware**.

5.6.4 Submask Validation (FPT_VAL_EXT)

5.6.4.1 FPT_VAL_EXT.1 Validation

FPT_VAL_EXT.1.1 The TSF shall perform validation of the authorisation data submask using the following methods: **[selection:**

- ***key wrap as specified in FCS_CKM.3,***
- ***hash the authorisation data submask as specified in [selection: FCS_COP.1/Hash, FCS_COP.1/HMAC] and compare it to a stored hashed value of the authorisation data,***
- ***decrypt a known value using the submask as specified in FCS_COP.1/KeyEnc and compare it against a stored known value*****].**

Application Note 21

The ST Author may need to iterate this requirement if different methods are used to validate the submask. The link between individual submask validation actions and the definition of an authorisation attempt failure for FIA_AFL.1 needs to be made clear as part of the description in the ST.

Once the ST Authors have selected a method to perform validation of the submask, they include the appropriate requirement from Appendix B (Selection-based security requirements).

5.7 TOE Access (FTA)

The SFR in this section is part of the user authorisation model described in section 5.5.

5.7.1 TOE access authorisation (FTA_USB_EXT)

5.7.1.1 FTA_USB_EXT.1 User Authorisation

FTA_USB_EXT.1.1 The TSF shall require that a valid passphrase is supplied before starting a session that allows any access to user data on the TOE.

FTA_USB_EXT.1.2 The TSF shall authorise the user under the following conditions:

- connection of the TOE to a host device
- recovery of a host device from a power-down or sleep state while the TOE is connected to it
- recovery of the TOE from its own power-down or sleep state
- changing the value of authorisation data
- **[assignment:** ***list of other conditions under which re-authorisation is required*****].**

FTA_USB_EXT.1.3 The TSF shall allow host-initiated termination of the current session.

5.8 Security Management (FMT)

The TOE is required to support a function that enables the user to change their authorisation data.

5.8.1 Specification of Management Functions (FMT_SMF)

5.8.1.1 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions:

- **change the value of the authorisation data.**

Application Note 22

When changing the authorisation data re-authorisation at the point of making the change is required as in FTA_USB_EXT.1 (section 5.7.1.1).

6. Security Assurance Requirements

This cPP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of SARs from CC part 3 that are required in evaluations against this cPP. Individual Evaluation Activities to be performed are specified in [SD]. The Evaluation Activities are interpretations of the CEM work units.

The general model for evaluation of TOEs against STs written to conform to this cPP is as follows: after the ST has been approved for evaluation, the evaluator will obtain the TOE, supporting environmental IT (if required), and the administrative/user guides for the TOE. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The evaluator also performs the Evaluation Activities contained within [SD], which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The Evaluation Activities that are captured in [SD] also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the cPP.

The TOE security assurance requirements are identified in Table 1.

Assurance Class	**Assurance Components**
Security Target (ASE)	Conformance claims (ASE_CCL.1)
	Extended components definition (ASE_ECD.1)
	ST introduction (ASE_INT.1)
	Security objectives for the operational environment (ASE_OBJ.1)
	Stated security requirements (ASE_REQ.1)
	Security Problem Definition (ASE_SPD.1)
	TOE summary specification (ASE_TSS.1)
Development (ADV)	Basic functional specification (ADV_FSP.1)
Guidance documents (AGD)	Operational user guidance (AGD_OPE.1)
	Preparative procedures (AGD_PRE.1)
Life cycle support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM coverage (ALC_CMS.1)
Tests (ATE)	Independent testing – conformance (ATE_IND.1)
Vulnerability assessment (AVA)	Vulnerability survey (AVA_VAN.1)[9]

Table 1: Security Assurance Requirements

[9] Please see section 6.6 for more information.

6.1 ASE: Security Target

6.1.1 Refinement of ASE_TSS

The ST is evaluated as per ASE activities defined in the CEM. In addition, there may be Evaluation Activities specified within [SD] that call for necessary descriptions to be included in the TSS that are specific to the TOE technology type.

Appendix D provides a description of the information expected to be provided regarding the quality of entropy in the random bit generator.

Given the criticality of the key management scheme, this cPP requires the developer to provide a detailed description of their key management implementation. This information can be submitted as an appendix to the ST and marked proprietary, as this level of detailed information is not expected to be made publicly available. See Appendix D.2 for details on the expectation of the developer's Key Management Description.

The requirement ASE_TSS.1.1C is therefore refined as follows.

ASE_TSS.1.1C The TOE summary specification shall describe how the TOE meets each SFR, including supplementary information on Entropy Documentation and Assessment (Appendix D.1) and a proprietary Key Management Description (Appendix D.2).

6.2 ADV: Development

The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any additional information required by this cPP that is not to be made public (e.g. Entropy Documentation and Assessment).

6.2.1 Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TOE Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces. For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional "functional specification" documentation is necessary to satisfy the Evaluation Activities specified in [SD].

The Evaluation Activities in [SD] are associated with the applicable SFRs; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2C is implicitly already done and no additional documentation is necessary.

6.3 AGD: Guidance Documentation

The guidance documents will be provided with the TOE. Guidance must include a description of how the IT personnel verify that the Operational Environment can fulfill its role for the

security functionality. The documentation should be in an informal style and readable by the target audience.

Guidance pertaining to particular security functionality must also be provided; requirements on such guidance are contained in the Evaluation Activities specified in [SD] and CC Part 3.

6.3.1 Operational User Guidance (AGD_OPE.1)

The operational user guidance does not have to be contained in a single document. Guidance to users can be spread among documents.

The developer should review the Evaluation Activities contained in [SD] to ascertain the specifics of the guidance that the evaluator will be checking for. This together with CC Part 3 will provide the necessary information for the preparation of acceptable guidance. The evaluator is also expected to perform the CEM work units associated with AGD_OPE.1.

6.3.2 Preparative Procedures (AGD_PRE.1)

As with the operational guidance, the developer should look to the Evaluation Activities together with CC Part 3 to determine the required content with respect to preparative procedures. The evaluator is also expected to perform the CEM work units associated with AGD_PRE.1.

6.4 Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this cPP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it is a reflection on the information to be made available for evaluation at this assurance level.

6.4.1 Labelling of the TOE (ALC_CMC.1)

This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being purchased by an end user. The evaluator performs the CEM work units associated with ALC_CMC.1.

6.4.2 TOE CM Coverage (ALC_CMS.1)

Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

6.5 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family,

while the latter is through the AVA_VAN family. For this cPP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

6.5.1 Independent Testing – Conformance (ATE_IND.1)

Testing is performed to confirm the functionality described in the TSS as well as the operational guidance (includes "evaluated configuration" instructions). The focus of the testing is to confirm that the requirements specified in Section 5 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing according to requirements stated in CC, Part 3.

6.5.2 Refinement of ATE_IND.1

Where tests cannot be performed using the normal product interfaces, the developer shall make available to the evaluator tools to enable testing as required in [SD].

Application Note 23

An example of such a case would be tools to enable the inspection of internal TOE state for testing FCS_CKM.4.

6.6 Class AVA: Vulnerability Assessment

[SD, Appendix A] together with [CC3] provides a guide to the evaluator in performing a vulnerability analysis.

A. Optional Requirements

As indicated in the introduction to this cPP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this cPP. Additionally, there are two other types of requirements specified in Appendices A and B.

The first type (in this Appendix) is requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this cPP. The second type (in Appendix B) is requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements in that appendix will need to be included in the body of the ST (e.g., cryptographic protocols selected in a trusted channel requirement).

A.1 Protection of the TSF (FPT)

If the TOE allows updates to system data then it must provide a mechanism for ensuring that only trusted updates can be successfully applied (FPT_TUD_EXT.1 in section A.1.1.1). The TOE may optionally provide additional prevention of rollback to an earlier version of system data (FPT_TUR_EXT.1 in section A.1.2.1).

A.1.1 Trusted Update (FPT_TUD_EXT)

A.1.1.1 FPT_TUD_EXT.1 Trusted Update

FPT_TUD_EXT.1.1 The TSF shall provide authorised users with the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The TSF shall provide authorised users with the ability to initiate updates to TOE firmware/software.

