Supporting Document
Mandatory Technical Document

Full Drive Encryption: Encryption Engine

February 2019

Version 2.0 + Errata 20190201

CCDB-2019-xxxx

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 *Full Drive Encryption iTC* and is designed to be used to support the evaluations of TOEs against the cPPs identified in section 1.1.

Technical Editor:

FDE iTC

Document history:

V0.7, September 2014 (Initial Release for Public review)

V0.11, October 2014 (Adjudicated comments from Public Review, submitted to CCDB)

V1.0 January 2015 (Incorporated changes due to comments received from CCDB review)

V1.5 September 2015 (Updated to reflect latest revision of cPP)

V2.0 August 2016 (Updated to reflect comments received)

V2.0 + Errata 20190201 February 2019 (Updated to reflect CC Part 3 evaluation findings and FDE Interpretation Team [FIT] rulings)

General Purpose:

The FDE technology type is special due to its physical scope and its limited external interfaces. This leads to some difficulties in evaluating the correctness of the implementation of the TOE’s provided security functions. In the case of the Encryption Engine, it may be difficult to trigger the interface to demonstrate the TSF is properly encrypting the user data. Therefore methods have to be described on how to overcome this challenge (as well as others) in a comparable, transparent and repeatable manner in this document.

Furthermore the main functionality of FDEs is to store user data in encrypted form 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 encrypted form (and not additionally in clear text), but also of definitions of possibly necessary special test tools and their manuals.

Field of special use:

Full Drive Encryption devices, specifically the set of security functional requirements associated with the encryption engine component.

Acknowledgements:

This Supporting Document was developed by the Full Drive Encryption international Technical Community with representatives from industry, Government agencies, Common Criteria Test Laboratories, and members of academia.

Contents

1 Introduction

1.1 Technology Area and Scope of Supporting Document

1 The purpose of the first set of Collaborative Protection Profiles (cPPs) for Full Drive Encryption (FDE): Authorization Acquisition (AA) and Encryption Engine (EE) is to provide requirements for Data-at-Rest protection for a lost device. These cPPs allow FDE solutions based in software and/or hardware to meet the requirements. The form factor for a storage device may vary, but could include: system hard drives/solid state drives in servers, workstations, laptops, mobile devices, tablets, and external media. A hardware solution could be a Self-Encrypting Drive or other hardware-based solutions; the interface (USB, SATA, etc.) used to connect the storage device to the host machine is outside the scope.

2 Full Drive Encryption encrypts all data (with certain exceptions) on the storage device and permits access to the data only after successful authorization to the FDE solution. The exceptions include the necessity to leave a portion of the storage device (the size may vary based on implementation) unencrypted for such things as the Master Boot Record (MBR) or other AA/EE pre-authentication software. These FDE cPPs interpret the term “full drive encryption” to allow FDE solutions to leave a portion of the storage device unencrypted so long as it contains no plaintext user or plaintext authorization data.

3 The FDE cPP - Encryption Engine describes the requirements for the Encryption Engine piece and details the necessary security requirements and evaluation activities for the actual encryption/decryption of the data by the DEK. Each cPP will also have a set of core requirements for management functions, proper handling of cryptographic keys, updates performed in a trusted manner, audit and self-tests.

4 This Supporting Document is mandatory for evaluations of TOEs that claim conformance to the following cPP:

5 a) Collaborative Protection Profile for Full Drive Encryption - Encryption Engine, Version 2.0 + Errata 20190201, February 2019.

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

1.2 Structure of the Document

7 Evaluation Activities can be defined for both Security Functional Requirements and Security Assurance Requirements. These are defined in separate sections of this Supporting Document.

8 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.

9 In general, if all Evaluation Activities (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’. To reach a ‘fail’ verdict when the Evaluation Activities have been successfully completed would require a specific justification from the evaluator as to why the Evaluation Activities were not sufficient for that TOE.

10 Similarly, at the more granular level of Assurance Components, if the Evaluation Activities for an Assurance Component and all of its related SFR Evaluation Activities are successfully completed in an evaluation then it would be expected that the verdict for the Assurance Component is a ‘pass’. To reach a ‘fail’ verdict for the Assurance Component when these Evaluation Activities have been successfully completed would require a specific justification from the evaluator as to why the Evaluation Activities were not sufficient for that TOE.

1.3 Terminology

1.3.1 Glossary

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

12 **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 4).

