Supporting Document
Mandatory Technical Document

USB Portable Storage Device

October 2018

Version 0.7

CCDB-<*Reference from CCDB, in the form 'YYYY-MM-nnn'*>

Document Purpose and Overview

This document serves as a template Supporting Document (SD) for use by international Technical Communities (iTCs). SDs are complementary to collaborative Protection Profiles (cPPs) and define the evaluation activities (EAs) required to satisfy the Security Functional Requirements (SFRs) in the cPP.

The intent of this SD template is to provide iTCs with a method for developing SDs that address the CEM work units. International Technical Communities are expected to tailor this template to appropriately address the technology type in question.

Sections of this document contain notes in <*brackets and italics*> for the SD author to take into consideration when developing the SD and should be removed by the iTC prior to finalizing the SD.

Similarly, some sections of this template are populated with examples to illustrate the structure of the section. They are expected to be replaced by the specific EAs deemed necessary by the iTC.

Foreword

This is a supporting document, intended to complement the Common Criteria version 3 and the associated Common Evaluation Methodology for Information Technology Security Evaluation.

Supporting documents may be "Guidance Documents", that highlight specific approaches and application of the standard to areas where no mutual recognition of its application is required, and as such, are not of normative nature, or "Mandatory Technical Documents", whose application is mandatory for evaluations whose scope is covered by that of the supporting document. The usage of the latter class is not only mandatory, but certificates issued as a result of their application are recognized under the CCRA.

This supporting document has been developed by the USB iTC and is designed to be used to support the evaluations of products against the cPPs identified in section 1.1.

Technical Editor:

USB iTC

Document history:

V0.7, 25 October 2018 (updating evaluation activities for crypto SFRs)

V0.5, 1 April 2016 (updating template, adding information about the AVA)

V0.4, 17 December 2015 (updating template, addressing review comments on v0.2)

V0.3, August 2015 (interim updates – not released)

V0.2, July 2015 (Initial draft for comment)

General Purpose:

USB Portable Storage Devices are ubiquitous data storage solutions used in a variety of capacities and form factors. As portable devices, their primary functionality is to encrypt and protect data-at-rest stored on the device.

In order to ensure comparable, transparent, and repeatable evaluation of the implemented cryptographic mechanisms, methods have to be described that may consist of agreed evaluation approaches, e.g. how to prove that the claimed encryption of user data is really done by the TOE or how to prove that the user data is only stored in an encrypted form (and not additionally in clear text), but also the definitions of possibly necessary special test tools and their manuals.

Field of special use:

USB Portable Storage Device

Acknowledgements:

This Supporting Document was developed by the USB international Technical Community with representatives from industry, Government agencies, Common Criteria Test Laboratories, and members of academia <*check: do we have any academia?*>.

Contents

List of figures

No table of figures entries found.

List of tables

1 Introduction

1.1 Technology Area and Scope of Supporting Document

1 This Supporting Document (SD) is mandatory for evaluations of products that claim conformance to any of the following cPP(s):

> *Collaborative Protection Profile for USB Portable Storage Devices – [Month, Year]*

2 The purpose of the collaborative Protection Profile (cPP) for USB Portable Storage Devices is to provide a minimal set of security requirements that provide protection against a set of defined threats. The specific form factor of a USB Portable Storage Device is not defined, however devices that use a USB interface to store data on another form of storage (such as an external optical drive writing to a CD or DVD) or more complex device (such as a tablet or smartphone) are outside the scope of the cPP.

3 A USB Portable Storage Device is dedicated to storing user data, and protecting that data using specified cryptographic protocols. System data, such as device driver software or configuration data is considered separate from user data and may reside on the device in an unencrypted state.

4 Although Evaluation Activities (EAs) are defined mainly for the evaluators to follow, in general they will also help developers prepare for evaluation by identifying specific requirements for their Target of Evaluation (TOE). The specific requirements in EAs may in some cases clarify the meaning of Security Functional Requirements (SFRs), and may identify particular requirements for the content of Security Targets (especially the TOE Summary Specification), user guidance documentation, and possibly required supplementary information (e.g. for entropy analysis or cryptographic key management architecture).

1.2 Structure of the Document

5 Evaluation Activities can be defined for both SFRs and Security Assurance Requirements (SARs). These are defined in separate sections of this SD. The EAs associated with the SFRs are considered to be interpretations of applying the appropriate SAR activity. For instance, activities associated with testing are representative of what is required by ATE_IND.1.

6 If any Evaluation Activity cannot be successfully completed in an evaluation then the overall verdict for the evaluation is a ‘fail’. In rare cases, there may be acceptable reasons why an Evaluation Activity may be modified or deemed not applicable for a particular TOE, but this must be agreed with the Certification Body for the evaluation.

7 In general, if all EAs (for both SFRs and SARs) are successfully completed in an evaluation then it would be expected that the overall verdict for the evaluation is a 'pass'.

8 In some cases, the Common Evaluation Methodology (CEM) work units have been interpreted to require the evaluator to perform specific EAs. In these instances, EAs will be specified in Section 2 (*),* Section 5 (*),* and possibly Section 3 (Evaluation Activities for Optional Requirements*) and Section 4 (Evaluation Activities for Selection-Based Requirements)*. In cases where there are no CEM interpretations, the CEM activities are to be used to determine if SARs are satisfied and references to the CEM work units are identified as being the sole EAs to be performed.

9 Finally, there are cases where EAs have rephrased CEM work units to provide clarity on what is required. The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. In these cases, the EA supplements the CEM work unit. These EAs will be specified in Section 5 (*)*.

1.3 Terminology

1.3.1 Glossary

10 For definitions of standard CC terminology, see [CC] part 1.

Term	Meaning
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
Data Encryption Key (DEK)	A key used to encrypt data-at-rest.
Error State	The device has failed a self-test and could not reset
Key Chaining	The method of using multiple layers of encryption keys to protect data. A top layer key encrypts a lower layer key which encrypts the data; this method can have any number of layers.
Key Encryption Key (KEK)	A key used to encrypt other keys, such as DEKs or storage that contains keys.
Keying Material	A data item that is used in combination with other data in order to derive a cryptographic key (e.g. a passphrase, seed, or each of the values used in an xor combination).
Passphrase Authorisation Factor	A type of authorisation factor requiring the user to provide a secret set of characters to gain access.
Powered-Off State	The device has been shutdown.
Required Supplementary Information	Information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP (see description in Section 6).

Term	Meaning
Submask	A submask is 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.
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.

1.3.2 Acronyms

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

2 Evaluation Activities for SFRs

11 The EAs presented in this section capture the actions the evaluator performs to address technology specific aspects covering specific SARs (e.g.., ASE_TSS.1, ADV_FSP.1, AGD_OPE.1, and ATE_IND.1) – this is in addition to the CEM work units that are performed in Section 5 (*)*.

12 Regarding design descriptions (designated by the subsections labelled TSS, as well as any required supplementary material that may be treated as proprietary), the evaluator must ensure there is specific information that satisfies the EA. For findings regarding the TSS section, the evaluator's verdicts will be associated with the CEM work unit ASE_TSS.1-1. Evaluator verdicts associated with the supplementary evidence will also be associated with ASE_TSS.1-1, since the requirement to provide such evidence is specified in ASE in the cPP.

13 For ensuring the guidance documentation provides sufficient information for the administrators/users as it pertains to SFRs, the evaluator's verdicts will be associated with CEM work units ADV_FSP.1-7, AGD_OPE.1-4, and AGD_OPE.1-5.

14 Finally, the subsection labelled Tests is where the iTC has determined that testing of the product in the context of the associated SFR is necessary. While the evaluator is expected to develop tests, there may be instances where it is more practical for the developer to construct tests, or where the developer may have existing tests. Therefore, it is acceptable for the evaluator to witness developer-generated tests in lieu of executing the tests. In this case, the evaluator must ensure the developer's tests are executing both in the manner declared by the developer and as mandated by the EA. The CEM work units that are associated with the EAs specified in this section are: ATE_IND.1-3, ATE_IND.1-4, ATE_IND.1-5, ATE_IND.1-6, and ATE_IND.1-7.

2.1 Cryptographic Support (FCS)

2.1.1 Introduction

15 This section defines the Evaluation Activities associated with the cryptographic requirements included in the collaborative Protection Profile for Portable Storage Devices. This document defines three types of Evaluation Activities (EAs) – TOE Summary Specification (TSS), Guidance Documentation, and Tests and is designed to be used in conjunction with the "cPP for Portable Storage Devices Cryptographic SFR Instantiation". The security requirement naming convention is consistent between these documents ensuring a clear one to one correspondence between the security requirements and evaluation activities.

2.1.1.1 Application of the Evaluation Activity document

16 In the cryptographic SFRs, several operations need to be performed (mainly selections and assignments). As a result, the EAs may define separate actions for different selected or assigned values in SFRs. The evaluator shall neither carry out EAs related to SFRs that are not claimed in the Security Target nor EAs related to specific selected or assigned values that are not claimed in the Security Target.

17 In addition, EAs do not necessarily have to be executed independently from each other. A description in a guidance documentation or one test case, for example, can cover multiple EAs at a time, no matter whether the EAs are related to the same or different SFRs.

2.1.1.2 Evaluation Activity Notes applicable to all SFRs

18 When an SFR (the 'dependent SFR') identifies other cryptographic SFRs that it depends on, then the evaluator shall confirm that the ST includes those other SFRs, with relevant selections as appropriate for the dependent SFR, and that the TSS identifies that those SFRs are used for the implementation of the dependent SFR. For example, where key derivation functions in FCS_CKM_EXT.5 include selections for pseudorandom functions using HMAC and AES then the evaluator would check that the ST includes FCS_COP.1 iterations for the relevant HMAC and AES operations, including corresponding key lengths and modes. The evaluator would also check that the TSS specifies that these FCS_COP.1 implementations are used in the implementation of the relevant aspects of FCS_CKM_EXT.5[1].

[1] The developer is thereby confirming the use of the evaluated cryptographic functionality for the dependent SFR. In many cases this will be a trivial confirmation, however in some cases multiple implementations of the primitive cryptographic operation may be available in the product and it is then important to establish that only the evaluated primitive is used for the dependent SFR.

2.1.2 Cryptographic Key Generation (FCS_CKM.1)

2.1.2.1 FCS_CKM.1/DEK Cryptographic key generation (DEK)

19 The following EAs apply for Identifier: DEK1.

2.1.2.1.1 TSS

20 The evaluator shall examine the TSS to verify that it describes how the TOE obtains a DEK through direct generation from a random bit generator as specified in FCS_RBG_EXT.1. The evaluator shall review the TSS to verify that it describes how the functionality described by FCS_RBG_EXT.1 is invoked.

2.1.2.1.2 Guidance Documentation

21 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key name(s) for all uses identified in the ST.

2.1.2.1.3 Key Management Description (KMD)

22 The evaluator shall confirm that the KMD describes:

- The RBG interface and how it is used in the key generation
- If the TOE uses the generated key in a key chain/hierarchy then the KMD shall describe how the key is used as part of the key chain/hierarchy.

2.1.2.1.4 Tests

23 For each selected key size, the evaluator shall configure the DEK generation capability. The evaluator shall use the description of the RBG interface to verify that the TOE requests and receives an amount of RBG output greater than or equal to the requested key size.

2.1.3 Cryptographic Key Access (FCS_CKM.3)

2.1.3.1 FCS_CKM.3/DEK Cryptographic key access (Key Wrapping)

2.1.3.1.1 TSS

24 The evaluator shall check that the TSS includes a description of the key wrap function(s) and shall check that this uses a key wrap algorithm and key sizes according to the specification selected in the ST out of the table as provided in the cPP table.

2.1.3.1.2 Guidance Documentation

25 The evaluator checks the AGD documents to confirm that the instructions for establishing the evaluated configuration use only those key wrap function(s) selected in the ST. If multiple key access modes are supported, the evaluator

shall examine the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

2.1.3.1.3 KMD

26 The evaluator shall examine the KMD to ensure that it describes when the key wrapping occurs, that the KMD description is consistent with the description in the TSS, and that for all keys that are wrapped the TOE uses a method as described in the cPP table. No uncertainty should be left over which is the wrapping key and the key to be wrapped and where the wrapping key potentially comes from i.e. is derived from.

27 If "KW3: AES-GCM" or "KW4: AES-CCM" is used the evaluator shall examine the KMD to ensure that it describes how the IV is generated and that the same IV is never reused to encrypt different plaintext pairs under the same key. Moreover in the case of GCM, he must ensure that, at each invocation of GCM, the length of the plaintext is at most (2^32)-2 blocks.

2.1.3.1.4 Tests

28 The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following tests or witness respective tests executed by the developer if technically possible, otherwise an analysis of the implementation representation has to be performed.

29 Preconditions for testing:

- Specification of wrapping keys as input parameter to the function to be tested
- Specification of further required input parameters if required
- Specification of keys to be wrapped (plaintext, as function's argument)
- Direct access to wrapped key (ciphertext), e.g. in the non-volatile memory

30 ***KW2: AES-KW [SP 800-38F, sec. 6.2]***

31 The tests below are derived from "The Key Wrap Validation System (KWVS), Updated: June 20, 2014" from the National Institute of Standards and Technology.

32 The evaluator shall test the authenticated-encryption functionality of AES-KW for each combination of the following input parameters:

- Supported key lengths selected in the ST (e.g. 128 bits, 256 bits)
- Five plaintext lengths:
    - Two lengths that are non-zero multiples of 128 bits (two semi-block lengths)

- Two lengths that are odd multiples of the semi-block length (64 bits)
- The largest supported plaintext length less than or equal to 4096 bits

33 For each set of the above parameters the evaluator shall generate a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall compare the results with those obtained from the AES-KW authenticated-encryption function of a known good implementation.

34 The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption (KW-AE) with AES-KW authenticated-decryption (KW-AD). For the authenticated-decryption test, 20 out of the 100 trials per plaintext length must have ciphertext values that are not authentic; that is, they fail authentication.

35 Additionally, the evaluator shall perform the following negative test:

- Test 1 (invalid plaintext length):

Determine the valid plaintext lengths of the implementation from the TOE specification. Verify that the implementation of KW-AE in the TOE rejects plaintexts of invalid length by testing plaintext of the following lengths: 1) plaintext length greater than 64 semi- blocks, 2) plaintext bit-length not divisible by 64, 3) plaintext with length 0, and 4) plaintext with one semi-block.

- Test 2 (invalid ciphertext length):

Determine the valid ciphertext lengths of the implementation from the TOE specification. Verify that the implementation of KW-AD in the TOE rejects ciphertexts of invalid length by testing ciphertext of the following lengths: 1) ciphertext with length greater than 65 semi-blocks, 2) ciphertext with bit-length not divisible by 64, 3) ciphertext with length 0, 4) ciphertext with length of one semi-block, and 5) ciphertext with length of two semi- blocks.

- Test 3 (invalid ICV1):

Test that the implementation detects invalid ICV1 values by encrypting any plaintext value eight times using a different value for ICV1 each time as follows: Start with a base ICV1 of 0xA6A6A6A6A6A6A6A6. For each of the eight tests change a different byte to a different value, so that each of the eight bytes is changed once. Verify that the implementation of KW-AD in the TOE outputs FAIL for each test.

36 ***KW1: AES-KWP [SP 800-38F, sec. 6.3]***

37 The tests below are derived from "The Key Wrap Validation System (KWVS), Updated: June 20, 2014" from the National Institute of Standards and Technology.