FPT_TUD_EXT.1.3 The TSF shall verify updates to the TOE firmware/software using a **[selection: *digital signature mechanism, published hash, keyed hash*]** by the manufacture prior to installing those updates.

Application Note 24

This SFR is applicable only if the TOE provides the capability to update the TOE firmware. If included, then FMT_SMF.1/Extensions must also be present in the ST to enable the ability to query the version(s) and initiate updates.

The digital signature mechanism referenced in the third element is specified in FCS_COP.1/SigVer, the published hash mechanism is specified in FCS_COP.1/Hash and keyed hash mechanism is specified in FCS_COP.1/HMAC.

A.1.2 Trusted Update Rollback (FPT_TUR_EXT)

A.1.2.1 FPT_TUR_EXT.1 Trusted Update Rollback

FPT_TUR_EXT.1 The TSF shall not allow rollback of previously applied updates.

Application Note 25

Prevention of Rollback of previously applied updates is optional even when the device supports updates to the TOE firmware. If it is claimed, then the ST must also include FPT_TUD_EXT.1.

B. Selection-Based Requirements

As indicated in the introduction to this cPP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this cPP. There are additional requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements below will need to be included.

B.1 Cryptographic Support (FCS)

These cryptographic functions are used as appropriate for operations in the key chain (see section 5.3).

B.1.1 Cryptographic Key Management (FCS_CKM)

B.1.1.1 FCS_CKM.1/Asymm Key generation (Asymmetric Encryption/Decryption Key)

FCS_CKM.1.1/Asymm The TSF shall generate **asymmetric** cryptographic keys **[selection: *key name*]** ~~in accordance with a specified cryptographic key generation algorithm [selection: *cryptographic key generation algorithm*] and~~ **with** specified cryptographic key sizes **[selection: *key sizes*]** that meet the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_CKM.1.1/Asymm.

Identifier	key name	key sizes	list of standards
AKG1	RSA	[selection: 2048 bit, 3072 bit	FIPS PUB 186-4 (Section B.3)
AKG2	ECC	[selection: 256 (P-256), 384 (P-384), 512 (P-521)]	FIPS PUB 186-4 (Section B.4 & D.1.2)
AKG3	ECC	[selection: 256 (brainpoolP256r1), 384 (brainpoolP384r1), 512 (brainpoolP512r1)]	RFC5639 (Section 3) [Brainpool Curves] FIPS PUB 186-4 (Section B.4)

Application Note 26

This SFR is included for the purposes of encryption and decryption operations only.

FCS_CKM.1/Asymm mandates compliance to FIPS 186-4. Implementations according to FIPS 186-2 or FIPS 186-3 will not be accepted.

B.1.1.2 FCS_CKM.3/Chain Cryptographic Key Access (Key Wrapping)

Note that the iteration of FCS_CKM.3 below is identical to that in section 5.3.1.2, but is only selected if the TSF performs wrapping at other points in the key chain in addition to the final wrapping of the DEK with the KEK.

FCS_CKM.3.1/Chain The TSF shall perform **[selection: *type of cryptographic key access*]** in accordance with a specified cryptographic key access method **[selection: *cryptographic key access method*]** that meets the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_CKM.3/Chain.

Identifier	type of cryptographic key access	cryptographic key access method	list of standards
KW1	Key wrapping	KWP[10] based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), NIST SP 800-38F, sec. 6.3 (KWP) ISO/IEC 19772, clause 7 (Key wrap)
KW2	Key wrapping	KW based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), NIST SP 800-38F, sec 6.2 (KW)
KW3	Key wrapping	GCM based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), ISO/IEC 19772, clause 11 (GCM)
KW4	Key wrapping	CCM based on AES with [selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (AES), ISO/IEC 19772, clause 8 (CCM)

[10] Key Wrap with Padding

Application Note 27

FCS_CKM.3/DEK is always used in an ST in order to capture the last stage of the key chain in which the DEK is wrapped by the KEK (see Figure 8 and section 5.3.1.2). However, this additional iteration may be included in an ST if needed to cover other use in the product-specific part of the key chain (see Figure 7 and FCS_KYC_EXT.1 in section 5.3.3.1).

Application Note 28

Key wrapping mechanisms use always authenticated-encryption modes.

B.1.2 Cryptographic Key Derivation (FCS_CKM_EXT)

B.1.2.1 Cryptographic Key Derivation FCS_CKM_EXT.5/Chain

FCS_CKM_EXT.5.1/Chain The TSF shall derive cryptographic keys **[assignment: *key type*]** from **[selection: *input parameters*]** in accordance with a specified cryptographic key derivation algorithm **[selection: *key derivation algorithm*]** and specified cryptographic key sizes **[selection: *key sizes*]** that meet the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_CKM_EXT.5/Chain.

Identifier	key type	input parameters	key derivation algorithm	key sizes	list of standards
1	Authorization Factor Submask	Password	HMAC-[selection: SHA-256, SHA-512], with [assignment: positive integer of 1000 or more] iterations	[selection: 128, 256] bits	NIST SP 800-132
2	[assignment: key name]	Intermediary keys	[selection: exclusive OR (XOR), SHA-256, SHA-512]	[selection: 128 bits, 256] bits	[selection: ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6) [SHA]]
3	[assignment: key name]	Direct Generation from a Random Bit Generator as specified in	KDF in Counter Mode using [selection: CMAC-	[selection: 128, 256] bits	NIST SP 800-108 (Section 5.1) [KDF in Counter Mode]

Identifier	key type	input parameters	key derivation algorithm	key sizes	list of standards
		FCS_RBG_EXT.1	AES-128, CMAC-AES-192, CMAC-AES-256 HMAC-SHA-256, HMAC-SHA-512] as the PRF		[selection: ISO/IEC9797-1(Section B.6, B7), NIST SP800-38B) [CMAC] ISO/IEC 18033-3:2010 [AES], ISO/IEC 9797-2:2011 (Section 7 MAC Algorithm 2 (HMAC)), FIPS 198-1, ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6) [SHA]]
4	[assignment: key name]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Feedback Mode using [selection: CMAC-AES-128, CMAC-AES-192, CMAC-AES-256 HMAC-SHA-256, HMAC-SHA-512] as the PRF	[selection: 128, 256] bits	NIST SP 800-108 (Section 5.2) [KDF in Feedback Mode] [selection: ISO/IEC9797-1(Section B.6, B7), NIST SP800-38B) [CMAC] ISO/IEC 18033-3:2010 [AES], ISO/IEC 9797-2:2011 (Section 7 MAC Algorithm 2 (HMAC)), FIPS 198-1, ISO10118-3, (Section 10, 11); FIPS180-4,

Identifier	key type	input parameters	key derivation algorithm	key sizes	list of standards
					(Section 6) [SHA]]
5	[assignment: key name]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Double-Pipeline Iteration Mode using [selection: CMAC-AES-128, CMAC-AES-192, CMAC-AES-256 HMAC-SHA-256, HMAC-SHA-512] as the PRF	[selection: 128, 256] bits	NIST SP 800-108 (Section 5.3) [KDF in Double-Pipeline Iteration Mode] [selection: ISO/IEC9797-1(Section B.6, B7), NIST SP800-38B) [CMAC] ISO/IEC 18033-3:2010 [AES], ISO/IEC 9797-2:2011 (Section 7 MAC Algorithm 2 (HMAC)), FIPS 198-1, ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6) [SHA]]

Application Note 29

For identifier 1, 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 $L2 \leq k \leq L1$.

Application Note 30

FCS_CKM_EXT.5/Chain is used in the body of the ST (in addition to the FCS_CKM_EXT.5/KEK iteration for derivation of the KEK) if the ST Author chooses to use key derivation in the product-specific key chaining approach that is specified in FCS_KYC_EXT.1.

It should be noted that Section 7.5 of NIST SP 800-108 describes the approaches when keying material for multiple cryptographic keys is obtained from the output of a key derivation function.

B.1.3 Cryptographic Operation (FCS_COP)

B.1.3.1 FCS_COP.1/KeyEnc Cryptographic operation (Key Encryption)

FCS_COP.1.1/KeyEnc The TSF shall perform **key encryption and decryption** in accordance with a specified cryptographic algorithm **AES in CBC mode**, and the cryptographic key size **[selection: *128 bits, 256 bits*]** that meet the following: **ISO/IEC 18033-3 (AES), ISO/IEC 10116(CBC)**.