Term	***Meaning***
Authorization Factor	A value that a user knows, has, or is (e.g. password, token, etc.) submitted to the TOE to establish that the user is in the community authorized to use the hard disk and that is used in the derivation or decryption of the BEV and eventual decryption of the DEK. Note that these values may or may not be used to establish the particular identity of the user.
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
Border Encryption Value	A value passed from the AA to the EE intended to link the key chains of the two components.
Key Sanitization	A method of sanitizing encrypted data by securely overwriting the key that was encrypting the data.
Data Encryption Key (DEK)	A key used to encrypt data-at-rest.

Term	*Meaning*
Full Drive Encryption	Refers to partitions of logical blocks of user accessible data as managed by the host system that indexes and partitions and an operating system that maps authorization to read or write data to blocks in these partitions. For the sake of this Security Program Definition (SPD) and cPP, FDE performs encryption and authorization on one partition, so defined and supported by the OS and file system jointly, under consideration. FDE products encrypt all data (with certain exceptions) on the partition of the storage device and permits access to the data only after successful authorization to the FDE solution. The exceptions include the necessity to leave a portion of the storage device (the size may vary based on implementation) unencrypted for such things as the Master Boot Record (MBR) or other AA/EE pre-authentication software. These FDE cPPs interpret the term "full drive encryption" to allow FDE solutions to leave a portion of the storage device unencrypted so long as it contains no protected data.
Intermediate Key	A key used in a point between the initial user authorization and the DEK.
Host Platform	The local hardware and software the TOE is running on, this does not include any peripheral devices (e.g. USB devices) that may be connected to the local hardware and software.
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.
Key Material	Key material is commonly known as critical security parameter (CSP) data, and also includes authorization data, nonces, and metadata.
Key Release Key (KRK)	A key used to release another key from storage, it is not used for the direct derivation or decryption of another key.
Operating System (OS)	Software which runs at the highest privilege level and can directly control hardware resources.
Non-Volatile Memory	A type of computer memory that will retain information without power.
Powered-Off State	The device has been shut down.

Term	*Meaning*
Protected Data	This refers to all data on the storage device with the exception of a small portion required for the TOE to function correctly. It is all space on the disk a user could write data to and includes the operating system, applications, and user data. Protected data does not include the Master Boot Record or Pre-authentication area of the drive – areas of the drive that are necessarily unencrypted.
Submask	A submask is a bit string that can be generated and stored in a number of ways.
Target of Evaluation	A set of software, firmware and/or hardware possibly accompanied by guidance. [CC1]

1.3.2 Acronyms

Acronym	*Meaning*
AA	Authorization Acquisition
AES	Advanced Encryption Standard
BEV	Border Encryption Value
BIOS	Basic Input Output System
CBC	Cipher Block Chaining
CC	Common Criteria
CCM	Counter with CBC-Message Authentication Code
CEM	Common Evaluation Methodology
CPP	Collaborative Protection Profile
DEK	Data Encryption Key
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EE	Encryption Engine
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standards
FDE	Full Drive Encryption
FFC	Finite Field Cryptography
GCM	Galois Counter Mode
HMAC	Keyed-Hash Message Authentication Code
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
ITSEF	IT Security Evaluation Facility
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
IV	Initialization Vector
KEK	Key Encryption Key
KMD	Key Management Description
KRK	Key Release Key
MBR	Master Boot Record

NIST	National Institute of Standards and Technology
OS	Operating System
RBG	Random Bit Generator
RNG	Random Number Generator
RSA	Rivest Shamir Adleman Algorithm
SAR	Security Assurance Requirement
SED	Self Encrypting Drive
SHA	Secure Hash Algorithm
SFR	Security Functional Requirement
SPD	Security Problem Definition
SPI	Serial Peripheral Interface
ST	Security Target
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSF	TOE Security Functionality
TSS	TOE Summary Specification
USB	Universal Serial Bus
XOR	Exclusive or
XTS	XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing

2 Evaluation Activities for SFRs

13 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 (Evaluation Activities for SARs).

14 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.

15 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.

16 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 Cryptographic Key Management (FCS_CKM)

2.1.1.1 FCS_CKM.1(c) Cryptographic Key Generation (Data Encryption Key)

2.1.1.1.1 TSS

17 The evaluator shall examine the TSS to determine that it describes how the TOE obtains a DEK (either generating the DEK or receiving from the environment).