38 The evaluator shall test the authenticated-encryption functionality of AES-KWP (KWP-AE) using the same test as for AES-KW authenticated-encryption with the following change in the file plaintext lengths:

- Four lengths that are multiples of 8 bits
- The largest supported length less than or equal to 4096 bits

39 The evaluator shall test the authenticated-decryption (KWP-AD) functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption. For the Authenticated Decryption test, 20 out of the 100 trials per plaintext length have ciphertext values that fail authentication.

40 Additionally, the evaluator shall perform the following negative test:

- Test 1 (invalid plaintext length):

  Determine the valid plaintext lengths of the implementation from the TOE specification. Verify that the implementation of KW-AE in the TOE rejects plaintexts of invalid length by testing plaintext of the following lengths: 1) plaintext with length greater than 64 semi-blocks, 2) plaintext with bit-length not divisible by 8, and 3) plaintext with length 0.

- Test 2 (invalid ciphertext length):

  Determine the valid ciphertext lengths of the implementation from the TOE specification. Verify that the implementation of KWP-AD in the TOE rejects ciphertexts of invalid length by testing ciphertext of the following lengths: 1) ciphertext with length greater than 65 semi-blocks, 2) ciphertext with bit-length not divisible by 64, 3) ciphertext with length 0, and 4) ciphertext with length of one semi-block.

- Test 3 (invalid ICV2):

  Test that the implementation detects invalid ICV2 values by encrypting any plaintext value four times using a different value for ICV2 each time as follows: Start with a base ICV2 of 0xA65959A6. For each of the four tests change a different byte of ICV2 to a different value, so that each of the four bytes is changed once. Verify that the implementation of KWP-AD in the TOE outputs FAIL for each test.

- Test 4 (invalid padding length):

  Generate one ciphertext using algorithm KWP-AE with substring $[len(P)/8]_{32}$ of S replaced by each of the following 32-bit values, where

len(P) is the length of P in bits and [ ]$_{32}$ denotes the representation of an integer in 32 bits:

- $[0]_{32}$
- $[\text{len(P)/8-8}]_{32}$
- $[\text{len(P)/8-8}]_{32}$
- $[513]_{32}$

Verify that the implementation of KWP-AD in the TOE outputs FAIL on those inputs.

- Test 5 (invalid padding bits):

If the implementation supports plaintext of length not a multiple of 64-bits, then

for each PAD length [1..7]

for each byte in PAD

set a zero PAD value;

replace current byte by a non-zero value and use the resulting plaintext as input to algorithm KWP-AE to generate ciphertexts;

verify that the implementation of KWP-AD in the TOE outputs FAIL on this input.

41 ***KW3: AES-GCM [ISO 19772, clause 11]***

42 Refer to [cPP FCS_COP.1/UDE] for the required AES-GCM testing. Each distinct AES-GCM implementation shall be tested separately.

43 ***KW4: AES-CCM [ISO 19772, clause 8]***

44 Refer to [cPP FCS_COP.1/UDE] for the required AES-CCM testing. Each distinct AES-CCM implementation shall be tested separately.

2.1.4 Cryptographic Key Destruction (FCS_CKM.4)

2.1.4.1 FCS_CKM.4 Cryptographic key destruction

2.1.4.1.1 TSS

45 The evaluator examines the TSS to ensure it lists all relevant keys and keying material (describing 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)), all relevant destruction situations (including the point in time at which the destruction occurs; e.g. factory reset or device wipe function, change of authorisation data, change of DEK, completion of use of an intermediate key) and the destruction method used in each case. The evaluator confirms that the description of the data and

storage locations is consistent with the functions carried out by the TOE (e.g. that all keys in the key chain are accounted for[2]). This evaluation activity may be combined with those dealing with protection of keys and keying material in FPT_KYP_EXT.1.

46 The evaluator shall check that the TSS identifies any configurations or circumstances that may not conform to the key destruction requirement (see further discussion in the Operational Guidance section below). Note that reference may be made to the Guidance Documentation for description of the detail of such cases where destruction may be prevented or delayed.

47 Where the ST specifies the use of "a value that does not contain any sensitive data" to overwrite keys, the evaluator examines the TSS to ensure that it describes how that pattern is obtained and used, and that this justifies the claim that the pattern does not contain any sensitive data.

2.1.4.1.2 Guidance Documentation

48 The evaluator shall check that the guidance documentation for the TOE requires users to ensure that the TOE remains under the user's control while a session is active.

49 A TOE may be subject to situations that could prevent or delay data destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and KMD). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer, identifying any additional mitigation actions for the user (e.g. there might be some operation the user can invoke, or the user might be advised to retain control of the device for some particular time to maximise the probability that garbage collection will have occurred).

50 For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the data that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command[3] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and guidance documentation).

[2] Where keys are stored encrypted or wrapped under another key then this may need to be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions.

[3] Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table).

2.1.4.1.3 KMD

51 The KMD identifies and describes the interface(s) that are used to service commands to read/write memory. The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) made by the ST Author.

52 The evaluator examines the KMD to ensure that all keys and keying material identified in the TSS and KMD have been accounted for.

53 Note that where selections include 'destruction of reference to the key directly followed by a request for garbage collection' (for volatile memory) then the KMD is examined by the evaluator to ensure that it explains the nature of the destruction of the reference, the request for garbage collection, and of the garbage collection process itself.

2.1.4.1.4 Tests

54 Note: The following tests require the developer to provide access to a test platform that provides the evaluator with interfaces that are typically not found on factory products. The developer must describe the architecture of the test platform and give a rationale that it accurately exposes the TOE state without interfering with its intended operations.

55 ***[**USB iTC: to integrate the paragraph above and make it consistent with any other description of the test platform: the aim is to describe this in one place for all USB cPP SFRs]***

56 Test 1: *Applied to each key or keying material held as plaintext in volatile memory and subject to destruction by overwrite by the TOE (whether or not the plaintext value is subsequently encrypted for storage in volatile or non-volatile memory).*

57 The evaluator shall:

1. Record the value of the key or keying material.
2. Cause the TOE to dump the entire memory of the TOE into a binary file.
3. Search the content of the binary file created in Step #2 to locate all instances of the known key value from Step #1. (Note that the primary purpose of Step #3 is to demonstrate that appropriate search commands are being used for Step #8 and #9)
4. Cause the TOE to perform normal cryptographic processing with the key from Step #1.
5. Cause the TOE to destroy the key.
6. Cause the TOE to stop the execution but not exit.

7. Cause the TOE to dump the entire memory of the TOE into a binary file.
8. Search the content of the binary file created in Step #7 for instances of the known key value from Step #1.
9. ***[**USB iTC to decide whether to add an additional search for key fragments (based on the specific types of risk for the TOE type/deployment environments) and, if so, the relevant length/types of fragment to search for - e.g. fragment sizes might be based on implementation-level key storage structures. Example text follows - to be deleted or modified by the iTC]*** Break the key value from Step #1 into an evaluator-chosen set of fragments and perform a search using each fragment. (Note that the evaluator shall first confirm with the developer how the key is normally stored, in order to choose fragment sizes that are the same or smaller than any fragmentation of the data that may be implemented by the TOE. The endianness or byte-order should also be taken into account in the search.)

58 Steps #1-8 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

59 ***[**USB iTC to delete or modify this paragraph to make it consistent with the decision on Step 9 above]*** Step #9 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is also found in this repeated run then the test fails and the reason for the collision must be analysed and explained by the developer.

60 Test 2: *Applied to each key or keying material held in non-volatile memory and subject to destruction by overwrite by the TOE.*

61 The evaluator shall:

1. Record the value of the key or keying material.
2. Cause the TOE to perform normal cryptographic processing with the key from Step #1.
3. Search the non-volatile memory in which the key was stored for instances of the known key value from Step #1. (Note that the primary purpose of Step #3 is to demonstrate that appropriate search commands are being used for Step #5 and #6)
4. Cause the TOE to clear the key.
5. Search the non-volatile memory in which the key was stored for instances of the known key value from Step #1. If a copy is found, then the test fails.

6. ***[**As with Step 9 of Test 1: USB iTC to decide whether to add an additional search for key fragments (based on the specific types of risk for the TOE type/deployment environments) and, if so, the relevant length/types of fragment to search for - e.g. fragment sizes might be based on implementation-level key storage structures. Example text follows - to be deleted or modified by the iTC]*** Break the key value from Step #1 into an evaluator-chosen set of fragments and perform a search using each fragment. (Note that the evaluator shall first confirm with the developer how the key is normally stored, in order to choose fragment sizes that are the same or smaller than any fragmentation of the data that may be implemented by the TOE. The endianness or byte-order should also be taken into account in the search.)

62 ***[**USB iTC to delete or modify this paragraph to make it consistent with the decision on Step 6 above]*** Step #6 ensures that partial key fragments do not remain in non-volatile memory. If a fragment is found, there is a chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is also found in this repeated run then the test fails and the reason for the collision must be analysed and explained by the developer.

63 Test 3: *Applied to each key or keying material held in non-volatile memory and subject to destruction by overwrite by the TOE.*

1. Record the storage location (logical address) of the key or keying material.
2. Cause the TOE to perform normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key. Record the value to be used for the overwrite of the key.
4. Examine the storage location from Step #1 to ensure the appropriate pattern (recorded in Step #3) is utilised.

64 The test succeeds if correct pattern is found in the memory location. If the pattern is not found then the test fails.

2.1.5 Cryptographic Key Derivation (FCS_CKM_EXT.5)

2.1.5.1 FCS_CKM_EXT.5/KEK Cryptographic key derivation (Cryptographic Authorisation Data Conditioning)

65 In order to use a NIST SP 800-108 conformant method of key derivation, the TOE must also implement algorithms to generate the key derivation key and KDF. The permitted methods are as follows:

- Generation of key derivation key: NIST SP 800-56A key agreement scheme or NIST SP 800-90A DRBG

- Underlying algorithm of KDF: HMAC or CMAC

2.1.5.1.1 TSS

66 The evaluator shall check that the TSS includes a description of the key derivation function(s) and shall check that this uses a key derivation algorithm and key size(s) according to the specification selected in the ST out of the table as provided in the cPP table per row.

2.1.5.1.2 Guidance Documentation

67 If a selection of key derivation functions (KDF) or parameters are supported, the evaluator shall examine the guidance documentation to determine that the method of choosing a specific mode/derivation function/parameter by the end user is described.

2.1.5.1.3 KMD

68 The evaluator shall examine the KMD to ensure that:

69 The KMD describes the complete key derivation chain and the description must be consistent with the description in the TSS. For all key derivations the TOE must use a method as described in the cPP table. No uncertainty should be left over about how a key is derived from another in the chain.

70 The length of the key derivation key is defined by the PRF. The evaluator should check whether the key derivation key length is consistent with the length provided by the selected PRF.

71 If a key is used as an input to several KDFs, each invocation must use a distinct context string. If the output of a KDF execution is used for multiple cryptographic keys, those keys must be disjoint segments of the output.

72 If the TOE implements Password-Based Key Derivation (KeyDrv4) then the KMD shall describe how the TOE obtains a salt from the RBG to use in the PBKDF.

2.1.5.1.4 Tests

73 The evaluator shall perform the following tests or witness respective tests executed by the developer if technically possible, otherwise an analysis of the implementation representation has to be performed.

74 Preconditions for testing:

- Specification of input parameter to the key derivation function to be tested
- Specification of further required input parameters
- Access to derived key(s)

75 The below tests are derived from *Key Derivation using Pseudorandom Functions (SP 800-108) Validation System (KBKDFVS)*, Updated 4 January 2016, Section 6.2, from the National Institute of Standards and Technology.

76 The evaluator shall perform one or more of the following tests to verify the correctness of the key derivation function, depending on the mode(s) that are supported:

77 ***KeyDrv1: Counter Mode Tests:***

78 The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions (PRFs) that are included in the 'key derivation algorithm' selection in the SFR, and their output lengths in bits (h)
- One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r), and the location of the counter relative to the fixed input data: before, after, or in the middle. If the counter is in the middle then the lengths of data before and after the counter must be determined
- The 'key size' selections in the SFR, i.e. the lengths (in bits) of the derived keying material (L)

79 For each supported combination of PRF, counter location, value of r, and value of L, the evaluator shall generate 20 pseudorandom key derivation key values (K_I).

80 For each value of K_I, the evaluator shall supply this data to the TOE in order to produce the keying material output K_O. The evaluator shall verify that the resulting output matches the results from submitting the same inputs to a known-good implementation of the key derivation function, having the same characteristics.

81 ***KeyDrv2: Feedback Mode Tests:***

82 The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions (PRFs) that are included in the 'key derivation algorithm' selection in the SFR, and their output lengths in bits (h)
- If the implementation includes a counter then one or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r), and the location of the counter relative to the fixed input data: before, after, or in the middle. If the counter is in the middle then the lengths of data before and after the counter must be determined
- The 'key size' selections in the SFR, i.e. the lengths (in bits) of the derived keying material (L)
- The supported IV lengths

83 For each supported combination of PRF, counter location (if a counter is used), value of r (if a counter is used), value of L, and IV length, the evaluator shall generate 20 pseudorandom key derivation key values (K_I).

84 For each value of K_I, the evaluator shall supply this data to the TOE in order to produce the keying material output K_O. The evaluator shall verify that the resulting output matches the results from submitting the same inputs to a known-good implementation of the key derivation function, having the same characteristics.

85 ***KeyDrv3: Double Pipeline Iteration Mode Tests:***

86 The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions (PRFs) that are included in the 'key derivation algorithm' selection in the SFR, and their output lengths in bits (h)
- If the implementation includes a counter then one or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r), and the location of the counter relative to the fixed input data: before, after, or in the middle. If the counter is in the middle then the lengths of data before and after the counter must be determined
- The 'key size' selections in the SFR, i.e. the lengths (in bits) of the derived keying material (L)

87 For each supported combination of PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 20 pseudorandom key derivation key values (K_I).

88 For each value of K_I, the evaluator shall supply this data to the TOE in order to produce the keying material output K_O. The evaluator shall verify that the resulting output matches the results from submitting the same inputs to a known-good implementation of the key derivation function, having the same characteristics.

89 ***KeyDrv4: Password-based Key Derivation***

90 For each combination of algorithm and output key size the evaluator shall supply 10 passphrases as input and obtain the 10 outputs from the PBKDF performed by the TOE, along with the salt(s) used by the TOE. These 10 passphrases shall be different and shall be conformant to the passphrase conditions defined in FIA_SOS.1 and FIA_PPS_EXT.1. The resulting output shall be compared to the results from an independent implementation of the PBKDF for the same salt and passphrase inputs.

91 ***[**USB iTC: The above tests only within the required passphrase range. The test activities for the associated FIA SFRs should include testing that verifies behaviour both below and above the required range.]***

92 ***KeyDrv5: Intermediate Keys Method***

93 If the selected algorithm is a hash then the testing of the hash primitive is the only required Evaluation Activity. If the selected algorithm is XOR then no separate primitive testing is necessary (the testing is covered by Evaluation Activities for FCS_KYC_EXT.1).

94 ***CMAC-AES Tests***

95 These tests are intended to be equivalent to those described in the NIST document, "The CMAC Validation System (CMACVS)", updated 23 August 2011, found at *http://csrc.nist.gov/groups/STM/cavp/documents/mac/CMACVS.pdf*.