Application Note 31

This requirement is used in the body of the ST if the ST Author chooses to use AES encryption/decryption for protecting the keys as part of the key chaining approach that is specified in FCS_KYC_EXT.1.

Key wrapping as in FCS_CKM.3 provides key validation as well as authenticity; but FCS_COP.1/KeyEnc provides encryption for confidentiality only: there is no validation nor authenticity.

B.1.3.2 FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)

FCS_COP.1.1/Hash The TSF shall perform **cryptographic hashing services** in accordance with a specified cryptographic algorithm **[selection: *SHA-1, SHA-256, SHA-384, SHA-512*]** ~~and cryptographic key sizes [*assignment: cryptographic key sizes*]~~ that meet the following: **ISO/IEC 10118-3:2004**.

Application Note 32

The hash selection should be consistent with the overall strength of the algorithm used for FCS_KYC_EXT.1. For example, SHA-256 should be chosen for 2048-bit RSA or ECC with P-256, SHA-384 should be chosen for 3072-bit RSA, 4096-bit RSA, or ECC with P-384, and SHA-512 should be chosen for ECC with P-521. The selection of the standard is made based on the algorithms selected.

Application Note 33

SHA-1 may be used for the following applications: generating and verifying hash-based message authentication codes (HMACs), key derivation functions (KDFs), and random bit/number generation[11].

B.1.3.3 FCS_COP.1/HMAC Cryptographic Operation (Keyed hash)

FCS_COP.1.1/HMAC The TSF shall perform **keyed hash message authentication** in accordance with a specified cryptographic algorithm **[selection: *HMAC-SHA-256, HMAC-***

[11] In certain cases SHA-1 may also be used for verifying old digital signatures and time stamps, provided that this is explicitly allowed by the application domain.

***SHA-512*]** and cryptographic key sizes **[assignment: *key size (in bits) used in HMAC*]** that meet the following: **ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2".**

Application Note 34

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 $L2 \leq k \leq L1$.

B.1.3.4 FCS_COP.1/SigVer Cryptographic Operation (Signature Verification)

FCS_COP.1.1/SigVer The TSF shall perform **digital signature verification for authenticity** in accordance with a specified cryptographic algorithm **[selection: *cryptographic algorithm*]** and cryptographic key sizes **[selection: *cryptographic key sizes*]** that meet the following: **[selection: *list of standards*]**.

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/SigVer.

Identifier	cryptographic algorithm	key sizes	list of standards
SigVer1	RSASSA-PKCS1-v1_5 using [selection*: SHA-256, SHA-512*]	[selection: 2048 bit, 3072 bit]	[selection: RFC3447, PKCS #1 v2.1 (Section 8.2), FIPS186-4, (Section 5.5)] [RSASSA-PKCS1-v1_5] [selection: ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6)] [SHA]
SigVer2	Digital signature scheme 2 using [selection: SHA-256, SHA-512]	[selection: 2048 bit, 3072 bit]	ISO9796-2, (Section 9) [Digital signature scheme 2] [selection: ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6)] [SHA]
SigVer3	Digital signature scheme 3 using [selection: SHA-256, SHA-512]	[selection: 2048 bit, 3072 bit]	ISO9796-2, (Section 10) [Digital signature scheme 3] [selection: ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6)] [SHA]
SigVer4	RSASSA-PSS using [selection: *SHA-256, SHA-512]*	[selection: 2048 bit, 3072 bit]	RFC3447, PKCS#1v2.1 (Section 8.1) [RSASSA-PSS] [selection: ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6)] [SHA]

Identifier	cryptographic algorithm	key sizes	list of standards
SigVer5	ECDSA on [selection: *brainpoolP256r1, brainpoolP384r1, brainpoolP512r1, NIST P-256, NIST P-384, NIST P-521*] using [selection: SHA-256, SHA-512]	[selection: *256 bits, 384 bits, 512 bits*]	[selection: ISO14888-3; FIPS186-4 (Section 6)] [ECDSA] RFC5639 (Section 3) [Brainpool Curves] FIPS186-4 (Appendix D.1.2) [NIST Curves] [selection: ISO10118-3, (Section 10, 11); FIPS180-4, (Section 6)] [SHA]

Application Note 35

This SFR is applicable only if the TOE provides the capability to update the TOE firmware and uses digital signatures for update verification.

B.1.4 Random Bit Generation (FCS_RBG_EXT)

The ST Author must choose one of the following options

- one internal source (as in FCS_RBG_EXT.3)
- one or more internal source and external interface (as in FCS_RBG_EXT.3/.4 and FCS_RBG_EXT.2) combined by FCS_RBG_EXT.5
- a combination of multiple internal sources (as in FCS_RBG_EXT.4) combined by FCS_RBG_EXT.5

Endorsement Statements can be used to state what an entity desires to see.

B.1.4.1 FCS_RBG_EXT.2 Random Bit Generation (External Seeding)

FCS_RBG_EXT.2.1 The TSF shall be able to accept a minimum input of **[assignment: *128 bits or greater*]** from a TSF interface for the purpose of seeding.

Application Note 36

For this SFR, the interface to obtain the entropy noise source can be used multiple times to provide input. For instance, if the input length is 128 bits, it could be used twice to gather 256 bits. In this instance, the 128 bits would not be provided to the DRBG, since the DRBG can only be instantiated once, rather a function would gather the 128 bits twice and provide the DRBG with 256 bits of entropy noise source.

This SFR does not describe requirements on seed quality: it is the responsibility of the operational environment to define their requirement in this regard and to ensure that it is met by the external source.

Guidance in the introduction to PP/ST authors should address protection from modification and disclosure of the value from the external noise source, as well as the leaking of any pertinent information (e.g., internal state) regarding the RBG.

The Operational Guidance describes the use of the external input interface, and any requirements for the inputs provided (e.g. requirements for achieving a desired level of entropy).

B.1.4.2 FCS_RBG_EXT.3 Random Bit Generation (Internal Seeding Single Source)

FCS_RBG_EXT.3.1 The TSF shall be able to seed the RBG using a **[selection: choose one of: *TSF software-based noise source, TSF hardware-based noise source*] [assignment: *name of noise source*]** with a minimum of **[assignment: *number of bits*]** bits of min-entropy.

B.1.4.3 FCS_RBG_EXT.4 Random Bit Generation (Internal Seeding Multiple Sources)

FCS_RBG_EXT.4.1 The TSF shall be able to seed the RBG using **[selection: *one TSF software-based noise source(s), one TSF hardware-based noise source(s)*].**

Application Note 37

If an ST Author wishes to use multiple internal noise sources, they iterate this requirement for each noise source being used by the TSF.

Hardware-based noise sources are sources whose primary function is noise generation, such as e ring oscillators, diodes, and thermal noise. While software is used to collect the noise from these hardware sources, these are not software-based. Software-based noise sources are those sources that have some other primary function and the noise is a byproduct of their normal operation. Examples of software-based noise sources are user or system based events, reading the least significant bits from an event timer, etc.

Hardware-based noise sources may be stochastically modelled, in which case the amount of entropy is better understood. Software-based noise sources are usually less well understood and therefore will typically take a more conservative approach, gathering larger numbers of bits than required and then performing a compression function to derive the final output. Software-based noise sources often rely on an entropy estimator.

Application Note 38

If ST Author wants to allow a single internal source for seed (only use FCS_RBG_EXT.3), they do not need SFR FCS_RBG_EXT.5. If they want to combine noise sources, then they would use the following SFR.

B.1.4.4 FCS_RBG_EXT.5 Random Bit Generation (Combining Noise Sources)

FCS_RBG_EXT.5.1 The TSF shall **concatenate** [selection: *output from TSF noise source(s), input from TSF interface(s) for seeding*] to create the entropy input into the derivation function as defined in [**selection:**

- ***ISO/IEC 18031:2011 Section: [selection: C.2.2.2.2 Instantiation of Hash_DRBG (…), C.2.3.2.2 Internal function: The Update function, C.3.2.2.4 Derivation function using a block cipher algorithm];***

- ***NIST SP 800-90A Section: [selection: 10.3.1 Derivation Function Using a Hash Function (Hash_df), 10.1.2.2 The HMAC_DRBG Update Function (Update), 10.3.2 Derivation Function Using a Block Cipher Algorithm (Block_Cipher_df)]***,

resulting in a minimum of [**selection:** ***128, 256***] bits of min-entropy.