18 If the TOE generates a DEK, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. If the DEK is generated outside of the TOE, the evaluator checks to ensure that for each platform identified in the TOE the TSS, it describes the interface used by the TOE to invoke this functionality. The evaluator uses the description of the interface between the RBG and the TOE to determine that it requests a key greater than or equal to the required key sizes.

19 If the TOE received the DEK from outside the host platform, then the evaluator shall examine the TSS to determine that the DEK is sent wrapped using the appropriate encryption algorithm.

2.1.1.1.2 Operational Guidance

20 There are no AGD evaluation activities for this SFR.

2.1.1.1.3 KMD

21 If the TOE received the DEK from outside the host platform, then the evaluator shall verify that the KMD describes how the TOE unwraps the DEK.

2.1.1.1.4 Test

22 The evaluator shall perform the following tests:

23 Test 1: The evaluator shall configure the TOE to ensure the functionality of all selections.

2.1.1.2 FCS_CKM.4(a) Cryptographic Key Destruction (Power Management)

2.1.1.2.1 TSS

24 The evaluator shall verify the TSS provides a high level description of how keys stored in volatile memory are destroyed. The valuator to verify that TSS outlines:
- if and when the TSF or the Operational Environment is used to destroy keys from volatile memory;
- if and how memory locations for (temporary) keys are tracked;
- details of the interface used for key erasure when relying on the OE for memory clearing.

2.1.1.2.2 Operational Guidance

25 The evaluator shall check the guidance documentation if the TOE depends on the Operational Environment for memory clearing and how that is achieved.

2.1.1.2.3 KMD

26 The evaluator shall check to ensure the KMD lists each type of key, its origin, possible memory locations in volatile memory.

2.1.1.2.4 Test

27 There are no test evaluation activities for this SFR.

2.1.2 Cryptographic Key Management (FCS_CKM_EXT)

2.1.2.1 FCS_CKM_EXT.4(a) Cryptographic Key and Key Material Destruction (Destruction Timing)

2.1.2.1.1 TSS

28 The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when then should be expected to be destroyed.

2.1.2.1.2 Operational Guidance

29 There are no AGD evaluation activities for this SFR.

2.1.2.1.3 KMD

30 The evaluator shall verify the KMD includes a description of the areas where keys and key material reside and when the keys and key material are no longer needed.

31 The evaluator shall verify the KMD includes a key lifecycle, that includes a description where key material reside, how the key material is used, how it is determined that keys and key material are no longer needed, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM.4(a) for the destruction.

2.1.2.1.4 Test

32 There are no test evaluation activities for this SFR.

2.1.2.2 FCS_CKM_EXT.4(b) Cryptographic Key and Key Material Destruction (Power Management)

2.1.2.2.1 TSS

33 The evaluator shall verify the TSS provides a description of what keys and key material are destroyed when entering any Compliant power saving state.

2.1.2.2.2 Operational Guidance

34 The evaluator shall validate that guidance documentation contains clear warnings and information on conditions in which the TOE may end up in a non-Compliant power saving state indistinguishable from a Compliant power saving state. In that case it must contain mitigation instructions on what to do in such scenarios.

2.1.2.2.3 KMD

35 The evaluator shall verify the KMD includes a description of the areas where keys and key material reside.

36 The evaluator shall verify the KMD includes a key lifecycle that includes a description where key material resides, how the key material is used, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM_EXT.6 for the destruction.

2.1.2.2.4 Test

37 There are no test evaluation activities for this SFR.

2.1.2.3 FCS_CKM_EXT.6 Cryptographic Key Destruction Types

2.1.2.3.1 TSS/KMD (Key Management Description may be used if necessary details describe proprietary information)

38 The evaluator shall examine the TOE's keychain in the TSS/KMD and verify all keys subject to destruction are destroyed according to one of the specified methods.

2.1.2.3.2 Operational Guidance

39 There are no AGD evaluation activities for this SFR.

2.1.2.3.3 Test

40 There are no test evaluation activities for this SFR.

2.1.3 Key Chaining (FCS_KYC_EXT)

2.1.3.1 FCS_KYC_EXT.2 Key Chaining (Recipient)

2.1.3.1.1 TSS

41 There are no TSS evaluation activities for this SFR.

2.1.3.1.2 Operational Guidance

42 There are no AGD evaluation activities for this SFR.

2.1.3.1.3 KMD

43 The evaluator shall examine the KMD to ensure it describes a high level key hierarchy and details of the key chain. The description of the key chain shall be reviewed to ensure it maintains a chain of keys using key wrap or key derivation methods that meet FCS_KDF_EXT.1, FCS_COP.1(d), FCS_COP.1(e), and/or FCS_COP.1(g).