96 It is not recommended that evaluators use values obtained from static sources such as *http://csrc.nist.gov/groups/STM/cavp/documents/mac/cmactestvectors.zip* or use values not generated expressly to exercise the CMAC-AES implementation.

97 The evaluator shall test the generation-encryption and decryption-verification functionality of CMAC-AES for the following input parameters:

- **Keys**: All supported and selected key sizes (e.g., 128, 256 bits).
- **Message Length**: Two values that are divisible by the block size of 16 bytes, two values that are not divisible by the block size, a length of 0 (if supported), and the maximum length supported or 2^16, whichever is smaller.
- **CMAC Length**: The minimum length (1 byte), the middle length (8 bytes), and the maximum length (16 bytes).

98 The testing for CMAC consists of two tests:

99 ***CMAC Generation Test***

100 For each supported key size, message length, and MAC length, the evaluator shall supply eight key-message combinations to obtain the resulting MACs. The evaluator shall compare the resulting MACs with the result of providing the same inputs to a known-good implementation.

101 ***CMAC Verification Process Test***

102 For each supported key size, message length, and MAC length, the evaluator shall supply 20 key-message-MAC combinations and determine whether the MAC passes the verification process. The evaluator shall compare the results with the results of providing the same inputs to a known-good implementation.

2.1.6 Cryptographic Operation (FCS_COP.1)

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

2.1.6.1.1 TSS

103 The evaluator shall check that the TSS includes a description of encryption function(s) used for user data encryption. The evaluator should check that this description of the selected encryption function includes the key sizes and modes of operations as specified in the table above per row.

104 The evaluator shall check that the TSS describes the means by which the TOE satisfies constraints on algorithm parameters included in the selections made for ‘cryptographic algorithm’ and ‘list of standard’.

105

2.1.6.1.2 Guidance Documentation

106 If multiple encryption modes are supported, the evaluator examines the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

2.1.6.1.3 KMD

107 The evaluator shall examine the KMD to ensure that the points at which user data encryption and decryption occurs are described, and that the complete data path for user data encryption is described. The evaluator checks that this description is consistent with the relevant parts of the TSS.

108 Assessment of the complete data path for user data encryption includes confirming that the KMD describes the data flow from the device’s host interface to the device’s non-volatile memory storing the data, and gives information enabling the user data datapath to be distinguished from those situations in which data bypasses the data encryption engine (e.g. read-write operations to an unencrypted Master Boot Record area). 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).

109 If XTS-ATE is used as the user data encryption algorithm then the evaluator shall check that the full length keys are created by methods that ensure that the two halves are different and independent.

2.1.6.1.4 Test

110 ***[**Negative tests and constraint tests to be added]***

111 The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer if technically possible, otherwise an analysis of the implementation representation has to be performed.

112 Preconditions for testing:

- Specification of keys as input parameter to the function to be tested
- Specification of required input parameters such as modes
- Specification of user data (plaintext)
- Tapping of encrypted user data (ciphertext) directly in the non-volatile memory

113 ***UDE1: AES-CBC Tests***

114 For the AES-CBC tests described below, the plaintext, ciphertext, and IV values shall consist of 128-bit blocks. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known-good implementation.

115 These tests are intended to be equivalent to those described in NIST's AES Algorithm Validation Suite (AESAVS) (http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf). It is not recommended that evaluators use values obtained from static sources such as the example NIST's AES Known Answer Test Values from the AESAVS document, or use values not generated expressly to exercise the AES-CBC implementation.

116 ***AES-CBC Known Answer Tests***

117 KAT-1 (GFSBox): To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different plaintext values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros.

118 To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different ciphertext values for each selected key size and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using a key value of all zeros and an IV of all zeros.

119 KAT-2 (KeySBox): To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros.

120 To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the plaintext

that results from AES-CBC decryption of an all-zeros ciphertext using the given key and an IV of all zeros.

121 KAT-3 (Variable Key): To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of keys for each selected key size (as described below) and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using each key and an IV of all zeros.

122 Key *i* in each set shall have the leftmost *i* bits set to ones and the remaining bits to zeros, for values of *i* from 1 to the key size. The keys and corresponding ciphertext are listed in AESAVS, Appendix E.

123 To test the decrypt functionality of AES-CBC, the evaluator shall use the same keys as above to decrypt the ciphertext results from above. Each decryption should result in an all-zeros plaintext.

124 KAT-4 (Variable Text): To test the encrypt functionality of AES-CBC, for each selected key size, the evaluator shall supply a set of 128-bit plaintext values (as described below) and obtain the ciphertext values that result from AES-CBC encryption of each plaintext value using a key of each size and IV consisting of all zeros.

125 Plaintext value *i* shall have the leftmost *i* bits set to ones and the remaining bits to zeros, for values of *i* from 1 to 128. The plaintext values are listed in AESAVS, Appendix D.

126 To test the decrypt functionality of AES-CBC, for each selected key size, use the plaintext values from above as ciphertext input, and AES-CBC decrypt each ciphertext value using key of each size consisting of all zeros and an IV of all zeros.

127 ***AES-CBC Multi-Block Message Tests***

128 The evaluator shall test the encrypt functionality by encrypting nine *i*-block messages for each selected key size, for 2 <= *i* <=10. For each test, the evaluator shall supply a key, an IV, and a plaintext message of length *i* blocks, and encrypt the message using AES-CBC. The resulting ciphertext values shall be compared to the results of encrypting the plaintext messages using a known good implementation.

129 The evaluator shall test the decrypt functionality by decrypting nine *i*-block messages for each selected key size, for 2 <= *i* <=10. For each test, the evaluator shall supply a key, an IV, and a ciphertext message of length *i* blocks, and decrypt the message using AES-CBC. The resulting plaintext values shall be compared to the results of decrypting the ciphertext messages using a known good implementation.

130 ***AES-CBC Monte Carlo Tests***

131 The evaluator shall test the encrypt functionality for each selected key size using 100 3-tuples of pseudo-random values for plaintext, IVs, and keys.

132 The evaluator shall supply a single 3-tuple of pseudo-random values for each selected key size. This 3-tuple of plaintext, IV, and key is provided as input to the below algorithm to generate the remaining 99 3-tuples, and to run each 3-tuple through 1000 iterations of AES-CBC encryption.

```
# Input: PT, IV, Key
Key[0] = Key
IV[0] = IV
PT[0] = PT
for i = 0 to 99 {
    Output Key[i], IV[i], PT[0]
    For j = 0 to 999 {
        if (j == 0) {
                CT[j] = AES-CBC-Encrypt(Key[i], IV[i], PT[j])
                PT[j+1] = IV[i]
        } else {
                CT[j] = AES-CBC-Encrypt(Key[i], PT[j])
                PT[j+1] = CT[j-1]
        }
    }
    Output CT[j]
    If (KeySize == 128) Key[i+1] = Key[i] xor CT[j]
    If (KeySize == 192) Key[i+1] = Key[i] xor (last 64 bits of CT[j-1] ||
CT[j])
    If (KeySize == 256) Key[i+1] = Key[i] xor (CT[j-1] || CT[j])
    IV[i+1] = CT[j]
    PT[0] = CT[j-1]
}
```

133 The ciphertext computed in the 1000th iteration (CT[999]) is the result for each of the 100 3-tuples for each selected key size. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

134 The evaluator shall test the decrypt functionality using the same test as above, exchanging CT and PT, and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

135 ***UDE2: AES-CCM Tests***

136 These tests are intended to be equivalent to those described in the NIST document, "The CCM Validation System (CCMVS)", updated 9 Jan 2012, found at http://csrc.nist.gov/groups/STM/cavp/documents/mac/CCMVS.pdf.

137 It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

138 The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

- **Keys**: All supported and selected key sizes (e.g., 128, 256 bits).
- **Associated Data**: Two or three values for associated data length: The minimum (>=0 bytes) and maximum (<=32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.
- **Payload**: Two values for payload length: The minimum (>=0 bytes) and maximum (<=32 bytes) supported payload lengths.
- **Nonces**: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13) in bytes.
- **Tag**: All supported tag lengths (4, 6, 8, 10, 12, 14, 16) in bytes.

139 The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

140 Variable Associated Data Test: For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

141 Variable Payload Test: For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

142 Variable Nonce Test: For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the evaluator

shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

143 Variable Tag Test: For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

144 Decryption-Verification Process Test: To test the decryption-verification functionality of AES- CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

145 ***UDE3: AES-GCM Tests***

146 These tests are intended to be equivalent to those described in the NIST document, "The Galois/Counter Mode (GCM) and GMAC Validation System (GCMVS) with the Addition of XPN Validation Testing", rev. 15 Jun 2016, section 6.2, found at http://csrc.nist.gov/groups/STM/cavp/documents/mac/gcmvs.pdf.

147 It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/gcmtestvectors.zip, or use values not generated expressly to exercise the AES-GCM implementation.

148 The evaluator shall test the authenticated encrypt functionality of AES-GCM by supplying 15 sets of Key, Plaintext, AAD, IV, and Tag data for every combination of the following parameters as selected in the ST and supported by the implementation under test:

- **Key size in bits**: Each selected and supported key sizes (128, 256).
- **Plaintext length in bits**: Up to four values for plaintext length: Two values that are non-zero integer multiples of 128, if supported. And two values that are non-multiples of 128, if supported.
- **AAD length in bits**: Up to five values for AAD length: Zero-length, if supported. Two values that are non-zero integer multiples of 128, if supported. And two values that are integer non-multiples of 128, if supported.
- **IV length in bits**: Up to three values for IV length: 96 bits. Minimum and maximum supported lengths, if different.
- **Tag length in bits**: Each supported length (128, 120, 112, 104, 96, 64, 32).

149 To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

150 The evaluator shall test the authenticated decrypt functionality of AES-GCM by supplying 15 Ciphertext-Tag pairs for every combination of the above parameters, replacing Plaintext length with Ciphertext length. For each parameter combination the evaluator shall introduce an error into either the Ciphertext or the Tag such that approximately half of the cases are correct and half the cases contain errors. To determine correctness, the evaluator shall compare the resulting pass/fail status and Plaintext values to the results obtained by submitting the same inputs to a known-good implementation.

151 ***UDE4: XTS-AES Tests***

152 These tests are intended to be equivalent to those described in the NIST document, "The XTS-AES Validation System (XTSVS)", updated 5 Sept 2013, found at http://csrc.nist.gov/groups/STM/cavp/documents/aes/XTSVS.pdf.

153 It is not recommended that evaluators use values obtained from static sources such as the XTS-AES test vectors at http://csrc.nist.gov/groups/STM/cavp/documents/aes/XTSTestVectors.zip or use values not generated expressly to exercise the XTS-AES implementation.

154 The evaluator shall generate test values as follows:

155 For each supported key size (256 bit (for AES-128) and 512 bit (for AES-256) keys), the evaluator shall provide up to five data lengths:

- Two data lengths divisible by the 128-bit block size, if data unit lengths of complete block sizes are supported.
- Two data lengths not divisible by the 128-bit block size, if data unit lengths of partial block sizes are supported.
- The largest data length supported by the implementation, or 2^16 (65536), whichever is larger.

156 The evaluator shall specify whether the implementation supports tweak values of 128-bit hexadecimal strings or a data unit sequence number, or both.

157 For each combination of key size and data length, the evaluator shall provide 100 sets of input data and obtain the ciphertext that results from XTS-AES encryption. If both kinds of tweak values are supported then each type of tweak value shall be used in half of every 100 sets of input data, for all combinations of key size and data length. The evaluator shall verify that the resulting ciphertext matches the results from submitting the same inputs to a known-good implementation of XTS- AES.

158 The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

159 ***UDE5: Camellia-CBC Tests***

160 To test the encrypt and decrypt functionality of Camellia in CBC mode, the evaluator shall perform the tests as specified in 10.2.1.2 of ISO/IEC 18367:2016.

161 ***UDE6: Camellia-CCM Tests***

162 To test the encrypt functionality of Camellia in CCM mode, the evaluator shall perform the tests as specified in 10.6.1.1 of ISO/IEC 18367:2016.

163 To test the decrypt functionality of Camellia in CCM mode, the evaluator shall perform the tests as specified in 10.6.1.2 of ISO/IEC 18367:2016.

164 As a prerequisite for these tests, the evaluator shall perform the test for encrypt functionality of Camellia in ECB mode as specified in 10.2.1.2 of ISO/IEC 18367:2016.

165 ***UDE7: Camellia-GCM Tests***

166 To test the encrypt functionality of Camellia in GCM, the evaluator shall perform the tests as specified in 10.6.1.1 of ISO/IEC 18367:2016.

167 To test the decrypt functionality of Camellia in GCM, the evaluator shall perform the tests as specified in 10.6.1.2 of ISO/IEC 18367:2016.

168 As a prerequisite for these tests, the evaluator shall perform the test for encrypt functionality of Camellia in ECB mode as specified in 10.2.1.2 of ISO/IEC 18367:2016.

169 ***UDE8: XTS-Camellia Tests***

170 These tests are intended to be equivalent to those described in the IPA document, ATR-01-B, "*Specifications of Cryptographic Algorithm Implementation Testing – Symmetric-Key Cryptography*", found at https://www.ipa.go.jp/security/jcmvp/jcmvp_e/documents/atr/atr01b_en.pdf.

171 The evaluator shall generate test values as follows:

172 For each supported key size (256 bit (for Camellia-128) and 512 bit (for Camellia-256) keys), the evaluator shall provide up to five data lengths:

- Two data lengths divisible by the 128-bit block size, if data unit lengths of complete block sizes are supported.
- Two data lengths not divisible by the 128-bit block size, if data unit lengths of partial block sizes are supported.
- The largest data length supported by the implementation, or 2^16 (65536), whichever is larger.

173 The evaluator shall specify whether the implementation supports tweak values of 128-bit hexadecimal strings or a data unit sequence number, or both.

174 For each combination of key size and data length, the evaluator shall provide 100 sets of input data and obtain the ciphertext that results from XTS-Camellia encryption. If both kinds of tweak values are supported, 50 of each 100 sets of input data shall use each type of tweak value. The resulting ciphertext shall be compared to the results of a known-good implementation.

175 As a prerequisite for this test, the evaluator shall perform the test for encrypt functionality of Camellia in ECB mode as specified in 10.2.1.2 of ISO/IEC 18367:2016.

176 The evaluator shall test the decrypt functionality of XTS-Camellia using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-Camellia encrypt with XTS-Camellia decrypt.

177 As a prerequisite for this test, the evaluator shall perform the test for decrypt functionality of Camellia in ECB mode as specified in 10.2.1.2 of ISO/IEC 18367:2016.

2.1.7 Cryptographic Key Chaining (FCS_KYC_EXT.1)

2.1.7.1 FCS_KYC_EXT.1 Cryptographic key chaining

2.1.7.1.1 TSS

178 The evaluator shall check that the TSS contains a high-level description of the chain of intermediary keys (including the type and length of each key) originating from the authorisation data and ending with the DEK.

2.1.7.1.2 Guidance Documentation

179 None.

2.1.7.1.3 KMD

180 The evaluator shall examine the KMD to verify that it describes the chain of intermediary keys originating from the authorisation data and ending in the DEK using methods selected in FCS_KYC_EXT. The evaluator shall ensure that the description of the key chain demonstrates that it maintains the chain of keys using an authorisation data submask according to FCS_CKM_EXT.5, key wrapping according to FCS_CKM.3 and uses only other selected methods in FCS_KYC_EXT.1 in accordance with the definition of their associated SFRs.