Application Note 39

In either standard, the CTR_DRBG algorithm allows for instances where a derivation function is not required. This SFR mandates the use of the derivation function, regardless of the standard.

B.2 Identification and Authentication (FIA)

These functions define passphrase composition requirements according to the passphrase interface specified in FIA_PPS_EXT.1 (see section 5.5.2).

B.2.1 Passphrase support (FIA_PPS_EXT)

B.2.1.1 FIA_PPS_EXT.2/num Passphrase composition – numeric

This SFR defines passphrase composition requirements for passphrases entered at a TOE interface (e.g. a keypad on the device), and is only needed when that keypad contains a limited character set (in this case it is assumed that the keypad is purely numeric).

FIA_PPS_EXT.2.1/num The TSF shall support passphrases that consist of:

- **Digits 0-9**
- **[assignment:** ***other supported characters*****]**

FIA_PPS_EXT.2.2/num The TSF shall provide a mechanism to verify that passphrases have a minimum length of **8 characters**.

FIA_PPS_EXT.2.3/num The TSF shall support passphrases with a maximum length of at least **64 characters**.

Application Note 40

The TSF in this case is accepting passphrases entered on a keypad that forms part of the TOE itself. In this case FIA_UAU.7 also applies and is included in the ST.

Note that FIA_PPS_EXT.2.2/num means that the TOE must reject attempts to set a passphrase that is shorter than this minimum length.

The ST author must make the appropriate assignment to specify other characters supported by the TOE. If this is 'none' then the last bullet may simply be omitted.

B.2.1.2 FIA_PPS_EXT.2/alph Passphrase composition – alphanumeric

This SFR defines passphrase composition requirements for passphrases received by the TOE via an interface that allows a larger range of characters than in FIA_PPS_EXT.2/num. This will typically be the case when a passphrase is entered via software running on the host, but may also be used it there is a TOE interface that allows a larger character set.

FIA_PPS_EXT.2.1/alph The TSF shall support passphrases that consist of:

- **upper case characters**
- **lower case characters**
- **digits**
- **[assignment: *other supported characters*]**

FIA_PPS_EXT.2.2/alph The TSF shall provide a mechanism to verify that passphrases have a minimum length of **8 characters**.

FIA_PPS_EXT.2.3/alph The TSF shall support passphrases with a maximum length of at least **64 characters**.

Application Note 41

The TSF in this case is accepting passphrases submitted from the host, or where entry is via an interface that is part of the TOE itself and a larger range of characters is available than in FIA_PPS_EXT.2/num. If the entry of characters is via the TOE itself then FIA_UAU.7 must also be included in the ST.

Note that FIA_PPS_EXT.2.2/alph means that the TOE must reject attempts to set a passphrase that is shorter than this minimum length.

The ST author must make the appropriate assignment to specify other characters supported by the TOE. If this is 'none' then the last bullet may simply be omitted.

B.2.2 User authentication (FIA_UAU)

B.2.2.1.1 FIA_UAU.7 Protected authentication feedback

FIA_UAU.7.1 The TSF shall provide only **obscured feedback** to the user while the **authorisation** is in progress.

Application Note 42

This SFR is included in an ST if passphrases are entered using a physical interface device provided by the TOE itself (such as a numeric keypad incorporated into the device). This SFR is not inlcuded if passwords are entered on the host and sent to the TOE via a logical interface.

"Obscured feedback" implies that the TSF does not display values of the passphrase, although an obscured feedback may be provided (such as an asterisk for each character). This requirement is only applicable for devices that allow users to enter the passphrase on the TOE.

Please note that "authentication" refers to "authorisation" to reflect the fact that the device does not need to hold reference authentication data, or other data that identifies users as individuals. The term "authentication" is used here to comply with CC Part 2 wording.

B.3 Security Management (FMT)

This SFR captures management actions associated with other optional SFRs (as described in Application Note 43).

B.3.1 Specification of Management Functions (FMT_SMF)

B.3.1.1 FMT_SMF.1/Extensions Specification of Management Functions

FMT_SMF.1.1/Extensions The TSF shall be capable of performing the following management functions: **[selection:**

- ***define a user configurable number of unsuccessful authentication attempts***
- ***disable data recovery mechanism***
- ***enable data recovery mechanism, delete the current DEK, and then generate a new DEK as specified in FCS_CKM.1/DEK,***
- ***query the current version of the TOE firmware/software***
- ***initiate updates to the TOE firmware/software*****]**

Application Note 43

The ST author shall include the first bullet in the ST only if the user can configure number of unsuccessful authentication attempts, i.e. the second selection has been made in FIA_AFL.1.1.

The ST author shall include the second and the third bullet in the ST only if the device provides a data recovery mechanism (cf. section 3.3.2). Note that if data recovery is enabled from the disabled state then it is required to generate a replacement DEK immediately (meaning that data stored on the device up to the point of enabling the mechanism will no longer be accessible).

The ST author shall include the last two bullets in the ST only if the TOE provides the capability to update the TOE firmware (in whichf case the ST will also include FPT_TUD_EXT.1 (and possibly FPT_TUR_EXT.1)).

C. Extended Component Definitions

This appendix contains the definitions for the extended requirements that are used in the cPP, including those used in Appendices A and B.

FCS_CKM_EXT.5	Key Derivation Function
FCS_KYC_EXT.1	Key Chaining
FCS_RBG_EXT.1	Random Bit Generation (RBG)
FCS_RBG_EXT.2	Random Bit Generation (External Seeding)
FCS_RBG_EXT.3	Random Bit Generation (Internal Seeding Single Source)
FCS_RBG_EXT.4	Random Bit Generation (Internal Seeding Multiple sources)
FCS_RBG_EXT.5	Random Bit Generation (Combining Noise Sources)
FCS_RBG_EXT.6	Random Bit Generation (RBG Services)
FCS_SLT_EXT.1	Cryptographic Operation (Cryptographic Salt Generation)
FDP_UDD_EXT.1	Protection of User Data on Device
FDP_SDD_EXT.1	Protection of System Data on Device
FIA_PPS_EXT.1	Passphrase entry interface
FIA_PPS_EXT.2/num	Passphrase composition – numeric
FIA_PPS_EXT.2/alph	Passphrase composition – alphanumeric
FPT_KYP_EXT.1	Protection of Keys and Keying Material
FPT_TUD_EXT.1	Trusted Update
FPT_TUR_EXT.1	Trusted Update Rollback
FPT_VAL_EXT.1	Validation
FTA_USB_EXT.1	User Authorisation

Table 2: Extended Component Definitions

C.1 Cryptographic Support (FCS)

C.1.1 Cryptographic Key Derivation (FCS_CKM_EXT)

Family Behaviour

This family specifies the means by which an intermediate key is derived from a specified set of submasks.

Component levelling

FCS_CKM_EXT.5 Cryptographic Key Derivation, requires the TSF to derive intermediate keys from submasks using the specified hash functions.

Management: FCS_CKM_EXT.5

There are no management activities foreseen.

Audit: FCS_CKM_EXT.5

There are no auditable events foreseen.

FCS_CKM_EXT.5 Cryptographic Key Derivation

Hierarchical to: No other components

Dependencies: No dependencies

FCS_CKM_EXT.5.1 The TSF shall derive cryptographic keys **[assignment: *key type*]** from **[selection: *input parameters*]** in accordance with a specified cryptographic key derivation algorithm **[selection: *key derivation algorithm*]** and specified cryptographic key sizes **[selection: *key sizes*]** that meet the following: **[selection: *list of standards*]**.

C.1.2 Key Chaining (FCS_KYC_EXT)

Family Behaviour

This family provides the specification to be used for using multiple layers of encryption keys to ultimately secure the protected data encrypted on the device.

Component levelling

FCS_KYC_EXT.1 Key Chaining, requires the TSF to maintain a key chain and specifies the characteristics of that chain.