44 The evaluator shall examine the KMD to ensure that it describes how the key chain process functions, such that it does not expose any material that might compromise any key in the chain. (e.g. using a key directly as a compare value against a TPM) This description must include a diagram illustrating the key hierarchy implemented and detail where all keys and keying material is stored or what it is derived from. The evaluator shall examine the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or knowledge of the BEV and the effective strength of the DEK is maintained throughout the Key Chain.

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

2.1.3.1.4 Test

46 There are no test evaluation activities for this SFR.

2.1.4 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)

2.1.4.1 FCS_SNI_EXT.1 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)

2.1.4.1.1 TSS

47 The evaluator shall ensure the TSS describes how salts are generated. The evaluator shall confirm that the salt is generating using an RBG described in FCS_RBG_EXT.1 or by the Operational Environment. If external function is used for this purpose, the TSS should include the specific API that is called with inputs.

48 The evaluator shall ensure the TSS describes how nonces are created uniquely and how IVs and tweaks are handled (based on the AES mode). The evaluator shall confirm that the nonces are unique and the IVs and tweaks meet the stated requirements.

2.1.4.1.2 Operational Guidance

49 There are no AGD evaluation activities for this SFR.

2.1.4.1.3 KMD

50 There are no KMD evaluation activities for this SFR.

2.1.4.1.4 Test

51 There are no test evaluation activities for this SFR.

2.1.5 Validation of Cryptographic Elements (FCS_VAL_EXT)

2.1.5.1 FCS_VAL_EXT.1 Validation

2.1.5.1.1 TSS

52 The evaluator shall examine the TSS to determine which authorization factors support validation.

53 The evaluator shall examine the TSS to review a high-level description if multiple submasks are used within the TOE, how the submasks are validated (e.g., each submask validated before combining, once combined validation takes place).

54 The evaluator shall also examine the TSS to determine that a subset or all of the authorization factors identified in the SFR can be used to exit from a Compliant power saving state.

2.1.5.1.2 Operational Guidance

55 (conditional) If the validation functionality is configurable, 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.

56 (conditional) If the validation functionality is specified by the ST author, the evaluator shall examine the operational guidance to ensure that it states the values that the TOE uses for limits regarding validation attempts.

57 The evaluator shall verify that the guidance documentation states which authorization factors are allowed to exit a compliant power saving state.

2.1.5.1.3 KMD

58 The evaluator shall examine the KMD to verify that it described the method the TOE employs to limit the number of consecutively failed authorization attempts.

59 The evaluator shall examine the vendor’s KMD to ensure it describes how validation is performed. The description of the validation process in the KMD provides detailed information how the TOE validates the BEV.

60 The KMD describes how the process works, such that it does not expose any material that might compromise the submask(s).

2.1.5.1.4 Test

61 The evaluator shall perform the following tests:

62 Test 1: The evaluator shall determine the limit on the average rate of the number of consecutive failed authorization attempts. The evaluator will test the TOE by entering that number of incorrect authorization factors in consecutive attempts to access the protected data. If the limit mechanism includes any “lockout” period, the time period tested should include at least one such period. Then the evaluator will verify that the TOE behaves as described in the TSS.

63 Test 2: The evaluator shall force the TOE to enter a Compliant power saving state, attempt to resume it from this state, and verify that only a valid authorization factor as

defined by the guidance documentation is sufficient to allow the TOE to exit the Compliant power saving state.

2.2 User Data Protection (FDP)

2.2.1 Protection of Data on Disk (FDP_DSK_EXT)

2.2.1.1 FDP_DSK_EXT.1 Protection of Data on Disk

2.2.1.1.1 TSS

64 The evaluator shall examine the TSS to ensure that the description is comprehensive in how the data is written to the disk and the point at which the encryption function is applied. The TSS must make the case that standard methods of accessing the disk drive via the host platforms operating system will pass through these functions.