181 The evaluator shall examine the KMD to verify that the effective strength of the DEK (based only on key length) is maintained throughout the key chain. The evaluator shall examine the key hierarchy to ensure that at no point could the chain be broken without a cryptographic exhaust or knowledge of the initial authorisation value.

182 The evaluator shall verify the KMD includes a description of the effective strength of keys throughout the key chain.

183 The evaluator shall examine the KMD to verify that the description of the key chain is consistent with the information given in the TSS (e.g. by examining the description of the key chain in both places), the Operational Guidance (e.g. by examining the description of user inputs required, any configuration options available, and the operations available to directly or indirectly create and use keys[4]), and any observations made during evaluator testing.

2.1.7.1.4 Tests

184 TBD.

2.1.8 Cryptographic Salt Generation (FCS_SLT_EXT.1)

2.1.8.1 FCS_SLT_EXT.1 Cryptographic salt generation

2.1.8.1.1 TSS

185 The evaluator shall ensure the TSS describes how salts are generated using the RBG.

2.1.8.1.2 Guidance Documentation

186 None.

2.1.8.1.3 Tests

187 The evaluator shall confirm by testing that the salts obtained in the cryptographic operations that use the salts are of the length specified in FCS_SLT_EXT.1, are obtained from the RBG, and are fresh on each invocation.

188 Note: in general these tests may be carried out as part of the tests of the relevant cryptographic operations.

2.1.9 Random Bit Generation (FCS_RBG_EXT)

2.1.9.1 FCS_RBG_EXT.1 Random Bit Generation (RBG)

2.1.9.1.1 TSS

189 TBD.

2.1.9.1.2 Guidance Documentation

190 TBD.

191

[4] For example: the relationship of authorisation data validation to the decryption of the DEK should be examined for consistency with the key chain description to check for any possible intermediate validation operations and/or data that are not mentioned in the key chain description.

2.1.9.1.3 Tests

192 The following test is intended to be equivalent to that defined in *The NIST SP 800-90A Deterministic Random Bit Generator Validation System (DRBGVS)*, Updated 29 October 2015, from the National Institute of Standards and Technology (http://csrc.nist.gov/groups/STM/cavp/documents/drbg/DRBGVS.pdf). It is not recommended that evaluators use values obtained from static sources such as the sample DRBG Test Vectors on the CAVP Test site.

193 The evaluator shall verify the implementation of the Deterministic Random Bit Generation function by running 15 tests for each combination of the following parameters as selected in FCS_RBG_EXT.1.1 and supported by the implementation:

- Mechanism: Hash_DRBG, HMAC_DRBG, CTR_DRBG
- Option:
  - for Hash_DRBG and HMAC_DRBG: selected hash function and size
  - for CTR_DRBG: selected block cipher and whether or not a Derivation Function (df) is used
- Prediction Resistance enabled or disabled
- Entropy input length
- Nonce length
- Personalization String length
- Additional Input length
- Returned Bits length

194 *Tests with Prediction Resistance Enabled consist of the following steps:*

1. Instantiate DRBG
2. Generate a first block of random bits
3. Generate a second block of random bits
4. Uninstantiate DRBG

195 For each test, the evaluator shall provide the following randomly generated inputs:

- Entropy, Nonce, and Personalization string for step (1)
- Additional Input and Entropy for step (2)
- Additional Input and Entropy for step (3)

196 The evaluator shall use a known-good implementation to verify that the Returned Bits output from step (3) is the result expected.

197 *Tests with Prediction Resistance Disabled consist of the following steps:*

1. Instantiate DRBG
2. Reseed (if the implementation supports reseed functionality)
3. Generate a first block of random bits
4. Generate a second block of random bits
5. Uninstantiate DRBG

198 For each test, the evaluator shall provide the following randomly generated inputs:

- Entropy, Nonce, and Personalization String for step (1)
- Additional Input and Entropy for step (2) (if reseed is supported)
- Additional Input for step (3)
- Additional Input for step (4)

199 The evaluator shall use a known-good implementation to verify that the Returned Bits output from step (4) is the result expected.

200 The implementation passes the DRBG test if the Returned Bits result matches the Returned Bits from the known-good implementation.

2.2 User Data Protection (FDP)

2.2.1 Protection of User Data on Device (FDP_UDD_EXT)

2.2.1.1 FDP_UDD_EXT.1 Protection of User Data on Device

2.2.1.1.1 TSS

201 The evaluator shall examine the TSS to ensure that it describes how user data is written to the device's storage medium and the point at which the encryption function is applied. The evaluator examines the TSS to confirm its justification

of why standard methods of accessing the device via the host platform's operating system will always pass through these functions.

202 The evaluator shall verify that the TSS describes the initialization of the TOE and the activities the TOE performs to ensure that it encrypts the entirety of the user data when a user first provisions the TOE. The evaluator shall verify that the TSS describes areas of the storage medium that it does not encrypt, and confirms that no user data is stored in those areas.

2.2.1.1.2 KMDSD

203 The evaluator shall examine the KMDSD to verify that it includes all of the requirements for this document in [USBcPP, D].

204 The evaluator shall examine the KMDSD to verify that it provides sufficient description of all platforms to ensure that the product encrypts all user data storage areas. In performing this examination the evaluator shall take into account (at least) the description of the relevant datapaths, the situations identified in the KMDSD in which user data may be read and stored in other parts of the TOE (e.g. as part of a caching or look-ahead strategy), and the KMDSD rationale for why no stored unencrypted user data can survive beyond the session in which it is written and/or read.

205 The evaluator shall examine the KMDSD to verify that it provides information on those conditions in which data bypasses the data encryption engine (e.g. for system data) and shall confirm that this does not include user data.

206 The evaluator shall examine the KMDSD to verify that it provides a description of the platform's boot initialisation, the encryption initialisation process, and at what point the product enables the encryption. The evaluator shall confirm that the description shows that the product does not allow for the transfer of user data before it fully initialises the encryption.

207 The evaluator shall examine the KMDSD to ensure the consistency and accuracy of the description as judged against the TSS, the operational guidance, and any observations made during testing.

2.2.1.1.3 Operational Guidance

208 The evaluator shall examine the AGD guidance to determine that it describes the initial steps needed to enable all necessary cryptographic functions. The guidance shall provide instructions that are sufficient to ensure that all user data stored on the device will be encrypted. The evaluator shall examine the AGD guidance to determine that user data encryption is performed without user intervention. The user data encryption shall occur transparently to the user and the decision to protect the data is outside the discretion of the user.

2.2.1.1.4 Test

209 The evaluator examines the tool and its documentation to confirm that it cannot be used to compromise instances of the TOE in a real operational environment (i.e. that they can be used only in test/diagnostic environments).

210 The evaluator shall perform the following tests:

211 Test 1: The evaluator shall utilize developer provided tools which allow inspection of the encrypted drive, and may allow provisioning with a known key. The evaluator shall ensure that the TOE is initialized and that the encryption engine is ready. The evaluator shall:

1. Determine a random character pattern of at least 64 KB;
10. Retrieve information on the TOE's lowest and highest logical address for which encryption is enabled;
11. Write pattern to storage device in multiple locations: randomly select several logical address locations within the device's lowest to highest address range and write pattern to those addresses.
12. Verify data is encrypted: engage device's functionality for generating a new encryption key, thus performing an erase of the key per FCS_CKM.4. Read from the same locations at which the data was written; compare the retrieved data to the written data and ensure they do not match.

2.2.2 Protection of System Data on Device (FDP_SDD_EXT)

2.2.2.1 FDP_SDD_EXT.1 Protection of System Data on Device

2.2.2.1.1 TSS

212 The evaluator shall examine the TSS to ensure that it identifies the users authorised to write to system data, and describes how system data is written to the device's storage medium, including the nature of the authorisation mechanism and the point at which it is applied. The evaluator examines the TSS to confirm its justification of why standard methods of accessing the device via the host platform's operating system will always pass through these functions.

213 The evaluator shall examine the TSS to ensure the accuracy of the description as judged against other parts of the ST, the KMDSD, the operational guidance, and any observations made during testing.

214 The evaluator shall verify that the TSS describes the initialisation of the TOE and the activities the TOE performs to ensure that it protects the system data from unauthorised access when a user first provisions the TOE.

2.2.2.1.2 KMDSD

215 The evaluator shall examine the KMDSD to verify that it provides sufficient description of all platforms to enable the evaluator to ensure that the product protects against unauthorised access to all system data storage areas.

216 The evaluator shall examine the KMDSD to verify that it provides a description of the platform's boot initialisation, and at what point the product enables the system data protection. The evaluator shall confirm that the description shows that the product does not allow modification of system data before it fully initialises the access protection.

2.2.2.1.3 Operational Guidance

217 The evaluator shall check the AGD guidance to determine that system data can only change in ways that reflect legitimate use of the device by authorised users. The evaluator shall verify that descriptions provided in the AGD guidance corresponds to descriptions in the TSS and the KMDSD.

2.2.2.1.4 Test

218 The evaluator shall perform the following tests:

219 Test 1: The evaluator shall initialise the TOE and before the device fully initialises the access protection, the evaluator shall attempt to modify system data via the host platform's operating system.

220 Test 2: The evaluator shall not provide any authorisation data and attempt to modify system data via the host platform's operating system.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication Failures (FIA_AFL)

2.3.1.1 FIA_AFL.1 Authentication failure handling

2.3.1.1.1 TSS

221 The evaluator shall check that the TSS identifies the maximum number of unsuccessful authentication attempts prior to the deletion of the DEK by the TSF. The evaluator shall also examine the TSS to determine whether the user is able to configure the limit of unsuccessful authentication attempts and, if so, shall verify that the TSS specifies a range of acceptable values that is consistent with FIA_AFL.1.

2.3.1.1.2 KMDSD

222 The evaluator shall examine the KMDSD to verify that it describes the methods the TOE employs to limit the number of consecutively failed authorisation attempts.

2.3.1.1.3 Operational Guidance

223 The evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established. The operational guidance shall also list a range of acceptable values. If this value is not configurable, the limit shall simply be stated in the guidance.

224 The evaluator shall examine the operational guidance to ensure that it clearly alerts the user to the fact that the DEK is deleted and that therefore the encrypted user data will be permanently inaccessible after the defined number of unsuccessful authorisation attempts has been met.

2.3.1.1.4 Test

225 The evaluator shall perform the following test

226 Test 1: The evaluator shall confirm that the TSF will not allow to configure a number of unsuccessful authorisation attempts that is outside of the specified range of acceptable values. This test case is only applicable for devices that allow configuration of the authentication failure threshold value.

227 Test 2: The evaluator shall enter invalid authorisation data so that the documented maximum number of unsuccessful authorisation attempts is reached. The evaluator shall verify that the encrypted user data is no longer available on the device.

2.3.2 Passphrase support (FIA_PPS)

2.3.2.1 FIA_PPS_EXT.1 Passphrase entry interface

2.3.2.1.1 TSS

228 The evaluator shall check that the TSS describes the method of passphrase entry on the device.

2.3.2.1.2 Operational Guidance

229 The evaluator shall examine the operational guidance to ensure that the method of passphrase entry on the device is described. The operational guidance shall specify if the passphrase is entered via the host software or if the TOE includes a passphrase-entry interface. The guidance documentation shall describe all passphrase entry methods in case the device support more than one passphrase entry method/interface.

2.4 Protection of the TSF (FPT)

2.4.1 Fail secure (FPT_FLS)

2.4.1.1 FPT_FLS.1 Failure with preservation of secure state

2.4.1.1.1 TSS

230 The evaluator shall check that the TSS describes the failure conditions that cause the TOE to enter a mute state, and that the mute state is specified as being irreversible.

2.4.1.1.2 KMDSD

231 The evaluator shall examine the KMDSD to verify it specifies how the TOE ensures that all data output via the data output interface is to be inhibited during error states or self-test conditions. The evaluator shall also verify, by inspection of the design of the TOE, that the data output interface is, in fact, logically or physically inhibited under these conditions.

2.4.1.1.3 Operational Guidance

232 The evaluator shall verify that the operational guidance describes the method by which the product verifies the correct operation of the TSF. The evaluator shall verify that the operational guidance describes security-relevant events related to the self-testing failures, such that each user knows what events may occur and what action (if any) he may have to take in order to maintain security.

233 The evaluator shall verify that the operational guidance specifies that all data output via the data output interface is inhibited whenever the TOE is in an error state. The evaluator shall verify from the operational guidance that once an error condition is detected and the error state is entered, all data output via the data output interface is inhibited and the device enters an irreversible mute state. Status information to identify the type of error may be allowed from the status output interface, as long as the evaluator can verify that no CSPs, plaintext data, or other information that if misused could lead to a compromise.

2.4.1.1.4 Test

234 The evaluator shall perform the following tests:

235 Test 1: The evaluator shall cause self-testing errors and firmware integrity test errors during initial start-up to verify that the device preserves a secure state i.e. enters a mute state. This test should be repeated for all different failure conditions. The evaluator shall:

1. cause known answer self-testing and firmware integrity tests errors. *<Check: is this feasible?>*
2. verify that all data output via the data output interface is inhibited and the device enters the mute state. If status information is output

from the status output interface to identify the type of error, the evaluator shall verify that the information output is not sensitive. The evaluator shall verify that no plaintext data, or other information that if misused could lead to a compromise.

2.4.2 Protection of Keys and Keying Material (FPT_KYP_EXT)

2.4.2.1 FPT_KYP_EXT.1 Protection of Keys and Keying Material

2.4.2.1.1 TSS

236 The evaluator shall check the TSS to confirm that protection of keys and keying material is described in the TSS.

2.4.2.1.2 KMDSD

237 The evaluator shall examine the KMDSD to ensure that the methods used to protect the keys stored in non-volatile memory are described, and that this is consistent with the description in the TSS, the Operational Guidance, and any observations made during evaluator testing.

238 The evaluator shall examine the KMDSD to ensure that it describes the storage location of all keys and the protection of all keys stored in non-volatile memory, verifying that they are wrapped as specified in FCS_CKM.3 or encrypted as specified in FCS_COP.1/KeyEnc.

239 The evaluator is reminded that plaintext keys or keying material that are not part of the key chain for the purposes of FCS_KYC_EXT.1, and plaintext keys or keying material that no longer provide access to the encrypted user data after initial provisioning, do not need to be stored encrypted or wrapped in non-volatile memory.

2.4.2.1.3 Test

240 The evaluator shall perform the following test:

241 Test 1: The evaluator shall utilize developer provided tools which allow inspection of the encrypted drive, and may allow provisioning with a known key. The evaluator shall ensure that the TOE is initialized and that the encryption engine is ready. The evaluator shall ensure that keys and keying material are stored wrapped or encrypted, i.e. keys that are part of the key chain are not stored in plaintext.

2.4.3 TSF self test (FPT_TST)

2.4.3.1 FPT_TST.1 TSF testing

2.4.3.1.1 TSS

242 The evaluator shall examine the TSS to confirm that it describes the known-answer tests for cryptographic functions and firmware integrity tests.

243 The evaluator shall examine the TSS to confirm that it describes the method by which the product verifies the correct operation of the TSF and the integrity of TSF data and firmware. The evaluator shall verify that the TSS indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user.