Management: FCS_KYC_EXT.1

There are no management activities foreseen.

Audit: FCS_KYC_EXT.1

There are no auditable events foreseen.

FCS_KYC_EXT.1 Key Chaining

Hierarchical to: No other components

Dependencies: No dependencies

FCS_KYC_EXT.1.1 The TSF shall maintain a chain of intermediary keys originating from the authorisation data submask and ending in the DEK using the following method(s):

- authorisation factor submask as specified in **[assignment:** ***specification of method*****]**
- key wrapping as specified in **[assignment:** ***specification of method*****]**

[selection:

- ***intermediate key generation as specified in [assignment: specification of method]***
- ***key encryption as specified in [assignment: specification of method]***
- ***key wrapping as specified in [assignment: specification of method]***
- ***no others*****]**

resulting in a key size of **[selection:** ***128 bits or 256 bits*****]**.

Application Note 44

An example of completion of the 'specification of methd' assignments would be to identify an appropriate SFR in the PP (e.g. FCS_CKM.3 for key wrapping).

Application Note 45

Key chaining is the method of using multiple layers of encryption keys to ultimately secure the protected user data on the TOE. The number of intermediate keys will vary. This applies to all keys that contribute to the ultimate wrapping or derivation of the DEK.

The ST Authors select appropriate iterations of other FCS requirements to specify each step in the key chain: it is allowable for an implementation to use multiple methods. One instance of the key wrapping function will always be mandatory in order to wrap the DEK with the KEK. However the first selection in FCS_KYC_EXT.1.1 also allows further iterations to be specified in case different key wrapping functions are used at other points in the product-specific key chaining.

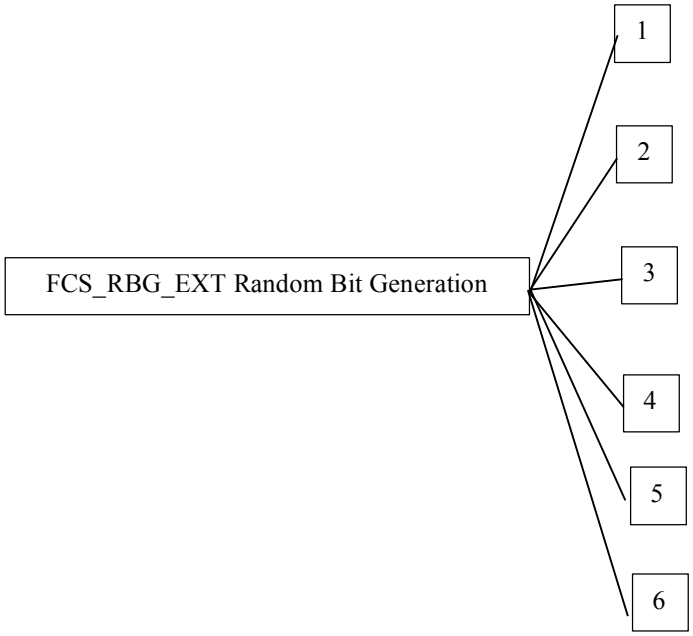
If 'no others' is applicable in the first selection then the ST author may simply omit the 3rd bullet in this list.

C.1.3 Random Bit Generation (FCS_RBG_EXT)

Family Behaviour

Components in this family address the requirements for random bit/number generation.

Component levelling



FCS_RBG_EXT.1 Random Bit Generation requires random bit generation to be performed in accordance with selected standards.

FCS_RBG_EXT.2 Random Bit Generation seeded by an external (outside the TOE) entropy source.

FCS_RBG_EXT.3 Random Bit Generation seeded by a TSF entropy source.

FCS_RBG_EXT.4 Random Bit Generation seeded by multiple TSF entropy sources.

FCS_RBG_EXT.5 Combining entropy sources – combining multiple entropy sources (multiple internal sources, internal and external).

FCS_RBG_EXT.6 Random Bit Generation Service requires random numbers to be supplied over an external interface as a service to other entities.

Management: FCS_RBG_EXT.1, FCS_RBG_EXT.2, FCS_RBG_EXT.3, FCS_RBG_EXT.4, FCS_RBG_EXT.5, FCS_RBG_EXT.6

There are no management activities foreseen.

Audit: FCS_RBG_EXT.1, FCS_RBG_EXT.2

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: failure of the randomization process, failure to initialize or reseed (as supported by the technology).

Audit: FCS_RBG_EXT.3, FCS_RBG_EXT.4, FCS_RBG_EXT.5, FCS_RBG_EXT.6

There are no auditable events foreseen.

FCS_RBG_EXT.1 Random Bit Generation (RBG)

Hierarchical to: No other components.

Dependencies: [FCS_RBG_EXT.2 Random Bit Generation (External Seeding), or FCS_RBG_EXT.3 Random Bit Generation (Internal Seeding Single Source)]

FPT_FLS.1 Failure with preservation of secure state

FPT_TST.1 TSF Testing

Application Note 46

No rationale is acceptable for not satisfying one of these dependencies.

FCS_RBG_EXT.1.1 The TSF shall perform deterministic random bit generation services using **[assignment: *RBG algorithm***] in accordance with **[assignment: *list of standards*]** after initialization with a seed.

FCS_RBG_EXT.1.2 The TSF shall use a **[selection: *TSF noise source* [assignment: *name of noise source*], *TSF interface for seeding*]** for initialized seeding.

FCS_RBG_EXT.1.3 The TSF shall update the RBG state using a noise source in the following situations: **[selection:**

- ***never;***
- ***the TSF shall reseed the RBG [selection: on demand; on the condition: [assignment: condition]; after [assignment: time]] in accordance with [assignment: list of standards]* using a [selection: *TSF noise source* [assignment: *name of noise source*], *TSF interface for seeding*];**
- ***the TSF shall uninstantiate and instantiate the RBG [selection: on demand; on the condition: [assignment: condition]; after [assignment: time]] in accordance with [assignment: list of standards]* using a [selection: *TSF noise source* [assignment: *name of noise source*], *TSF interface for seeding*].**

]

Application Note 47

If reseeding is not feasible, the TSF will uninstantiate RBGs, rather than produce output that is of insufficient quality. The listed standards should specify the reseed interval, and procedure

for uninstantiating and reseeding. The remaining selection allows the PP Author to require application-specific conditions for reseeding.

"Uninstantiate" means that the internal state of the DRBG is no longer available for use.

In the second selection, "on demand" means, that an interface to reseed is presented as a TSFI (e.g. an API call).

Health tests for the RBG are specified in FPT_TST.1.

FCS_RBG_EXT.2 Random Bit Generation (External Seeding)

Hierarchical to: No other components.

Dependencies: FCS_RBG_EXT.1

FCS_RBG_EXT.2.1 The TSF shall be able to accept a minimum input of **[assignment: *minimum input length greater than zero*]** from a TSF interface for the purpose of seeding.

Application Note 48

For this SFR, the interface to obtain the entropy noise source can be used multiple times to provide input. For instance, if the input length is 128 bits, it could be used twice to gather 256 bits. In this instance, the 128 bits would not be provided to the DRBG, since the DRBG can only be instantiated once, rather a function would gather the 128 bits twice and provide the DRBG with 256 bits of entropy noise source.

This SFR does not describe requirements on seed quality: it is the responsibility of the operational environment to define their requirement in this regard and to ensure that it is met by the external source.

Guidance in the introduction to PP/ST authors should address protection from modification and disclosure of the value from the external noise source, as well as the leaking of any pertinent information (e.g., internal state) regarding the RBG.

FCS_RBG_EXT.3 Random Bit Generation (Internal Seeding Single Source)

Hierarchical to: No other components.

Dependencies: FCS_RBG_EXT.1

FCS_RBG_EXT.3.1 The TSF shall be able to seed the RBG using a **[selection: choose one of: *TSF software-based noise source, TSF hardware-based noise source*] [assignment: *name of noise source*]** with a minimum of **[assignment: *number of bits*]** bits of min-entropy.

Application Note 49

If an ST Author wishes to use multiple internal noise sources, they iterate this requirement for each noise source being used by the TSF.