65 For the cryptographic functions that are provided by the Operational Environment, the evaluator shall check the TSS to ensure it describes, for each platform identified in the ST, the interface(s) used by the TOE to invoke this functionality.

66 The evaluator shall verify the TSS in performing the evaluation activities for this requirement. The evaluator shall ensure the comprehensiveness of the description, confirms how the TOE writes the data to the disk drive, and the point at which it applies the encryption function.

67 The evaluator shall verify that the TSS describes the initialization of the TOE and the activities the TOE performs to ensure that it encrypts all the storage devices entirely when a user or administrator first provisions the TOE. The evaluator shall verify the TSS describes areas of the disk that it does not encrypt (e.g., portions associated with the Master Boot Records (MBRs), boot loaders, partition tables, etc.). If the TOE supports multiple disk encryptions, the evaluator shall examine the administration guidance to ensure the initialization procedure encrypts all storage devices on the platform.

2.2.1.1.2 Operational Guidance

68 The evaluator shall review the AGD guidance to determine that it describes the initial steps needed to enable the FDE function, including any necessary preparatory steps. The guidance shall provide instructions that are sufficient, on all platforms, to ensure that all hard drive devices will be encrypted when encryption is enabled.

2.2.1.1.3 KMD

69 The evaluator shall verify the KMD includes a description of the data encryption engine, its components, and details about its implementation (e.g. for hardware: integrated within the device's main SOC or separate co-processor, for software: initialization of the product, drivers, libraries (if applicable), logical interfaces for encryption/decryption, and areas which are not encrypted (e.g. boot loaders, portions associated with the Master Boot Record (MBRs), partition tables, etc.)). The evaluator shall verify the KMD provides a functional (block) diagram showing the main components (such as memories and processors) and the data path between, for hardware, the device's host interface and the device's persistent media storing the data,

or for software, the initial steps needed to the activities the TOE performs to ensure it encrypts the storage device entirely when a user or administrator first provisions the product. The hardware encryption diagram shall show the location of the data encryption engine within the data path. The evaluator shall validate that the hardware encryption diagram contains enough detail showing the main components within the data path and that it clearly identifies the data encryption engine.

70 The evaluator shall verify the KMD provides sufficient instructions for all platforms to ensure that when the user enables encryption, the product encrypts all hard storage devices. The evaluator shall verify that the KMD describes the data flow from the device's host interface to the device's persistent media storing the data. The evaluator shall verify that the KMD provides information on those conditions in which the data bypasses the data encryption engine (e.g. read-write operations to an unencrypted Master Boot Record area).

71 The evaluator shall verify that the KMD provides a description of the platform's boot initialization, the encryption initialization process, and at what moment the product enables the encryption. The evaluator shall validate that the product does not allow for the transfer of user data before it fully initializes the encryption. The evaluator shall ensure the software developer provides special tools which allow inspection of the encrypted drive either in-band or out-of-band, and may allow provisioning with a known key.

2.2.1.1.4 Test

72 The evaluator shall perform the following tests:

73 Test 1: Write data to random locations, perform required actions and compare:

- Ensure TOE is initialized and, if hardware, encryption engine is ready;
- Provision TOE to encrypt the storage device. For SW Encryption products, or hybrid products use a known key and the developer tools.
- Determine a random character pattern of at least 64 KB;
- Retrieve information on what the device TOE's lowest and highest logical address is for which encryption is enabled.

74 Test 2: Write pattern to storage device in multiple locations:

- For HW Encryption, randomly select several logical address locations within the device's lowest to highest address range and write pattern to those addresses;
- For SW Encryption, write the pattern using multiple files in multiple logical locations.

75 Test 3: Verify data is encrypted:

- For HW Encryption:
  - engage device's functionality for generating a new encryption key, thus performing an erase of the key per FCS_CKM.4(a);
  - 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
- For SW Encryption, using developer tools;
  - Review the encrypted storage device for the plaintext pattern at each location where the file was written and confirm plaintext pattern cannot be found.

- Using the known key, verify that each location where the file was written, the plaintext pattern can be correctly decrypted using the key.
- If available in the developer tools, verify there are no plaintext files present in the encrypted range.

2.3 Security Management (FMT)

2.3.1 Specification of Management Functions (FMT_SMF)

2.3.1.1 FMT_SMF.1 Specification of Management Functions

2.3.1.1.1 TSS

76 If item a) is selected in FMT_SMF.1.1: The evaluator shall ensure the TSS describes how the TOE changes the DEK.

77 If item b) is selected in FMT_SMF.1.1: The evaluator shall ensure the TSS describes how the TOE cryptographically erases the DEK.