244 The evaluator shall check that the TSS includes a description of the irreversible mute state that the TSF enters when self-tests fail (cf. FPT_FLS.1).

2.4.3.1.2 KMDSD

245 The evaluator shall examine the KMDSD description of the initialisation process to ensure that it identifies the point at which the self-tests are run.

2.4.3.1.3 Operational Guidance

246 The evaluator shall examine the operational guidance to ensure that the self-tests performed during initial start-up of the device are described.

247 The user guidance shall include a description of the irreversible mute state that the TSF enters when self-tests fail. The user guidance shall also state that the mute state is irreversible. The evaluator shall verify that there no conditions and actions described in the user guidance to exit the mute state and resume normal operation.

2.4.4 Submask Validation (FPT_VAL_EXT)

2.4.4.1 FPT_VAL_EXT.1 Validation

2.4.4.1.1 TSS

248 The evaluator shall examine the TSS to check that the TSF supports a validation mechanism for each authorisation data submask used in the key chain.

249 The evaluator shall examine the TSS to verify that the link between individual submask validation actions and the definition of an authorisation attempt failure for FIA_AFL.1 is described.

2.4.4.1.2 KMDSD

250 The evaluator shall examine the KMDSD to ensure that it describes how validation is performed, to identify the validation mechanism for each authorisation data submask involved in the key chain and to verify that each validation is performed using a method that is specified in FPT_VAL_EXT.1.

251 The evaluator shall examine the KMDSD to verify that the validation process does not expose any material that might compromise the authorisation data submask(s).

2.4.4.1.3 Test

252 The evaluator shall perform the following test:

253 Test 1: The evaluator shall provide an incorrect authorisation factor and ensure that the authorisation submask validation has failed. The evaluator shall verify that the TOE behaves as described in the TSS. The evaluator shall ensure to test all validation mechanisms described in the KMDSD and repeat this test for different validation methods.

254 Test 2: The evaluator shall provide a correct authorisation factor and ensure that the authorisation submask validation has been successful. The evaluator shall verify that the TOE behaves as described in the TSS. The evaluator shall ensure to test all validation mechanisms described in the KMDSD and repeat this test for different validation methods.

2.5 TOE Access (FTA)

2.5.1 TOE access authorisation (FTA_USB)

2.5.1.1 FTA_USB_EXT.1 User Authorisation

2.5.1.1.1 TSS

255 The evaluator shall check that the TSS contains a description of user authorisation, re-authorisation, and session termination.

2.5.1.1.2 Operational Guidance

256 The evaluator shall review the operational guidance to verify that it contains instructions for starting a session with a valid passphrase, termination of a session by the host, and re-authorisation being required 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
- any other conditions identified in the assignment in FTA_USB_EXT.1.2.

257 The evaluator shall also review the operational guidance to verify it contains the description of an inactivity time limit, which terminates the session by putting the TOE into a powered-down or sleep state if exceeded.

2.5.1.1.3 Test

258 The evaluator shall perform the following tests:

259 Test 1: The evaluator shall connect the TOE to a host device and verify that correct authorisation is required before access to the related user data.

260 Test 3: The evaluator shall verify any previous sessions have expired when the host device has powered-down or gone into a sleep state while the TOE was still connected. The evaluator shall verify re-authorisation is required in order to access user data when the host device powers-up or awakes from sleep.

261 Test 4: The evaluator shall determine the inactivity time limit from the operational guidance and verify the TOE powers down or enters a sleep state when the inactivity time limit is reached. The evaluator shall verify any previous sessions have expired and user data is inaccessible when the TOE itself has powered-down or gone into a sleep state. The evaluator shall verify re-authorisation is required when the TOE powers-up or awakes from sleep.

262 Test 5: The evaluator shall initiate session termination from the host device using instructions provided in the operational guidance. The evaluator shall then verify user data is inaccessible once the session has been terminated via the host.

2.6 Security Management (FMT)

2.6.1 Specification of Management Functions (FMT_SMF)

2.6.1.1 FMT_SMF.1 Specification of Management Functions

2.6.1.1.1 TSS

263 The evaluator shall examine the TSS to confirm that the management functions included in FMT_SMF.1 are described.

2.6.1.1.2 Operational Guidance

264 The evaluation shall review the operational guidance to ensure that it contains instructions on how to change the value of the authorisation data.

2.6.1.1.3 Test

265 The evaluator shall perform the following tests:

266 Test 1: The evaluator shall change the value of the authorisation data following the instructions provided in the operational guidance. The evaluator shall verify that the TOE denies access to user's encrypted data when the evaluator uses the old authorisation factor values to gain access.

3 Evaluation Activities for Optional Requirements

3.1 Protection of the TSF (FPT)

3.1.1 Trusted Update (FPT_TUD_EXT)

[**USB iTC: Suggestion for FPT_TUD_EXT EA as this does not belong in FCS SigVer primitive testing]

3.1.1.1 FPT_TUD_EXT.1 Trusted Update

3.1.1.1.1 TSS

267 The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system software. The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. The evaluator shall verify that the TSS describes the method by which the digital signature is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature of the update, and the actions that take place for both successful and unsuccessful signature verification.

3.1.1.1.2 Guidance Documentation

268 The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

3.1.1.1.3 KMD

269 The evaluator shall examine the KMD to ensure the following aspects:

- KMDSD must describe how the integrity of digital signature verification keys in the TOE is protected. In the case of ECDSA, the EC domain parameters have to be integrity protected as well.
- KMDSD must describe how the private key was created and how it is integrity and confidentiality protected within the development site. The developer must state in the KMDSD that the private key is only used to sign the TOE firmware.
- KMDSD must describe which parts of the TOE can be updated. E.g. firmware incl. bootloader, firmware without bootloader, single files, etc.

3.1.1.1.4 Tests

270 Test 1: The evaluator performs the version verification activity to determine the current version of the product as well as the most recently installed version (should be the same version before updating). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

271 Test 2: The evaluator performs the version verification activity to determine the current version of the product as well as the most recently installed version (should be the same version before updating). The evaluator obtains or produces illegitimate update as described below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

- A modified version (e.g. using a hex editor) of a legitimately signed update. The modification must cover all parts of the update. If e.g. the update has the following format [Header | Firmware | Signature], then all of the three parts have to be modified independently. One modification must be an empty signature.

- The handling of version information of the most recently installed version might differ between different TOEs. Depending on the point in time when the attempted update is rejected, the most recently installed version might or might not be updated. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

- If there are several user roles defined for the TOE the evaluator has to examine the user guidance to identify roles authorized to initiate an update process. He has to test that the update process fails for unauthorized users according the guidance.

272 Test 3: The evaluator shall test if the TOE remains in a secure state after interrupting the update process e.g. by a power outage.

3.1.2 Trusted Update Rollback (FPT_TUR_EXT)

3.1.2.1 FPT_TUR_EXT.1 Trusted Update Rollback

3.1.2.1.1 TSS

273 The evaluator shall check the TSS to ensure that it describes any constraints on the ability to reverse previous successful updates, or to apply earlier updates after later updates have already been successfully applied.

3.1.2.1.2 Operational Guidance

274 The evaluator shall examine the operational guidance to confirm that it describes how authorised users can perform rollback of previously applied updates. The evaluator also ensures that the operational guidance describes how the product obtains candidate rollback updates; the processing associated with verifying the digital signature, published hash or keyed hash of the rollback updates; and the actions that take place for successful and unsuccessful cases.

3.1.2.1.3 Test

275 The evaluator shall perform the following test:

276 Test 1: The evaluator performs the version verification activity to determine the current firmware version of the product. The evaluator obtains a legitimate previous firmware update using procedures described in the operational guidance and verifies that it an update successfully installs it on the product. The evaluator verifies that the version correctly corresponds to that of the update. The evaluator shall perform a subset of other assurance activity tests to demonstrate that the update functions as expected.

4 Evaluation Activities for Selection-Based Requirements

4.1 Cryptographic Support (FCS)

4.1.1 Cryptographic Key Generation (FCS_CKM.1)

4.1.1.1 FCS_CKM.1/Asymm Cryptographic key generation (Asymmetric)

277 For any Identifier (AKG1-AKG3), this applies.

4.1.1.1.1 TSS

278 The evaluator shall examine the TSS to verify that it describes how the TOE obtains a key based on input from a random bit generator as specified in FCS_RBG_EXT.1. The evaluator shall review the TSS to verify that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator shall examine the TSS to verify that it identifies the usage for each row identifier (key name, key size, standards) selected in the ST.

4.1.1.1.2 Guidance Documentation

279 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key name(s) for all uses identified in the ST.

4.1.1.1.3 Key Management Description (KMD)

280 If the TOE uses the generated key in a key chain/hierarchy then the evaluator shall confirm that the KMD describes:

- If AKG1 is selected, then the KMD describes which methods for generating p and q are used
- How the key is used as part of the key chain/hierarchy.

4.1.1.1.4 Tests

281 The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are not found on the TOE in its evaluated configuration.

282 ***AKG1: RSA Key Generation***

283 The below tests are derived from *The 186-4 RSA Validation System (RSA2VS)*, Updated 8 July 2014, Section 6.2, from the National Institute of Standards and Technology.

284 The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent *e*, the private prime factors *p* and *q*, the public modulus *n* and the calculation of the private signature exponent *d*.

285 FIPS 186-4 Key Pair generation specifies 5 methods for generating the primes *p* and *q*.

286 These are:

1. Random Primes:

   - Provable primes
   - Probable primes

2. Primes with Conditions:

   - Primes *p1*, *p2*, *q1*, *q2*, *p* and *q* shall all be provable primes
   - Primes *p1*, *p2*, *q1* and *q2* shall be provable primes and *p* and *q* shall be probable primes
   - Primes *p1*, *p2*, *q1*, *q2*, *p* and *q* shall all be probable primes

287 To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair.

288 For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

289 If the TOE generates Random Probable Primes then if possible, the Random Probable primes method should also be verified against a known good implementation as described above. If verification against a known good implementation is not possible, the evaluator shall have the TSF generate 25 key pairs for each supported key length *nlen* and verify that all of the following are true:

- n= p*q
- p and q are probably prime according to Miller-Rabin tests with error probability < 2^(-125)
- 2^16 < e < 2^256 and e is an odd integer
- GCD(p-1,e) = 1
- GCD(q-1, e) = 1
- |p-q| > 2^(nlen/2 – 100)
- p >= squareroot(2)*(2^(nlen/2-1))
- q >= squareroot(2)*(2^(nlen/2-1))
- 2^(nlen/2) < d < LCM(p-1, q-1)
- e*d = 1 mod LCM(p-1, q-1)

290 ***AKG2 & AKG3: ECC Key Generation***

291 These tests are derived from *The 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS)*, Updated 18 Mar 2014, Section 6.

292 *ECC Key Generation Test*

293 For each selected curve, and for each key pair generation method as described in FIPS 186-4, section B.4, the evaluator shall require the implementation under test to generate 10 private/public key pairs (d, Q). The private key, d, shall be generated using a random bit generator as specified in FCS_RBG_EXT.1. The private key, d, is used to compute the public key, Q'. The evaluator shall confirm that 0<d<n (where n is the order of the group), and the computed value Q' is then compared to the generated public/private key pairs' public key, Q, to confirm that Q is equal to Q'.

294 *Public Key Validation (PKV) Test*

295 For each supported curve, the evaluator shall generate 12 private/public key pairs using the key generation function of a known good implementation and modify six of the public key values so that they are incorrect, leaving six values unchanged (i.e., correct). To determine correctness, the evaluator shall submit the 12 key pairs to the public key validation (PKV) function of the TOE and shall confirm that the results correspond as expected to the modified and unmodified values.

4.1.2 Cryptographic Key Access (FCS_CKM.3)

4.1.2.1 FCS_CKM.3/Chain Cryptographic key access (Key Wrapping)

296 *Same as for FCS_CKM.3/DEK?*

4.1.3 Cryptographic Key Derivation (FCS_CKM_EXT.5)

4.1.3.1 FCS_CKM_EXT.5/Chain Cryptographic key derivation

297 *Same as for FCS_CKM_EXT.5/KEK?*

4.1.4 Cryptographic operation (FCS_COP.1)

4.1.4.1 FCS_COP.1/KeyEnc Cryptographic operation (Key Encryption)

4.1.4.1.1 TSS

298 The evaluator shall examine the TSS to ensure that it identifies whether the implementation of this cryptographic operation for key encryption (including key lengths and modes) is the same as that used for user data encryption (FCS_COP.1/UDE) or a different implementation.

4.1.4.1.2 Guidance Documentation

299 No additional activities.

4.1.4.1.3 KMD

300 The evaluator shall examine the KMD to ensure that it confirms and is consistent with the identification of the implementation of the key encryption operation as the same or different compared to that used for user data encryption (FCS_COP.1/UDE).

4.1.4.1.4 Tests

301 If the implementation of the key encryption operation is the same as for the user data encryption (FCS_COP.1/UDE) and has been tested with the same key lengths and modes as part of the testing for user data encryption then no further testing is required here. If the key encryption uses a different implementation, (where "different implementation" includes the use of different ley lengths or modes), then the evaluator shall additionally test the key encryption implementation using the corresponding tests specified for FCS_COP.1/UDE.

4.1.4.2 FCS_COP.1/Hash Cryptographic operation (Hash Algorithm)

302 Reference: Secure Hash Algorithm Properties

Algorithm	***Message Size (bits)***	***Block Size (bits)***	***Word Size (bits)***	***Message Digest Size (bits)***
SHA-1	*<2^64*	*512*	*32*	*160*
SHA-224	*<2^64*	*512*	*32*	*224*
SHA-256	*<2^64*	*512*	*32*	*256*
SHA-384	*<2^128*	*1024*	*64*	*384*
SHA-512	*<2^128*	*1024*	*64*	*512*
SHA-512/224	*<2^128*	*1024*	*64*	*224*
SHA-512/256	*<2^128*	*1024*	*64*	*256*

Table 1: SHA Properties

4.1.4.2.1 TSS

303 The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification functions) is documented in the TSS. The evaluator shall also check that the TSS identifies whether the implementation is bit-oriented or byte-oriented.

4.1.4.2.2 Guidance Documentation

304 The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present. The evaluator

also checks the AGD documents to confirm that the instructions for establishing the evaluated configuration use only those hash algorithms selected in the ST.

4.1.4.2.3 Tests

305 The tests below are derived from the "The Secure Hash Algorithm Validation System (SHAVS), Updated: May 21, 2014" from the National Institute of Standards and Technology.

306 The TSF hashing functions can be implemented with one of two orientations. The first is a byte-oriented implementation: this hashes messages that are an integral number of bytes in length (i.e., the length (in bits) of the message to be hashed is divisible by 8). The second is a bit-oriented implementation: this hashes messages of arbitrary length. Separate tests for each orientation are given below.

307 The evaluator shall perform all of the following tests for each hash algorithm and orientation implemented by the TSF and used to satisfy the requirements of this PP. The evaluator shall compare digest values produced by a known-good SHA implementation against those generated by running the same values through the TSF.