Hardware-based noise sources are sources whose primary function is noise generation, such as ring oscillators, diodes, and thermal noise. While software is used to collect the noise from these hardware sources, these are not software-based. Software-based noise sources are those sources that have some other primary function and the noise is a byproduct of their normal

operation. Examples of software-based noise sources are user or system based events, reading the least significant bits from an event timer, etc.

Hardware-based noise sources may be stochastically modelled, in which case the amount of entropy is well understood. Software-based noise sources are usually less well understood and therefore will typically take a more conservative approach, gathering larger numbers of bits than required and then performing a compression function to derive the final output. Software-based noise sources often rely on an entropy estimator.

FCS_RBG_EXT.4 Random Bit Generation (Internal Seeding Multiple Sources)

Hierarchical to: No other components.

Dependencies: FCS_RBG_EXT.1

FCS_RBG_EXT.5

FCS_RBG_EXT.4.1 The TSF shall be able to seed the RBG using **[selection: *[assignment: number] TSF software-based noise source(s), [assignment: number] TSF hardware-based noise source(s)*]**.

FCS_RBG_EXT.5 Random Bit Generation (Combining Noise Sources)

Hierarchical to: No other components.

Dependencies: FCS_RBG_EXT.1 Random Bit Generation (RBG)

[FCS_RBG_EXT.2 Random Bit Generation (External Seeding), or FCS_RBG_EXT.3 Random Bit Generation (Internal Seeding Single Source), or
FCS_RBG_EXT.4 Random Bit Generation (Internal Seeding multiple sources)]

FCS_RBG_EXT.5.1 The TSF shall **[assignment: *combining operation*] [selection: *output from TSF noise source(s), input from TSF interface(s) for seeding*]** to create the entropy input into the derivation function as defined in **[assignment: *list of standards*]**, resulting in a minimum of **[assignment: *number of bits*]** bits of min-entropy.

FCS_RBG_EXT.6 Random Bit Generation (RBG Services)

Hierarchical to: No other components.

Dependencies: FCS_RBG_EXT.1 Random Bit Generation (RBG)

[FCS_RBG_EXT.2 Random Bit Generation (External Seeding), or FCS_RBG_EXT.3 Random Bit Generation (Internal Seeding Single Source)]

FCS_RBG_EXT.6.1 The TSF shall provide a **[selection: *hardware, software, [assignment: other interface type]*]** interface to make the RBG output, as specified in FCS_RBG_EXT.1, available as a service to entities outside of the TOE.

C.1.4 Cryptographic Salt Generation (FCS_SLT_EXT)

Family Behaviour

This family ensures that salts are generated with suitable randomness.

Component levelling

FCS_SLT_EXT.1 Cryptographic Salt Generation requires the generation of salts to be performed in the specified manner.

Management: FCS_SLT_EXT.1

There are no management activities foreseen.

Audit: FCS_SLT_EXT.1

There are no auditable events foreseen.

FCS_SLT_EXT.1 Cryptographic Salt Generation

Hierarchical to: No other components

Dependencies: No dependencies

FCS_SLT_EXT.1.1 The TSF shall only use salts that are generated **[assignment: *method of generating salts*]**.

Application Note 50

An example of completion of the assignment would be: "by an RBG as specified in FCS_RBG_EXT.1".

C.2 User Data Protection (FDP)

C.2.1 Protection of User Data on Device (FDP_UDD_EXT)

Family Behaviour

This family defines requirements on the TSF to protect the user data stored on the device.

Component levelling

FDP_UDD_EXT.1 Protection of User Data on Device, requires the TSF to encrypt all user data stored on the device without user intervention**.**

Management: FDP_UDD _EXT.1

There are no management activities foreseen.

Audit: FDP_UDD_EXT.1

There are no auditable events foreseen.

FDP_UDD_EXT.1 Protection of User Data on Device

Hierarchical to: No other components

Dependencies: FCS_COP.1/UDE Cryptographic Operation (AES User Data Encryption/Decryption)

FDP_UDD_EXT.1.1 The TSF shall encrypt all user data under the DEK without user intervention, in accordance with FCS_COP.1/UDE.

Application Note 51

The intent of this requirement is to specify that encryption of any protected data will not depend on a user electing to protect that data, and to show that all user data encryption depends on access to the DEK and therefore requires prior successful authorisation (in order to obtain the DEK via the key chain) to start the session. The drive encryption specified in FDP_UDD_EXT.1 occurs transparently to the user and the decision to protect the data is outside the discretion of the user, which is a characteristic that distinguishes it from file encryption. The definition of user data can be found in the glossary.

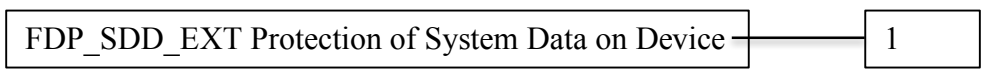
The TOE provides the cryptographic functions to encrypt/decrypt the user data as specified in FCS_COP.1/UDE.

C.2.2 Protection of System Data on Device (FDP_SDD_EXT)

Family Behaviour

This family defines requirements on the TSF to protect the system data stored on the device.

Component levelling



FDP_SDD_EXT.1 Protection of System Data on Device, requires the TSF to limit access to System Data to authorised actions and updates**.**

Management: FDP_SDD _EXT.1

There are no management activities foreseen.

Audit: FDP_SDD_EXT.1

There are no auditable events foreseen.

FDP_SDD_EXT.1 Protection of System Data on Device

Hierarchical to: No other components

Dependencies: No dependencies

FDP_SDD_EXT.1.1 The TSF shall allow changes to system data only in the form of changes to configuration data resulting from TSF actions or actions by authorised users **[selection: *or a trusted update in accordance with FPT_TUD_EXT.1, none*]**.

Application Note 52

The intent of this requirement is to specify that system data can only change in ways that reflect the legitimate use of the deveice by authorised users, or the application of a trusted update according to the FPT_TUD_EXT.1 requirements. Since system data is product-specific, it is recognised that normal operation of the TOE will result in updates to some of the data (e.g. if the DEK is regenerated, or if a data recovery mechanism is disabled, or to update DRBG state), and this is what is meant by the phrase "changes to system data only in the form of changes to configuration data resulting from TSF actions or actions by authorised users".

Note that if FPT_TUD_EXT.1 is chosen in the selection then this SFR has a dependency on FPT_TUD_EXT.1.

If 'none' is chosen in the selection then the selection may be left blank by the ST author.

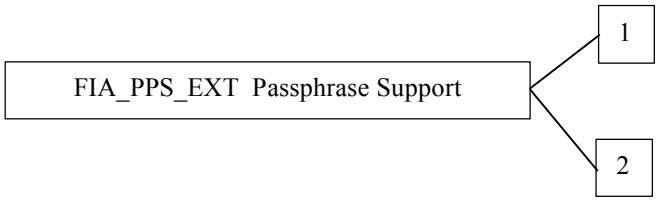
C.3 Identification and Authentication (FIA)

C.3.1 Passphrase Support (FIA_PPS_EXT)

Family Behaviour

This family defines requirements on the TSF to support passwords with varying composition requirements, e.g. the character set, and minimum and maximum supported passphrase lengths.

Component levelling



FIA_PPS_EXT.1 determines the possible interfaces the TSF uses to receive passphrases.

FIA_PPS_EXT.2 specifies the characteristics of passphrases entered by the interfaces in FIA_PPS_EXT.1 (in terms of passphrase composition, minimum length and maximum length). Multiple iterations of FIA_PPS_EXT.2 are used if the TOE supports multiple interfaces with different characteristics.

Management: FIA_PPS _EXT.1, FIA_PPS _EXT.2

There are no management activities foreseen.

Audit: FIA_PPS _EXT.1, FIA_PPS _EXT.2

There are no auditable events foreseen.

FIA_PPS_EXT.1 Passphrase entry interface

Hierarchical to: No other components

Dependencies: No dependencies

FIA_PPS_EXT.1.1 The TSF shall support **[assignment: *alternative methods of passphrase entry*].**

Application Note 53

The assignment in this SFR is intended to be turned into a selection by the PP author, with each alternative value of the selection identifying a further member of FIA_PPS_EXT that specifies the constraints on composition and maximum length for the relevant method of entry. The SFRs identified in this way become dependencies of FIA_PPS_EXT.1.