78 If item c) is selected in FMT_SMF.1.1: The evaluator shall ensure the TSS describes the process to initiate TOE firmware/software updates.

79 If item d) is selected in FMT_SMF.1.1: If additional management functions are claimed in the ST, the evaluator shall verify that the TSS describes those functions.

2.3.1.1.2 Operational Guidance

80 If item a) is selected in FMT_SMF.1.1: The evaluator shall review the AGD guidance and shall determine that the instructions for changing a DEK exist. The instructions must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate or re-generate the DEK.

81 If item c) is selected in FMT_SMF.1.1: The evaluator shall examine the operational guidance to ensure that it describes how to initiate TOE firmware/software updates.

82 If item d) is selected in FMT_SMF.1.1: Default Authorization Factors: It may be the case that the TOE arrives with default authorization factors in place. If it does, then the selection in item D must be made so that there is a mechanism to change these authorization factors. The operational guidance shall describe the method by which the user changes these factors when they are taking ownership of the device. The TSS shall describe the default authorization factors that exist.

83 Disable Key Recovery: The guidance for disabling this capability shall be described in the AGD documentation.

2.3.1.1.3 KMD

84 If item d) is selected in FMT_SMF.1.1: If the TOE offers the functionality to import an encrypted DEK, the evaluator shall ensure the KMD describes how the TOE imports a wrapped DEK and performs the decryption of the wrapped DEK.

2.3.1.1.4 Test

85 If item a) and/or b) is selected in FMT_SMF.1.1: The evaluator shall verify that the TOE has the functionality to change and cryptographically erase the DEK (effectively removing the ability to retrieve previous user data).

86 If item c) is selected in FMT_SMF.1.1: The evaluator shall verify that the TOE has the functionality to initiate TOE firmware/software updates.

87 If item d) is selected in FMT_SMF.1.1: If additional management functions are claimed, the evaluator shall verify that the additional features function as described.

2.4 Protection of the TSF (FPT)

2.4.1 Key and Key Material Protection (FPT_KYP_EXT)

2.4.1.1 FPT_KYP_EXT.1 Protection of Key and Key Material

2.4.1.1.1 TSS

88 The evaluator shall examine the TSS to verify that it describes the method by which intermediate keys are generated using submask combining.

2.4.1.1.2 Operational Guidance

89 There are no AGD evaluation activities for this SFR.

2.4.1.1.3 KMD

90 The evaluator shall examine the KMD for a description of the methods used to protect keys stored in non-volatile memory.

91 The evaluator shall verify the KMD to ensure it describes the storage location of all keys and the protection of all keys stored in non-volatile memory. The description of the key chain shall be reviewed to ensure the selected method is followed for the storage of wrapped or encrypted keys in non-volatile memory and plaintext keys in non-volatile memory meet one of the criteria for storage.

2.4.1.1.4 Test

92 There are no test evaluation activities for this SFR.

2.4.2 Power Management (FPT_PWR_EXT)

2.4.2.1 FPT_PWR_EXT.1 Power Saving States

2.4.2.1.1 TSS

93 The evaluator shall validate the TSS contains a list of Compliant power saving states.

2.4.2.1.2 Operational Guidance

94 The evaluator shall ensure that guidance documentation contains a list of Compliant power saving states. If additional power saving states are supported, then the evaluator shall validate that the guidance documentation states how the use of non-Compliant power saving states can be avoided.

2.4.2.1.3 KMD

95 There are no KMD evaluation activities for this SFR.

2.4.2.1.4 Test

96 The evaluator shall confirm that for each listed Compliant state all key/key materials are removed from volatile memory by using the test indicated by the selection in FCS_CKM_EXT.6.

2.4.2.2 FPT_PWR_EXT.2 Timing of Power Saving States

2.4.2.2.1 TSS

97 The evaluator shall validate that the TSS contains a list of conditions under which the TOE enters a Compliant power saving state.

2.4.2.2.2 Operational Guidance

98 The evaluator shall check that the guidance contains a list of conditions under which the TOE enters a Compliant power saving state. Additionally, the evaluator shall verify that the guidance documentation provides information on how long it is expected to take for the TOE to fully transition into the Compliant power saving state (e.g. how many seconds for the volatile memory to be completely cleared).