308 ***Short Messages Test, Bit-oriented Implementation***

309 The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm in bits (see SHA Properties Table). The length of the messages ranges sequentially from 0 to m bits. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

310 ***Short Messages Test, Byte-oriented Implementation***

311 The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm in bits (see SHA Properties Table). The length of the messages ranges sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

312 ***Selected Long Messages Test, Bit-oriented Implementation***

313 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm in bits (see SHA Properties Table). The length of the ith message is $m + 99*i$, where $1 \leq i \leq$ m. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

314 ***Selected Long Messages Test, Byte-oriented Implementation***

315 The evaluators devise an input set consisting of *m*/8 messages, where *m* is the block length of the hash algorithm in bits (see SHA Properties Table). The length of the *i*th message is $m + 8*99*i$, where $1 \leq i \leq m/8$. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

316 ***Pseudo-randomly Generated Messages Test***

317 The evaluators randomly generate a seed that is *n* bits long, where *n* is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of SHAVS, section 6.4. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

4.1.4.3 FCS_COP.1/HMAC Cryptographic operation (Keyed Hash)

4.1.4.3.1 TSS

318 The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: output MAC length used.

4.1.4.3.2 Guidance Documentation

319 No additional activities.

4.1.4.3.3 Tests

320 This test is derived from The Keyed-Hash Message Authentication Code Validation System (HMACVS). Updated 6 May 2016.

321 The evaluator shall provide 15 sets of messages and keys for each selected hash algorithm and hash length/key size/MAC size combination. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The evaluator shall verify that the resulting HMAC tags match the results from submitting the same inputs to a known-good implementation of the HMAC function, having the same characteristics.

4.1.4.4 FCS_COP.1/SigVer Cryptographic operation (Signature Verification)

4.1.4.4.1 TSS

322 The evaluator shall check the TSS to ensure that it describes the overall flow of the signature verification. This should at least include identification of the format and general location (e.g., "firmware on the hard drive device" rather than “memory location 0x00007A4B") of the data to be used in verifying the digital signature; how the data received from the operational environment are brought onto the device; and any processing that is performed that is not part of

the digital signature algorithm (for instance, checking of certificate revocation lists).

4.1.4.4.2 Guidance Documentation

323 No additional activities.

4.1.4.4.3 Tests

324 Each section below contains tests the evaluators must perform for each selected digital signature scheme. Based on the assignments and selections in the requirement, the evaluators choose the specific activities that correspond to those selections.

325 The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are not found on the TOE in its evaluated configuration.

326 ***SigVer1: RSASSA-PKCS1-v1_5 and SigVer4: RSASSA-PSS***

327 These tests are derived from The 186-4 RSA Validation System (RSA2VS), updated 8 Jul 2014, Section 6.4.

328 The FIPS 186-4 RSA Signature Verification Test tests the ability of the TSF to recognize valid and invalid signatures. The evaluator shall provide a modulus and three associated key pairs (d, e) for each combination of selected modulus size and hash size. Each private key d is used to sign six pseudorandom messages each of 1024 bits. For five of the six messages, the public key (e), message, IR format, padding, or signature is altered so that signature verification should fail. The test passes only if all the signatures made using unaltered parameters result in successful signature verification, and all the signatures made using altered parameters result in unsuccessful signature verification.

329 ***SigVer5: ECDSA on NIST and Brainpool Curves***

330 These tests are derived from The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS), updated 18 Mar 2014, Section 6.5.

331 The FIPS 186-4 ECC Signature Verification Test tests the ability of the TSF to recognize valid and invalid signatures. The evaluator shall provide a modulus and associated key pair (x, y) for each combination of selected curve, modulus size, and hash size. Each private key (x) is used to sign 15 pseudorandom messages of 1024 bits. For eight of the fifteen messages, the message, IR format, padding, or signature is altered so that signature verification should fail. The test passes only if all the signatures made using unaltered parameters result in successful signature verification, and all the signatures made using altered parameters result in unsuccessful signature verification.

332 ***SigVer2: Digital Signature Scheme 2***

333 The following or equivalent steps shall be taken to test the TSF.

334 For each supported modulus size, underlying hash algorithm, and length of the trailer field (1- or 2-byte), the evaluator shall generate N_T sets or recoverable message (M_1), non-recoverable message (M_2), salt, public key and Signature (Σ).

1. NT shall be greater than or equal to 20.

2. The length of salts shall be selected from its supported length range of salt. The typical length of salt is equal to the output block length of underlying hash algorithm (see 9.2.2 of ISO/IEC 9796-2:2010).

3. The length of recoverable messages should be selected by considering modulus size, output block length of underlying hash algorithm, and length of salt (L_S). As described in Annex D of ISO/IEC 9796-2:2010, it is desirable to maximise the length of recoverable message. The following table shows the maximum bit-length of recoverable message which is divisible by 512, for some combinations of modulus size, underlying hash algorithm, and length of salt.

Maximum length of recoverable message divisible by 512 (bits)	Modulus size (bits)	Underlying hash algorithm (bits)	Length of salt L_S (bits)
1536	2048	SHA-256	128
1024			256
1024		SHA-512	128
1024			256
512			512
2560	3072	SHA-256	128
2048			256
2048		SHA-512	128
2048			256
1536			512
Note that 2-byte trailer field is assumed in calculating the maximum length of recoverable message.			

Table 2: SigVer2 Test Lengths

4. The length of non-recoverable messages should be selected by considering the underlying hash algorithm and usage(s). If the TSF is used for verifying the authenticity of software/firmware updates, the length of non-recoverable messages should be selected greater than or equal to 2048-bit. With this length range, it means that the underlying hash algorithm is also tested for two or more input blocks.

5. The evaluator shall select approximately one half of N_T sets and shall alter one of the values (non-recoverable message, public key exponent or signature) in the sets. In altering public key exponent, the evaluator shall alter the public key exponent while keeping the exponent odd. In altering signatures, the following ways should be considered:

    i. Altering a signature just by replacing a bit in the bit-string representation of the signature

    ii. Altering a signature so that the trailer in the message representative cannot be interpreted. This can be achieved by following ways:

        — Setting the rightmost four bits of the message representative to the values other than '1100'.

        — In the case when 1-byte trailer is used, setting the rightmost byte of the message representative to the values other than '0xbc', while keeping the rightmost four bits to '1100'.

        — In the case when 2-byte trailer is used, setting the rightmost byte of the message representative to the values other than '0xcc', while keeping the rightmost four bits to '1100'.

    iii. In the case when 2-byte trailer is used, altering a signature so that the hash algorithm identifier in the trailer (i.e. the left most byte of the trailer) does not correspond to hash algorithm(s) identified in the SFR. The hash algorithm identifiers are 0x34 for SHA-256 (see Clause 10 of ISO/IEC 10118-3:2004), and 0x35 for SHA-512 (see Clause 11 of ISO/IEC 10118-3:2004).

    iv. Let L_S be the length of salt, altering a signature so that the intermediate bit string D in the message representative is set to all zeros except for the rightmost L_S bits of D.

    v. (non-conformant signature length) altering a signature so that the length of signature $\sum$ is changed to modulus size and the most significant bit of signature $\sum$ is set equal to '1'.

    vi. (non-conformant signature) altering a signature so that the integer converted from signature $\sum$ is greater than modulus n.

335 The evaluator shall supply the N_T sets to the TSF and obtain in response a set of N_T Verification-Success or Verification-Fail values. When the Verification-Success is obtained, the evaluator shall also obtain recovered Message (M_1*).

336 The evaluator shall verify the Verification-Success results correspond to the unaltered sets and Verification-Fail results correspond to the altered sets.

337 For each recovered message, the evaluator shall compare the recovered message (M_1*) with the corresponding recoverable message (M_1) in the unaltered sets.

338 The test passes only if all the signatures made using unaltered sets result in Verification-Success, each recovered message (M_1*) is equal to corresponding M_1 in the unaltered sets, and all the signatures made using altered sets result in Verification-Fail.

339 ***SigVer3: Digital Signature Scheme 3***

340 The evaluator shall perform the test described in ***SigVer2: Digital Signature Scheme 2*** while using a fixed salt for N_T sets.

4.1.5 Random Bit Generation (FCS_RBG_EXT)

4.1.5.1 FCS_RBG_EXT.2 Random Bit Generation (External Seeding)

4.1.5.1.1 TSS

341 *<TBD>*

4.1.5.1.2 Guidance Documentation

342 *<TBD>*

4.1.5.1.3 Tests

343 *<TBD>*

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

4.1.5.2.1 TSS

344 The evaluator will verify that the TSS documents the types of noise sources selected in FCS_RBG_EXT.3.1 and indicates the minimum amount of min-entropy provided by these sources. If this SFR is iterated, the evaluator shall check that the TSS indicates the purpose for each entropy source (e.g., initialization or reseed) and that the output from these entropy sources is not later combined into a single seed.

4.1.5.2.2 Guidance Documentation

345 The evaluator will check that the Operational Guidance describes any settings, operational requirements, or user input necessary for the proper function of the noise sources.

4.1.5.2.3 Entropy Documentation and Assessment (EAR)

346 The developer shall produce documentation and the evaluator shall perform evaluation activities in accordance with Appendix XX: Entropy Documentation and Assessment. When multiple noise sources are used to provide the minimum amount of min-entropy, the Entropy Documentation must demonstrate that entropy from each of these individual sources is generated independently.

4.1.5.2.4 Tests

347 *<TBD>*

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

4.1.5.3.1 TSS

348 *<TBD>*

4.1.5.3.2 Guidance Documentation

349 *<TBD>*

4.1.5.3.3 Tests

350 *<TBD>*

4.1.5.4 FCS_RBG_EXT.5 Random Bit Generation (Combining Noise Sources)

4.1.5.4.1 TSS

351 *<TBD>*

4.1.5.4.2 Guidance Documentation

352 *<TBD>*

4.1.5.4.3 Tests

353 *<TBD>*

354

4.2 Identification and Authentication (FIA)

4.2.1 Passphrase support (FIA_PPS_EXT)

4.2.1.1 FIA_PPS_EXT.2/num Passphrase composition - numeric

4.2.1.1.1 TSS

355 The evaluator shall examine the TSS to ensure that it describes the manner in which the TOE enforces the composition of passphrases, including the length, and requirements on characters.

4.2.1.1.2 Operational Guidance

356 The evaluator shall examine the operational guidance to ensure it provides guidance on the composition of passphrases, including the length, and requirements on characters.

4.2.1.1.3 Test

357 The evaluator shall perform the following test:

358 Test 1: The evaluator shall compose two types of passphrases - those specifically designed to meet the requirements and others designed to fail. For each passphrase, the evaluator shall verify that the TOE mechanism rejects the passphrase if it contains less than 8 characters. While the evaluator is not required (nor is it feasible) to test all possible compositions of passphrases, the evaluator shall ensure that the minimum and maximum length listed in the requirement is supported, and justify the subset of those characters chosen for testing.

4.2.1.2 FIA_PPS_EXT.2/alph Passphrase composition - alphanumeric

4.2.1.2.1 TSS

359 The evaluator shall examine the TSS to ensure that it describes the manner in which the TOE enforces the composition of passphrases, including the length, and requirements on characters.

4.2.1.2.2 Operational Guidance

360 The evaluator shall examine the operational guidance to ensure it provides guidance on the composition of passphrases, including the length, and requirements on characters.

4.2.1.2.3 Test

361 The evaluator shall perform the following test:

362 Test 1: The evaluator shall compose two types of passphrases - those specifically designed to meet the requirements and others designed to fail. For each passphrase, the evaluator shall verify that the TOE mechanism rejects the

passphrase if it contains less than 8 characters. While the evaluator is not required (nor is it feasible) to test all possible compositions of passphrases, the evaluator shall ensure that the minimum and maximum length listed in the requirement is supported, and justify the subset of those characters chosen for testing.

4.2.2 User authentication (FIA_UAU)

4.2.2.1 FIA_UAU.7 Protected authentication feedback

4.2.2.1.1 TSS

363 The evaluator shall check that the TSS describes how the TOE obscures feedback while authorisation on the device is in progress.

4.2.2.1.2 Test

364 The evaluator shall perform the following test for each method of authorisation allowed on the device:

365 Test 1: The evaluator shall enter authorisation data on the TOE. While making this attempt, the evaluator shall verify that any feedback provided is obscured while entering the authorisation data.

4.3 Security Management (FMT)

4.3.1 Specification of Management Functions (FMT_SMF)

4.3.1.1 FMT_SMF.1 Specification of Management Functions

4.3.1.1.1 TSS

366 The evaluator shall examine the TSS to determine that management functions included in FMT_SMF are described.

4.3.1.1.2 Operational Guidance

367 The evaluation shall review the operational guidance to ensure that it contains instructions on how the authorised user can:

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

4.3.1.1.3 Test

368 The evaluator shall perform the following tests:

369 Test 1: (optional)The evaluator shall set a valid number of unsuccessful authentication attempts within the range of acceptable values using instruction provided in the operational guidance and verify that configuration was successful. This test is not applicable for devices that do not allow users to configure a number of unsuccessful authentication attempts.

370 Test 2: (optional) The evaluator shall set an invalid number of unsuccessful authentication attempts using instruction provided in the operational guidance and verify that configuration was unsuccessful. The evaluator shall set numbers that are greater than and less than the number in the accepted range. This test is not applicable for devices that do not allow users to configure a number of unsuccessful authentication attempts.

371 Test 3: (optional) The evaluator shall define a user configurable number of unsuccessful authentication attempts within a range of acceptable values defined in FIA_AFL.1. The evaluator shall enter invalid authorisation factor the configured number of times to verify that the encrypted user data is no longer accessible to the users. This test is not applicable for devices that do not allow users to configure a number of unsuccessful authentication attempts.

372 Test 4: (optional) The evaluator shall disable the data recovery mechanism and verify that the data on the device could not be recovered. This test is not applicable for devices that do not provide data recovery mechanism.

373 Test 5: (optional) The evaluator shall enable data recovery mechanism. In order to ensure that the new DEK has been generated. The evaluator shall try to access use data that was previously stored on the device**.** This test is not applicable for devices that do not provide data recovery mechanism.

5 Evaluation Activities for SARs

374 The sections below specify EAs for the Security Assurance Requirements (SARs) included in the related cPPs. The EAs in Section 2 (*), Section 3 (Evaluation Activities for Optional Requirements), and Section 4 (Evaluation Activities for Selection-Based Requirements)* are an interpretation of the more general CEM assurance requirements as they apply to the specific technology area of the TOE.

375 In this section, each SAR that is contained in the cPP is listed, and the EAs that are not associated with an SFR are captured here, or a reference is made to the CEM, and the evaluator is expected to perform the CEM work units.

5.1 ASE: Security Target Evaluation

376 When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2 (*)*.

5.2 ADV: Development

5.2.1 Basic Functional Specification (ADV_FSP.1)

377 The EAs for this assurance component focus on understanding the interfaces (e.g., application programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2 (*)*, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

378 The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

379 The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

380 The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional "functional specification" documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their

assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7 is treated as implicit and no separate mapping information is required for this element.