FIA_PPS_EXT.2 Passphrase composition

Hierarchical to: No other components

Dependencies: FIA_PPS_EXT.1

FIA_PPS_EXT.2.1 The TSF shall support passphrases that consist of: **[assignment: *supported characters*]**

FIA_PPS_EXT.2.2 The TSF shall provide a mechanism to verify that passphrases have a minimum length of **[assignment: *number and unit*]**.

FIA_PPS_EXT.2.3 The TSF shall support passphrases with a maximum length of at least **[assignment: *number and unit*]**.

Application Note 54

This SFR is iterated in a PP or ST to cover each of the alternative methods of passphrase entry to support FIA_PPS_EXT.1.

C.4 Protection of the TSF (FPT)

C.4.1 Protection of Keys and Keying Material (FPT_KYP_EXT)

Family Behaviour

This family defines requirements on the TSF to protect the keys and keying material when written to non-volatile memory.

Component levelling

FPT_KYP_EXT Protection of Keys and Keying Material	1

FPT_KYP_EXT.1 Protection of Keys and Keying Material, requires the TSF to ensure that no plaintext keys or keying material are written to non-volatile memory.

Management: FPT_KYP_EXT.1

There are no management activities foreseen.

Audit: FPT_KYP_EXT.1

There are no auditable events foreseen.

FPT_KYP_EXT.1 Protection of Keys and Keying Material

Hierarchical to: No other components

Dependencies: FCS_CKM.3 Cryptographic Key Access (Key Wrapping)
FCS_COP.1/KeyEnc Cryptographic operation (Key Encryption)

FPT_KYP_EXT.1.1 The TSF shall only store keys and keying material in non-volatile memory when wrapped, as specified in FCS_CKM.3, or encrypted, as specified in FCS_COP.1/KeyEnc unless the keys or keying material meets any one of following criteria:

- The plaintext keys or keying material is not part of the key chain as specified in FCS_KYC_EXT.1.
- The plaintext keys or keying material will no longer provide access to the encrypted data after initial provisioning**.**

Application Note 55

The ST author needs to use FCS_CKM.3/Chain and/or FCS_COP.1/KeyEnc from Appendix B (Selection-based security requirements) for keys and keying material that are stored in non-volatile memory (in addition to the use of FCS_CKM.3/DEK for the wrapping of the DEK with the KEK). The ST Author may need to iterate this requirement if different methods are used to wrap and/or encrypt various keys and keying material..

C.4.2 Trusted Update (FPT_TUD_EXT)

Family Behaviour

Components in this family address the requirements for updating the TOE firmware and/or software.

Component levelling

FPT_TUD_EXT.1 Trusted Update, requires the TSF to have the capability to update the TOE firmware/software, including the ability to verify the updated prior to installation.

Management: FPT_TUD _EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to update the TOE and to verify the updates.

Audit: FPT_TUD _EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the cPP/ST:

Minimal: Initiation of the update process

Any failure to verify the integrity of the update

FPT_TUD_EXT.1 Trusted Update

Hierarchical to: No other components

Dependencies: FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)
FCS_COP.1/SigVer Cryptographic Operation (Signature Verification)
FCS_COP.1/HMAC Cryptographic Operation (Keyed Hash)

FPT_TUD_EXT.1.1 The TSF shall provide authorised users with the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The TSF shall provide authorised users with the ability to initiate updates to TOE firmware/software.

FPT_TUD_EXT.1.3 The TSF shall verify updates to the TOE firmware/software using a **[selection: *digital signature mechanism, published hash, keyed hash*]** by the manufacture prior to installing those updates.

C.4.3 Trusted Update Rollback (FPT_TUR_EXT)

Family Behaviour

Components in this family address the requirements for allowing the rollback of previous updates to the TOE firmware and/or software.

Component levelling

FPT_TUR_EXT.1 Trusted Update Rollback, requires the TSF to have the capability to rollback previously applied updates to the TOE firmware and/or software.

Management: FPT_TUR_EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to update the TOE and to verify the rollback of updates.

Audit: FPT_TUR_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the cPP/ST:

Minimal: Initiation of the update rollback process

Any failure to verify the integrity of the update

FPT_TUR_EXT.1 Trusted Update Rollback

Hierarchical to: No other components

Dependencies: No dependencies

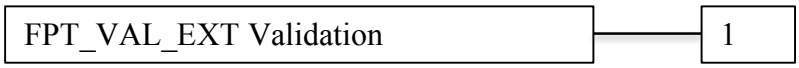
FPT_TUR_EXT.1.1 The TSF shall not allow rollback of previously applied updates.

C.4.4 Validation (FPT_VAL_EXT)

Family Behaviour

This family requires the TSF to provide a method to validate submasks that are provided as an input to a cryptographic function or cryptographic primitive acting as one part of a chain of cryptographic functions that calculates a cryptographic key as the end results of the chain.

Component levelling



FPT_VAL_EXT.1 Validation, requires the TSF to validate submasks by one or more of the specified methods.

Management: FPT_VAL_EXT.1

There are no management activities foreseen.

Audit: FPT_VAL_EXT.1

There are no auditable events foreseen.

FPT_VAL_EXT.1 Validation

Hierarchical to: No other components

Dependencies: FCS_CKM.3 Cryptographic Key Access (Key wrapping)
FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)
FCS_COP.1/HMAC Cryptographic Operation (Keyed Hash)

FPT_VAL_EXT.1.1 The TSF shall perform validation of the submask using the following methods: **[selection:**

- ***key wrap as specified in FCS_CKM.3,***
- ***hash the authorisation data submask as specified in [selection: FCS_COP.1/Hash, FCS_COP.1/HMAC] and compare it to a stored hashed value of the authorisation data,***
- ***decrypt a known value using the submask as specified in FCS_COP.1/KeyEnc and compare it against a stored known value*****].**

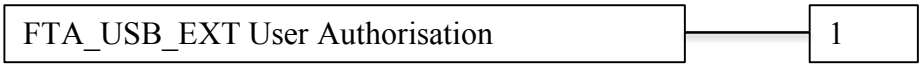
C.5 TOE Access (FTA)

C.5.1 User Authorisation (FTA_USB_EXT)

Family Behaviour

This family defines requirements for session establishment and termination.

Component levelling



FTA_USB_EXT.1 User Authorisation, requires the user to present a valid passphrase to establish/initiate a session.

Management: FTA_USB_EXT.1

The following actions could be considered for the management functions in FMT:

- if an authorized administrator could request re-authorisation, the management includes a re-authorisation request;
- management of the conditions under which re-authorisation is required.

Audit: FTA_USB_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the cPP/ST:

Minimal: Failure of re-authorisation;

Basic: All re-authorisation attempts.

FTA_USB_EXT.1 User Authorisation

Hierarchical to: No other components

Dependencies: No dependencies

FTA_USB_EXT.1.1 The TSF shall require that a valid passphrase is supplied before starting a session that allows any access to user data on the TOE.

FTA_USB_EXT.1.2 The TSF shall authorise the user under the following conditions:

- connection of the TOE to a host device
- recovery of a host device from a power-down or sleep state while the TOE is connected to it
- recovery of the TOE from its own power-down or sleep state
- changing the value of authorisation data
- **[assignment:** ***list of other conditions under which re-authorisation is required*****]**.

FTA_USB_EXT.1.3 The TSF shall allow host-initiated termination of the current session.

D. Required Supplementary Information

This appendix describes the required supplementary information for the USB portable storage device TOE. Required Supplementary Information refers to information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. This information typically extends the level of detail of particular areas of TOE design in selective areas that are considered necessary to give the intended level of value from evaluation of certain SFRs or SARs in the TOE. Although it may be presented in a separate document, in logical terms the information is treated as an extension of the TSS (this represents an exception to the general principle in [CEM, para 500] that "the descriptions …[in the TSS]… should not be overly detailed").

D.1 Entropy Documentation and Assessment

This is an optional appendix in the cPP, and only applies if the TOE is providing an internal entropy source.

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

The documentation of the entropy source(s) 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.

D.1.1 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.

D.1.2 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.