2.4.2.2.3 KMD

99 There are no KMD evaluation activities for this SFR.

2.4.2.2.4 Test

100 The evaluator shall trigger each condition in the list of identified conditions and ensure the TOE ends up in a Compliant power saving state by using the test indicated by the selection in FCS_CKM_EXT.6.

2.4.3 TSF Testing (FPT_TST_EXT)

2.4.3.1 FPT_TST_EXT.1 TSF Testing

2.4.3.1.1 TSS

101 The evaluator shall verify that the TSS describes the known-answer self-tests for cryptographic functions.

102 The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TOE and the method by which the TOE tests those functions. The evaluator shall verify that the TSS includes each of these functions, the method by which the TOE verifies the correct operation of the function. The evaluator shall verify that the TSF data are appropriate for TSF Testing. For example, more than blocks are tested for AES in CBC mode, output of AES in GCM mode is tested without truncation, or 512-bit key is used for testing HMAC-SHA-512.

103 If FCS_RBG_EXT.1 is implemented by the TOE and according to NIST SP 800-90, the evaluator shall verify that the TSS describes health tests that are consistent with section 11.3 of NIST SP 800-90.

104 If any FCS_COP functions are implemented by the TOE, the TSS shall describe the known-answer self-tests for those functions.

105 The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TSF, the method by which those functions are tested. The TSS will describe, for each of these functions, the method by which correct operation of the function/component is verified. The evaluator shall determine that all of the identified functions/components are adequately tested on start-up.

2.4.3.1.2 Operational Guidance

106 There are no AGD evaluation activities for this SFR.

2.4.3.1.3 KMD

107 There are no KMD evaluation activities for this SFR.

2.4.3.1.4 Test

108 There are no test evaluation activities for this SFR.

2.4.4 Trusted Update (FPT_TUD_EXT)

2.4.4.1 FPT_TUD_EXT.1 Trusted Update

2.4.4.1.1 TSS

109 The evaluator shall examine the TSS to ensure that it describes information stating that an authorized source signs TOE updates and will have an associated digital signature. The evaluator shall examine the TSS contains a definition of an authorized source along with a description of how the TOE uses public keys for the update verification mechanism in the Operational Environment. The evaluator ensures the TSS contains details on the protection and maintenance of the TOE update credentials.

110 If the Operational Environment performs the signature verification, then the evaluator shall examine the TSS to ensure it describes, for each platform identified in the ST, the interface(s) used by the TOE to invoke this cryptographic functionality.

2.4.4.1.2 Operational Guidance

111 The evaluator ensures that the operational guidance describes how the TOE obtains vendor updates to the TOE; the processing associated with verifying the digital signature of the updates (as defined in FCS_COP.1(a)); and the actions that take place for successful and unsuccessful cases.

2.4.4.1.3 KMD

112 There are no KMD evaluation activities for this SFR.

2.4.4.1.4 Test

113 The evaluators shall perform the following tests (if the TOE supports multiple signatures, each using a different hash algorithm, then the evaluator performs tests for different combinations of authentic and unauthentic digital signatures and hashes, as well as for digital signature alone):

114 Test 1: The evaluator performs the version verification activity to determine the current version of the TOE. After the update tests described in the following tests, the evaluator performs this activity again to verify that the version correctly corresponds to that of the update.

115 Test 2: The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that an update successfully installs on the TOE. The evaluator shall perform a subset of other evaluation activity tests to demonstrate that the update functions as expected.

3 Evaluation Activities for Optional Requirements

3.1 Cryptographic Support (FCS)

3.1.1 Cryptographic Key Management (FCS_CKM)

3.1.1.1 FCS_CKM.4(e) Cryptographic Key Destruction (Key Cryptographic Erase)

3.1.1.1.1 TSS

116 There is no TSS evaluation activity for this SFR.

3.1.1.1.2 Operational Guidance

117 There are no AGD evaluation activity for this SFR.

3.1.1.1.3 KMD

118 The evaluator shall examine the TOE's keychain in the TSS/KMD and identify each instance a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method.

3.1.1.1.4 Test

119 There is no test evaluation activity for this SFR.

3.2 Protection of the TSF (FPT)

3.2.1 Firmware Access Control (FPT_FAC_EXT)

3.2.1.1 FPT_FAC_EXT.1 Firmware Access Control

3.2.1.1.1 TSS

120 The evaluator shall examine the TSS to ensure that it describes information stating how the Access Control process takes place along with a description of the values that are used.