CEM ADV_FSP.1 Work Units	Evaluation Activities
ADV_FSP.1-1 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each SFR-supporting and SFR-enforcing TSFI.	5.2.1.1 Evaluation Activity: *The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.*
ADV_FSP.1-2 The evaluator ***shall examine*** the functional specification to determine that the method of use for each SFR-supporting and SFR-enforcing TSFI is given.	5.2.1.2 Evaluation Activity: *The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.*
ADV_FSP.1-3 The evaluator ***shall examine*** the presentation of the TSFI to determine that it identifies all parameters associated with each SFR-enforcing and SFR supporting TSFI.	5.2.1.3 Evaluation Activity: The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.
ADV_FSP.1-4 The evaluator ***shall examine*** the rationale provided by the developer for the implicit categorisation of interfaces as SFR-non-interfering to determine that it is accurate.	Paragraph 561 from the CEM: "In the case where the developer has provided adequate documentation to perform the analysis called for by the rest of the work units for this component without explicitly identifying SFR-enforcing and SFR-supporting interfaces, this work unit should be considered satisfied." Since the rest of the ADV_FSP.1 work units will have been satisfied upon completion of the EAs, it follows that this work unit is satisfied as well.
ADV_FSP.1-5 The evaluator ***shall check*** that the tracing links the SFRs to the corresponding TSFIs.	5.2.1.4 Evaluation Activity: The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.
ADV_FSP.1-6 The evaluator ***shall examine*** the functional specification to determine that it is	EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3

a complete instantiation of the SFRs.	and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e., at the TSFI) are covered. Therefore, the intent of this work unit is covered.
ADV_FSP.1-7 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the SFRs.	EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3 and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e., at the TSFI) are addressed, and that the description of the interfaces is accurate with respect to the specification captured in the SFRs. Therefore, the intent of this work unit is covered.

Table 3: Mapping of ADV_FSP.1 CEM Work Units to Evaluation Activities

5.2.1.1 Evaluation Activity

381 The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

382 In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g., audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent, is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

383 The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

5.2.1.2 Evaluation Activity

384 The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

5.2.1.3 Evaluation Activity

385 The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

386 The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2 (), including the EAs associated with testing of the interfaces.

387 It should be noted that there may be some SFRs that do not have an interface that is explicitly "mapped" to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

388 However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

5.3 AGD: Guidance Documents

389 It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and the AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to administrators and users (as appropriate) as part of the TOE.

5.3.1 Operational User Guidance (AGD_OPE.1)

390 The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

391 In addition, the evaluator performs the EAs specified below.

5.3.1.1 Evaluation Activity

392 The evaluator shall ensure the Operational guidance documentation is distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

5.3.1.2 Evaluation Activity

393 The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

5.3.1.3 Evaluation Activity

394 The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

5.3.1.4 Evaluation Activity

395 The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

5.3.2 Preparative Procedures (AGD_PRE.1)

396 The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

397 Preparative procedures are distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

398 In addition, the evaluator performs the EAs specified below.

5.3.2.1 Evaluation Activity

399 The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

400 The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

5.3.2.2 Evaluation Activity

401 The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

5.3.2.3 Evaluation Activity

402 The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

5.3.2.4 Evaluation Activity

403 The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

5.4 ALC: Life-cycle Support

5.4.1 Labelling of the TOE (ALC_CMC.1)

404 When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

5.4.2 TOE CM coverage (ALC_CMS.1)

405 When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

5.5 ATE: Tests

5.5.1 Independent Testing – Conformance (ATE_IND.1)

406 The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

407 The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Section 2: .

5.6 AVA: Vulnerability Assessment

408 <The iTC plays a key role in determining the scope of the vulnerability analysis with respect to what is publicly reported. The iTC must perform several activities to complete sections of this Supporting Document in order to ensure the flaws investigated by the evaluation team are meaningful in the context of the cPP and cover the areas of concern by the iTC for this technology.

409 There are four activities (and associated outputs) that need to be performed by the iTC:

410 1) identification of public sources of vulnerability information and actions to be taken on that information (this will be used for Type 1 flaw hypotheses as defined in Appendix A);

411 2) identification of specific vulnerabilities particular to the technology (perhaps from previous evaluations, or from flaw reports to vendors that are part of the iTC) (this will be used for Type 2 flaw hypotheses as defined in Appendix A);

412 3) identification of additional documentation to be used in the vulnerability analysis activity (this will be used for Type 3 flaw hypotheses as defined in Appendix A); and

413 4) identification of any tools—and actions to be performed with those tools—to support flaw identification (this will be used for Type 4 flaw hypotheses as defined in Appendix A).

414 Each of these activities is discussed in more detail below, with pointers to where the output of each activity should go in this Supporting Document.>

5.6.1 Vulnerability Survey (AVA_VAN.1)

415 While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

416 In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

CEM AVA_VAN.1 Work Units	Evaluation Activities
AVA_VAN.1-1 The evaluator ***shall examine*** the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.	The evaluator shall perform the CEM activity as specified. *If the iTC specifies any tools to be used in performing this analysis in section A.3.4, the following text is also included in this cell: "The calibration of test resources specified in paragraph 1418 of the CEM*

	applies to the tools listed in Appendix A, Section A.1.4."
AVA_VAN.1-2 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state	The evaluator shall perform the CEM activity as specified.
AVA_VAN.1-3 The evaluator ***shall examine*** sources of information publicly available to identify potential vulnerabilities in the TOE.	Replace CEM work unit with activities outlined in Appendix A, Section 1
AVA_VAN.1-4 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.	Replace the CEM work unit with the analysis activities on the list of potential vulnerabilities in Appendix A, section 1, and documentation as specified in Appendix A, Section 1.
AVA_VAN.1-5 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.	Replace the CEM work unit with the activities specified in Appendix A, section 1.
AVA_VAN.1-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include: a) identification of the potential vulnerability the TOE is being tested for; b) instructions to connect and setup all required test equipment as required to conduct the penetration test; c) instructions to establish all penetration test prerequisite initial conditions; d) instructions to stimulate the TSF; e) instructions for observing the behaviour of the TSF; f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against	The CEM work unit is captured in Appendix A, Section 1; there are no substantive differences.

expected results; g) instructions to conclude the test and establish the necessary post-test state for the TOE.	
AVA_VAN.1-7 The evaluator ***shall conduct*** penetration testing.	The evaluator shall perform the CEM activity as specified. See Appendix A, Section 1, paragraph 493 for guidance related to attack potential for confirmed flaws.
AVA_VAN.1-8 The evaluator ***shall record*** the actual results of the penetration tests.	The evaluator shall perform the CEM activity as specified.
AVA_VAN.1-9 The evaluator ***shall report*** in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.	Replace the CEM work unit with the reporting called for in Appendix A, Section 1.
AVA_VAN.1-10 The evaluator ***shall examine*** the results of all penetration testing to determine that the TOE, in its operational environment, is resistant to an attacker possessing a Basic attack potential.	This work unit is not applicable for Type 1 and Type 2 flaws (as defined in Appendix A, Section 1), as inclusion in this Supporting Document by the iTC makes any confirmed vulnerabilities stemming from these flaws subject to an attacker possessing a Basic attack potential. This work unit is replaced for Type 3 and Type 4 flaws by the activities defined in Appendix A, Section 1, paragraph 493.
AVA_VAN.1-11 The evaluator ***shall report*** in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each: a) its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication); b) the SFR(s) not met; c) a description; d) whether it is exploitable in its operational environment or not (i.e. exploitable or residual). e) the amount of time, level of expertise, level of knowledge of the TOE, level of opportunity and the equipment required to perform the	Replace the CEM work unit with the reporting called for in Appendix A, Section 1.

identified vulnerabilities, and the corresponding values using the tables 3 and 4 of Annex B.4.	

Table 4: Mapping of AVA_VAN.1 CEM Work Units to Evaluation Activities

417 Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A, while an "outline" of the assurance activity is provided below.

5.6.1.1 Evaluation Activity (Documentation):

418 *<If the iTC determines that no additional documentation beyond that specified below is required, it is acceptable to remove this Evaluation Activity in the Supporting Document.*

419 *If the iTC determines that additional documentation is appropriate, they will insert a description of that documentation in this paragraph. The iTC must specify the required documentation in as much detail as possible to eliminate issues associated with the evaluators evaluating the suitability of the documentation rather than using the documentation to evaluate the product. Therefore, documentation statements such as "Supply a high-level and low-level design" are discouraged. An example of a better statement is:*

420 *"The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components apply to all systems claimed in the ST, and should identify at a minimum the processors used by the TOE. Software components include any libraries used by the TOE, such as cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.">*

421 *The evaluator shall examine the documentation outlined below provided by the vendor to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.*

422 In addition to the activities specified by the CEM in accordance with Table 2 above, the evaluator shall perform the following activities.

5.6.1.2 Evaluation Activity

423 *The evaluator formulates hypotheses in accordance with process defined in Appendix* 1. *The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix* 1. *The evaluator shall perform vulnerability analysis in accordance with Appendix* 1. *The results of the analysis shall be documented in the report according to Appendix* 1.

6 Required Supplementary Information

424 This Supporting Document refers in various places to the possibility that 'required supplementary information' may need to be supplied as part of the deliverables for an evaluation. This term is intended to describe information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public.

425 The USP cPP requires an entropy analysis ([USBcPP, D.1]), and a Key Management and Data Storage Description ([USBcPP, D.2]). The evaluation activities that the evaluator is to perform with those documents are captured under the appropriate SFRs in sections 2-5.

7 References

[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model CCMB-2012-09-001, Version 3.1 Revision 4, September 2012

[CC2] Common Criteria for Information Technology Security Evaluation,
Part 2: Security Functional Components,
CCMB-2012-09-002, Version 3.1 Revision 4,
September 2012

[CC3] Common Criteria for Information Technology Security Evaluation,
Part 3: Security Assurance Components,
CCMB-2012-09-003, Version 3.1 Revision 4, September 2012

[CEM] Common Methodology for Information Technology Security Evaluation, CCMB-2012-09-004, Version 3.1 Revision 4, September 2012

[USBcPP] collaborative Protection Profile for USB Portable Storage Devices, <*Other details TBD*>

Appendixes

A. Vulnerability Analysis

A.1 Sources of vulnerability information

426 CEM Work Unit AVA_VAN.1-3 has been supplemented in this Supporting Document to provide a better-defined set of flaws to investigate and procedures to follow based on this particular technology. Terminology used is based on the flaw hypothesis methodology, where the evaluation team hypothesizes flaws and then either proves or disproves those flaws (a flaw is equivalent to a "potential vulnerability" as used in the CEM). Flaws are categorized into four "types" depending on how they are formulated:

1. A list of flaw hypotheses applicable to the technology described by the cPP derived from public sources as documented in Section A.1.1—this fixed set has been agreed to by the iTC. Additionally, this will be supplemented with entries for a set of public sources (as indicated below) that are directly applicable to the TOE or its identified components (as defined by the process in Section A.1.1 below); this is to ensure that the evaluators include in their assessment applicable entries that have been discovered since the cPP was published;
2. A list of flaw hypotheses contained in this document that are derived from lessons learned specific to that technology and other iTC input (that might be derived from other open sources and vulnerability databases, for example) as documented in Section A.1.2;
3. A list of flaw hypotheses derived from information available to the evaluators; this includes the baseline evidence provided by the vendor described in this Supporting Document (documentation associated with EAs, documentation described in Section 5.6.1.1, <*the iTC can remove the reference to Section 5.6.1.1 if no additional documentation is defined*> documentation described in Section 6), as well as other information (public and/or based on evaluator experience) as documented in Section A.1.3; and
4. A list of flaw hypotheses that are generated through the use of iTC-defined tools (e.g., nmap, protocol testers) and their application is specified in section A.1.4.

A.1.1 Type 1 Hypotheses—Public-Vulnerability-based

427 <The iTC must determine what public vulnerability databases are to be used as the basis for Type 1 hypotheses, and what entries in these databases apply. A sample list of resources is contained in Appendix D, but the iTC is not bound by that list.

428 In performing this activity, the iTC first agrees upon the sources to be used. The list of sources should be searched by the iTC with an agreed-upon set of terms such that the iTC feels a representative set of vulnerabilities with respect to the technology type is returned.

429 Having identified the sources, for each source the iTC defines criteria for selecting entries in the list. The lists and criteria should be identified in this section of the Supporting Document so that evaluators can use the same sources and criteria at evaluation time to select entries that were made after the cPP was published. For each entry that meets the criteria, the iTC determines whether or not to include it in the list of flaw hypotheses defined in this Supporting Document. This will likely necessitate the creation of some criteria by which to judge an entry that is agreed to by the iTC. For instance, CVEs that would generate flaw hypotheses related to buffer overflows would probably be rejected as a generic flaw hypothesis. The output of this activity is a list of specific entries from the selected sources that will be used as flaw hypotheses.>

430 The following list of public sources of vulnerability information was selected by the iTC:

431 The Common Vulnerabilities and Exposures database at http://cve.mitre.org/cve. The same database is also available at https://nvd.nist.gov.

432 ***<iTC comment: no other vulnerability database appear to add anything important to the search results from the CVE database >***

433 The list of sources above was searched with the following search terms:

434 "USB", "flash drive", "USB drive", "USB flash"

435 It should be noted that any attacks on the communication between the USB device and the host computer, or using the host computer, are out of scope since the protected data is available as plaintext here.

436 ***<iTC comment: The search results for "USB" contains all relevant search results for any of the other search terms, including many others, not listed above>***

437 **< *iTC comment:* Shouldn't there be a list of type 1 flaw hypotheses here, derived from database searches? Also, it would be relevant to specify some criteria for excluding search hits from consideration. In any case I present my results below and exclusion criteria above>**

438 No potential vulnerabilities applicable to the cPP was found by the iTC.

439 **< *iTC comment:* Several relevant potential flaws were found, but since these did were not reported as flaws in USB mass storage devices, but in other devices, the flaw hypotheses are listed as type 2>**

440 In order to supplement this list, the evaluators shall also perform a search on the sources listed above to determine a list of potential flaw hypotheses that are more recent that the publication date of the cPP, and those that are specific to the TOE and its components as specified by the additional documentation mentioned above. Any duplicates – either in a specific entry, or in the flaw

hypothesis that is generated from an entry from the same or a different source – can be noted and removed from consideration by the evaluation team.

441 As part of type 1 flaw hypothesis generation for the specific components of the TOE, the evaluator shall also search the component manufacturer's websites to determine if flaw hypotheses can be generated on this basis (for instance, if security patches have been released for the version of the component being evaluated, the subject of those patches may form the basis for a flaw hypothesis).

442 The Common Vulnerabilities and Exposures database at http://cve.mitre.org/cve should be searched for occurrences of the name and version of the USB controller and the cryptographic library used in the device. Any vulnerabilities found, that are applicable to the implemented versions of these components, shall be presented as flaw hypotheses.

A.1.2 Type 2 Hypotheses—iTC-Sourced

443 <The iTC must consider if there are any technology-specific vulnerabilities or types of vulnerabilities that the evaluators should consider that are not contained in the previous section. This could be based on previous evaluations against the cPP, experience of the iTC members, or other factors. These vulnerabilities should be limited to those exploitable with a Basic Attack Potential—as characterized by the time, technical expertise, knowledge of the TOE, equipment, and access needed for exploitation. Section B.4.2.2. of the CEM provides detailed guidance on how these factors should be considered in determining attack potential relative to vulnerabilities.

444 This set of vulnerabilities (Type 2) is listed below and would then need to be considered by the evaluation team. It is likely that there will be few or no entries identified for this type until more experience is gained with the cPP.>

445 The following list of flaw hypothesis generated by the iTC for this technology must be considered by the evaluation team as flaw hypotheses in performing the vulnerability assessment.