D.1.3 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. Similarly, documentation shall describe the conditions under which the entropy source is no longer guaranteed to provide sufficient entropy. Methods used to detect failure or degradation of the source shall be included.

D.1.4 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, TOE behavior upon entropy source failure, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

D.2 Key Management and Data Storage Description

In order to support the evaluation of the SFRs, some additional information is required to describe the keys and datapaths that implement the SFRs and the ways that the 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 and Data Storage Description' or 'KMDSD', and is specified below.

The Key Management and Data Storage Description should combine diagrams and text, and should be detailed enough that, after reading, the evaluator will thoroughly understand the keys and datapaths 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 separate Key Management and Data Storage subsections below give more detailed descriptions of the topics to be addressed in the KMDSD. The topics may not apply to all products, but where a requirement does not apply then the KMDSD should include an explanation as to why the details do not apply.

D.2.1 Key Management

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

- The purpose of the key (indicating its relationship to the Reference Key Model (see section 5.3), 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 and Data Storage 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.

- A clear justification that at no point in the key hierarchy 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. whether encryption hardware is part of the main SOC or in a separate co-processor, and identification of any parts of the encryption process carried out in firmware rather than hardware). 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 (including self-tests), the encryption initialisation process, and at what moment the device enables the encryption.
- The process for destroying keys and the authorisation data input when they are no longer needed by the device, including the type of storage location of all keys (and authorisation data) and the destruction method for that storage.

The Key Management and Data Storage Description diagrams shall identify all keys used, starting at the conditioning of authorisation data and continuing until the unwrapped DEK is reached. The KMDSD 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. The KMDSD 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.

An example of part of a KMDSD is given in Figure 10 (this diagram and the notations used are illustrative only: there is no prescribed form for the KMDSD). The diagram would also need to be accompanied by suitable explanatory text (including references to appropriate standards and links to the individual SFRs in the Security Target that capture each of the operations).

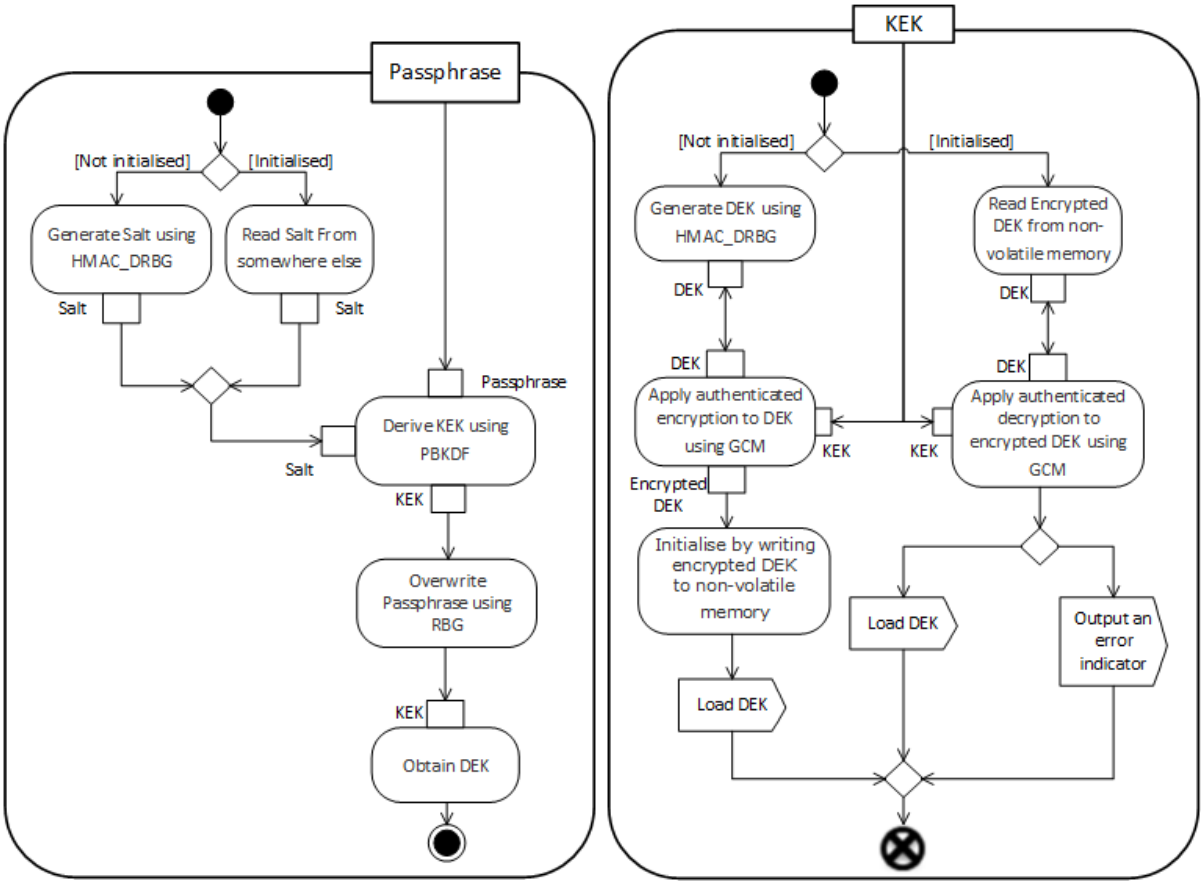


Figure 10: Example diagram for part of Key Management Description

D.2.2 Data Storage

The supplied documentation shall include the design of the data paths within the TOE for at least the following critical data items:

- User data written by the host to the TOE
- User data read by the host from the TOE
- The plaintext DEK
- The plaintext KEK
- Authentication data (received either from the host or, if available, from a point of entry on the TOE)

The documentation shall describe the data paths for each data item during all uses of the data item (indicating the part of the TOE that controls each step of the data flow). For example, the data path for user data written by the host shall cover from the arrival of the user data from the host at the TOE boundary to the point at which it is stored in its final destination in the TOE. The documentation of the data path should be detailed enough that the evaluator will

thoroughly understand the parts of the TOE that the data passes through (e.g. different memory types, processors and co-processors), its encryption state (i.e. encrypted or unencrypted) in each part, and any places where the data is stored. For example, any caching or buffering of the data should be identified and distinguished from the final destination in non-volatile memory (the latter represents the location from which the host will expect to retrieve the data in future).

For the DEK, the documentation shall describe the connection to the DEK decryption process (i.e. unwrapping of the DEK using the KEK), and to the user data encryption/decryption process, as well as the location of the plaintext DEK during an authenticated session. For the KEK, the datapath shall describe the connection to KEK calculation process (i.e. the calculation of the KEK from the Authentication Data received from the host), the connection to the DEK decryption process, and the location of the plaintext KEK during an authenticated session.

The design information shall also describe:

- any other situations where user data may be read and stored in other parts of the TOE (e.g. as part of a caching or look-ahead strategy)

- the effect of memory failures on the data path (e.g. where bad sectors are discovered and removed from the available physical storage locations)

- a rationale for why no stored unencrypted user data can survive beyond the session in which it is written and/or read.

E. 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
Intermediate Key	A key used in a point between the initial user authorisation and the KEK.
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.
System Data	System data may include (but are not limited to) software or firmware, patches, or configuration data.
Submask	A submask a bit string that is provided as an input to a cryptographic function or cryptographic primitive acting as one part of a chain of cryptographic functions that calculates a cryptographic key as the end result of the chain. Examples of submasks include: master keys, intermediate keys, wrapping keys, secret bit strings used for authentication or authorisation, and conditioned passphrases.
Plaintext user data	User data stored on a device that needs to be protected against the unauthorised disclosure.
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 [CC1] for other Common Criteria abbreviations and terminology.

F. Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CCM	Counter with CBC-Message Authentication Code
cPP	Collaborative Protection Profile
DEK	Data Encryption Key
DRBG	Deterministic Random Bit Generator
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
iTC	international Technical Community
KDF	Key Derivation Function
KEK	Key Encryption Key
MBR	Master Boot Record
NIST	National Institute of Standards and Technology
PBKDF	Passphrase-Based Key Derivation Function
PP	Protection Profile
RBG	Random Bit Generator
RSA	Rivest Shamir Adleman Algorithm
SD	Supporting Document
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SP	NIST Special Publication
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

G. Index of Application Notes