446 Flaw hypothesis type 2 number 1

447 Hypothesis:

448 Plaintext, key material, and intermediate results from DEK decryption may be left in persistent memory, or in powered volatile storage (if there is a power source in the device). Both when the device is unplugged prematurely and after the read/write operations have been completed need to be considered.

449 In combination with one of the potential flaws below, or by physically connecting to memory circuits in the device, plaintext, key material or intermediate DEK decryption results can be extracted.

450 Discussion:

451 The risk can be eliminated by verifying that no plaintext, key material, or intermediate DEK decryption results remains in memory after completed operation or after unplugging the device prematurely. This can be verified in conjunction with the (ATE) testing whether the keys are erased after operation. The iTC estimated these attacks to be exploitable for a basic attacker.

452 Flaw hypothesis type 2 number 2

453 Hypothesis:

454 There may be a privileged interface left available, that provides easy access to firmware, configuration parameters, key material and user data in the memory areas in the device. Possible examples could be debug interfaces, JTAG or similar.

455 This potential flaw could be exploited to change the configuration to allow unlimited password attempts, which would make it feasible to stage a brute force attack against the password. It could also be exploited to extract encrypted DEK and data to perform a brute force attack against the password outside the device.

456 Discussion:

457 The risk can be eliminated by trying to connect to any such interfaces that have been accessible during the development or production of the device. The existence/nonexistence of such interfaces should first be verified using the design documentation, and by asking the developer. For these attacks the iTC estimated the attack potential to be enhanced basic.

458 Flaw hypothesis type 2 number 3

459 Hypothesis:

460 It may be possible to update the firmware in the device through the USB interface, which enables an attacker to extract the encrypted DEK and the encrypted user data and perform an unlimited brute force attack against the password outside the device (the updated firmware itself can also perform the brute force attack within the device).

461 Discussion:

462 The risk can be eliminated by verifying that any features for updating the firmware have been disabled. The availability of such a feature should first be verified using the design documentation, and by asking the developer, then the evaluator should try to use the feature. For these attacks the iTC estimated the attack potential to be moderate.

463 Flaw hypothesis type 2 number 4

464 Hypothesis:

465 There may exist exploitable buffer overflow vulnerabilities in the firmware, that could provide access to firmware, configuration parameters, key material and user data in the memory areas in the device.

466 This potential flaw could be used to change the configuration to allow unlimited password attempts, which would make it feasible to stage a brute force attack against the password. It could also be used to extract encrypted DEK and data to perform a brute force attack against the password outside the device.

467 Discussion:

468 The risk of having easily found buffer overflow flaws can be reduced by fuzz testing. However, attacks using buffer overflows were estimated by the iTC at attack potential high, rendering this a residual vulnerability with margin. If exploits are developed and made publicly known, this may change.

469 **<iTC comment: The evaluator should re-calculate the attack potentials during the evaluation and consider the existence of exploits that would simplify the attacks>**

470 **<iTC comment: It should be noted that there are at least one relevant attack that does not need any flaw, that is estimated to be exploitable with an attack potential of moderate. This is when the attacker accesses the memory circuits physically, extracts encrypted user data and encrypted DEK and performs a brute force attack against the password to decrypt first the DEK, then the data. The physical access to circuits can be made much more difficult by covering the circuits in epoxy.>**

471 If the evaluators discover a Type 3 or Type 4 flaw that they believe should be considered as a Type 2 flaw in future versions of this cPP, they should work with their Certification Body to determine the appropriate means of submitting the flaw for consideration by the iTC.

472 ***<iTC comment: the above paragraph overlaps with the last statements in section A.1.3 and A.1.4. Should it be removed?>***

A.1.3 Type 3 Hypotheses—Evaluation-Team-Generated

473 Type 3 flaws are formulated by the evaluator based on information presented by the product (through on-line help, product documentation and user guides, etc.) and product behaviour during the (functional) testing activities. The evaluator is also free to formulate flaws that are based on material that is not part of the baseline evidence (e.g., information gleaned from an Internet mailing list, or reading interface documentation on interfaces not included in the set provided by the developer), although such activities have the potential to vary significantly based upon the product and evaluation facility performing the analysis.

474 If the evaluators discover a Type 3 flaw that they believe should be considered as a Type 2 flaw in future versions of this cPP, they should work with their

Certification Body to determine the appropriate means of submitting the flaw for consideration by the iTC.

475 <It may be the case that no activities of this type are appropriate for this technology; in that case, this section can be removed and references in other areas of this Supporting Document adjusted accordingly.>

A.1.4 Type 4 Hypotheses—Tool-Generated

476 No need for a tool based search for vulnerabilities is foreseen by the iTC.

477 ***<iTC comment: Fuzzing the USB interface would be relevant, but the attacks that would use buffer overflows also would have to go further and for example grab encrypted data and keys and then brute-force the password. These combined attacks were estimated to correspond to an attack potential of "high" and thus result in residual vulnerabilities. If a buffer overflow exploit is published, the attack potential for using this would be considerably lower. If this happens, the evaluator may propose to the scheme that fuzz testing is used.>***

478 If the evaluators discover a Type 4 flaw that they believe should be considered as a Type 2 flaw in future versions of this cPP, they should work with their Certification Body to determine the appropriate means of submitting the flaw for consideration by the iTC.

A.2 Process for Evaluator Vulnerability Analysis

479 As flaw hypotheses are generated from the activities described above, the evaluation team will disposition them; that is, attempt to prove, disprove, or determine the non-applicability of the hypotheses. This process is as follows.

480 The evaluator will refine each flaw hypothesis for the TOE and attempt to disprove it using the information provided by the developer or through penetration testing. During this process, the evaluator is free to interact directly with the developer to determine if the flaw exists, including requests to the developer for additional evidence (e.g., detailed design information, consultation with engineering staff); however, the CB should be included in these discussions. Should the developer object to the information being requested as being not compatible with the overall level of the evaluation activity/cPP and cannot provide evidence otherwise that the flaw is disproved, the evaluator prepares an appropriate set of materials as follows:

481 the source documents used in formulating the hypothesis, and why it represents a potential compromise against a specific TOE function;

482 an argument why the flaw hypothesis could not be proven or disproved by the evidence provided so far; and

483 the type of information required to investigate the flaw hypothesis further.

484 The Certification Body (CB) will then either approve or disapprove the request for additional information. If approved, the developer provides the requested evidence to disprove the flaw hypothesis (or, of course, acknowledge the flaw).

485 For each hypothesis, the evaluator will note whether the flaw hypothesis has been successfully disproved, successfully proven to have identified a flaw, or requires further investigation. It is important to have the results documented as outlined in Section 1 below.

486 If the evaluator finds a flaw, the evaluator must report these flaws to the developer. All reported flaws must be addressed as follows:

487 If the developer confirms that the flaw exists and that it is exploitable at Basic Attack Potential, then a change is made by the developer, and the resulting resolution is agreed by the evaluator and noted as part of the evaluation report.

488 If the developer, the evaluator, and the CB agree that the flaw is exploitable only above Basic Attack Potential and does not require resolution for any other reason, then no change is made and the flaw is noted as a residual vulnerability in the CB-internal report (ETR).

489 If the developer and evaluator agree that the flaw is exploitable only above Basic Attack Potential, but it is deemed critical to fix because of technology-specific or cPP-specific aspects such as typical use cases or operational environments, then a change is made by the developer, and the resulting resolution is agreed by the evaluator and noted as part of the evaluation report.

490 Disagreements between evaluator and vendor regarding questions of the existence of a flaw, its attack potential, or whether it should be deemed critical to fix are resolved by the CB.

491 Any testing performed by the evaluator shall be documented in the test report as outlined in Section 1 below.

492 As indicated in Section 1, Reporting, the public statement with respect to vulnerability analysis that is performed on TOEs conformant to the cPP is constrained to coverage of flaws associated with Types 1 and 2 (defined in Section 1) flaw hypotheses only. The fact that the iTC generates these candidate hypotheses indicates these must be addressed.

493 For flaws of Types 3 and 4, each CB is responsible for determining what constitutes Basic Attack Potential for the purposes of determining whether a flaw is exploitable in the TOE's environment. The determination criteria shall be documented in the CB-internal report as specified in Section 1. As this is a per-CB activity, no public claims are made with respect to the resistance of a particular TOE against flaws of Types 3 and 4; rather, the claim is that the activities outlined in this appendix were carried out, and the evaluation team and CB agreed that any residual vulnerabilities are not exploitable by an attacker with Basic Attack Potential.

A.3 Reporting

494 The evaluators shall produce two reports on the testing effort; one that is public-facing (that is, included in the non-proprietary evaluation report, which is a subset of the Evaluation Technical Report (ETR)), and the complete ETR that is delivered to the overseeing CB.

495 The public-facing report contains:

496 * The flaw identifiers returned when the procedures for searching public sources were followed according to instructions in the Supporting Document per Section A.1.1;

497 * A statement that the evaluators have examined the Type 1 flaw hypotheses specified in this Supporting Document in section A.1.1 (i.e. the flaws listed in the previous bullet) and the Type 2 flaw hypotheses specified in this Supporting Document by the iTC in Section A.1.2.

498 <The above two bullets encompass all flaw hypotheses of Type 1 and Type 2.>

499 * A statement that the evaluation team developed Types 3 and 4 flaw hypotheses in accordance with Sections A.1.3, A.1.4, and 1, and that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential as defined by the CB in accordance with the guidance in the CEM. It should be noted that this is just a statement about the "fact of" Types 3 and 4 flaw hypotheses being developed, and that no specifics about the number of flaws, the flaws themselves, or the analysis pertaining to those flaws will be included in the public-facing report.

500 No other information is provided in the public-facing report.

501 The internal CB report contains, in addition to the information in the public-facing report:

- a list of all of the flaw hypotheses generated (cf. AVA_VAN.1-4);
- the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results (cf. AVA_VAN.1-9);
- all documentation used to generate the flaw hypotheses (in identifying the documentation used in coming up with the flaw hypotheses, the evaluation team must characterize the documentation so that a reader can determine whether it is strictly required by this Supporting Document, and the nature of the documentation (design information, developer engineering notebooks, etc.));
- the evaluator shall report all exploitable vulnerabilities and residual vulnerabilities, detailing for each:
    - its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);
    - the SFR(s) not met;
    - a description;

  - whether it is exploitable in its operational environment or not (i.e. exploitable or residual).
  - the amount of time, level of expertise, level of knowledge of the TOE, level of opportunity and the equipment required to perform the identified vulnerabilities (cf. AVA_VAN.1-11);
- how each flaw hypothesis was resolved (this includes whether the original flaw hypothesis was confirmed or disproved, and any analysis relating to whether a residual vulnerability is exploitable by an attacker with Basic Attack Potential) (cf. AVA_VAN1-10); and
- in the case that actual testing was performed in the investigation (either as part of flaw hypothesis generation using tools specified by the iTC in Section A.1.4, or in proving/disproving a particular flaw) the steps followed in setting up the TOE (and any required test equipment); executing the test; post-test procedures; and the actual results (to a level of detail that allow repetition of the test, including the following:
  - identification of the potential vulnerability the TOE is being tested for;
  - instructions to connect and setup all required test equipment as required to conduct the penetration test;
  - instructions to establish all penetration test prerequisite initial conditions;
  - instructions to stimulate the TSF;
  - instructions for observing the behaviour of the TSF;
  - descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;
  - instructions to conclude the test and establish the necessary post-test state for the TOE. (cf. AVA_VAN.1-6, AVA_VAN.1-8).

B. Equivalency Considerations

B.1 Introduction

502 This appendix provides a foundation for evaluators to determine whether a vendor's request for equivalency of products is allowed.

503 For the purpose of this evaluation, equivalency can be broken into two categories:

- **Variations in models**: Separate TOE models/variations may include differences that could necessitate separate testing across each model. If there are no variations in any of the categories listed below, the models may be considered equivalent.
- **Variations in TOE dependencies on the environment (e.g., OS/platform the product is tested on):** The method a TOE provides functionality (or the functionality itself) may vary depending upon the environment on which it is installed. If there is no difference in the TOE-provided functionality or in the manner in which the TOE provides the functionality, the models may be considered equivalent.

504 Determination of equivalency for each of the above specified categories can result in several different testing outcomes.

505 If a set of TOE are determined to be equivalent, testing may be performed on a single variation of the TOE. However, if the TOE variations have security-relevant functional differences, each of the TOE models that exhibits either functional or structural differences must be separately tested. Generally speaking, only the difference between each variation of TOE must be separately tested. Other equivalent functionality may be tested on a representative model and not across multiple platforms.

506 If it is determined that a TOE operates the same regardless of the environment, testing may be performed on a single instance for all equivalent configurations. However, if the TOE is determined to provide environment-specific functionality, testing must take place in each environment for which a difference in functionality exists. Similar to the above scenario, only the functionality affected by environment differences must be retested.

507 If a vendor disagrees with the evaluator's assessment of equivalency, the Scheme arbitrates between the two parties whether equivalency exists.

B.2 Evaluator guidance for determining equivalence

B.2.1 Strategy

508 When performing the equivalency analysis, the evaluator should consider each factor independently. A factor may be any number of things at various levels of abstraction, ranging from the processor a device uses, to the underlying operating system and hardware platform a software application relies upon. Examples may be the various chip sets employed by the product, the type of network interface (different device drivers), storage media (solid state drive, spinning disk, EEPROM). It is important to consider how the difference in these factors may influence the TOE's ability to enforce the SFRs. Each analysis of an individual factor will result in one of two outcomes:

- For the particular factor, all variations of the TOE on all supported platforms are equivalent. In this case, testing may be performed on a single model in a single test environment and cover all supported models and environments.
- For the particular factor, a subset of the product has been identified to require separate testing to ensure that it operates identically to all other equivalent TOEs. The analysis would identify the specific combinations of models/testing environments that needed to be tested.

509 Complete CC testing of the product would encompass the totality of each individual analysis performed for each of the identified factors.

B.3 Test presentation/Truth in advertising

510 In addition to determining what to test, the evaluation results and resulting validation report must identify the actual module and testing environment combinations that have been tested. The analysis used to determine the testing subset may be considered proprietary and will only optionally be publicly included.

C. Public Vulnerability Sources

The following sources of public vulnerabilities are sources for the iTC to consider in both formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform key-word searches during the evaluation of a specific TOE.

a. Search Common Vulnerabilities and Exposures: http://cve.mitre.org/cve/

b. Search Core Security Technologies: http://www.coresecurity.com

c. Search eEye Digital Security: http://blog.beyondtrust.com/zd_threat?status=zeroday

d. Search Exploit / Vulnerability Search Engine: www.exploitsearch.net

e. Conduct SecurITeam Exploit Search: www.securiteam.com

f. Search SecurityTracker: www.securitytracker.com

g. Search VUPEN Security, formerly FrSIRT: www.vupen.com

h. Conduct Google search: www.google.com

i. Search McAfee Threat Intelligence http://www.mcafee.com/us/mcafee-labs/threat-intelligence.aspx

j. Search Open Source Vulnerability Database http://osvdb.org/

k. Search Secwatch Advisories & Exploits https://securitynewsportal.com/index.shtml

l. Search Symantec http://www.symantec.com/security_response/

m. Search Tenable Network Security http://nessus.org/plugins/index.php?view=search

n. Tipping Point Zero Day Initiative http://www.zerodayinitiative.com/advisories

o. Search US-CERT http://www.kb.cert.org/vuls/html/search

p. Search Vigil@nce http://vigilance.fr/