3.2.1.1.2 Operational Guidance

121 The evaluator ensures that the Operational Guidance describes how the user will be expected to interact with the authorization process.

3.2.1.1.3 KMD

122 There are no KMD evaluation activities for this SFR.

3.2.1.1.4 Test

123 The evaluator shall perform the following test.

124 Test 1: The evaluator shall try installing a firmware upgrade and verify that a prompt is required and the appropriate value is necessary for the update to continue.

3.2.2 Rollback Protection (FPT_RBP_EXT)

3.2.2.1 FPT_RBP_EXT.1 Rollback Protection

3.2.2.1.1 TSS

125 The evaluator shall examine the TSS to ensure that it describes at a high level the process for verifying that security version checking is performed before an upgrade is installed. The evaluator shall verify that a high level description of the types of error codes are provided and when an error would be triggered.

3.2.2.1.2 Operational Guidance

126 The evaluator ensures that a description is provided on how the user should interpret the error codes.

3.2.2.1.3 KMD

127 There are no KMD evaluation activities for this SFR.

3.2.2.1.4 Test

128 The evaluator shall perform the following test:

129 Test 1: The evaluator shall try installing a lower security version number upgrade (either by just modifying the version number or by using an upgrade provided by the vendor) and will verify that the lower version cannot be installed and an error is presented to the user.

4 Evaluation Activities for Selection-Based Requirements

4.1 Cryptographic Support (FCS)

4.1.1 Cryptographic Key Management (FCS_CKM)

4.1.1.1 FCS_CKM.1(a) Cryptographic Key Generation (Asymmetric Keys)

4.1.1.1.1 TSS

130 The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

4.1.1.1.2 Operational Guidance

131 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses specified by the AGD documentation and defined in this cPP.

4.1.1.1.3 KMD

132 If the TOE uses an asymmetric key as part of the key chain, the KMD should detail how the asymmetric key is used as part of the key chain.

4.1.1.1.4 Test

133 The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

134 ***Key Generation for FIPS PUB 186-4 RSA Schemes***

135 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.

136 Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

137 1. Random Primes:

- Provable primes
- Probable primes

138 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

139 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. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. 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 from a known good implementation.

140 ***Key Generation for Elliptic Curve Cryptography (ECC)***

141 *FIPS 186-4 ECC Key Generation Test*

142 For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

143 ***Key Generation for Finite-Field Cryptography (FFC)***

144 The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

145 The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

146 Cryptographic and Field Primes:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

147 Cryptographic Group Generator:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

148 The Key generation specifies 2 ways to generate the private key x:

149 Private Key:

- len(q) bit output of RBG where 1 <=x <= q-1
- len(q) + 64 bit output of RBG, followed by a mod (q-1) operation and +1 operation where 1<= x<=q-1.

150 The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

151 To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

152 For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

for each FFC parameter set and key pair.

4.1.1.2 FCS_CKM.1(b) Cryptographic Key Generation (Symmetric Keys)

4.1.1.2.1 TSS

153 The evaluator shall review the TSS to determine that a symmetric key is supported by the product, that the TSS includes a description of the protection provided by the product for this key. The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE.

4.1.1.2.2 Operational Guidance

154 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key size(s) for all uses specified by the AGD documentation and defined in this cPP.

4.1.1.2.3 KMD

155 If the TOE uses a symmetric key as part of the key chain, the KMD should detail how the symmetric key is used as part of the key chain.

4.1.1.2.4 Test

156 There are no test evaluation activities for this SFR.

4.1.1.3 FCS_CKM.4(b) Cryptographic Key Destruction (TOE-Controlled Hardware)

4.1.1.3.1 TSS + KMD (Key Management Description may be used if necessary details describe proprietary information)

157 The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

158 The evaluator shall check to ensure the TSS lists each type of key that is stored, and identifies the memory type where key material is stored. When listing the type of memory employed, the TSS will list each type of memory selected in the FCS_CKM.4.1 SFR, as well as any memory types that employ a different memory controller or storage algorithm. For example, if a TOE uses NOR flash and NAND flash, both types are to be listed.

159 The evaluator shall examine the TSS to ensure it describes the method that is used by the memory controller to write and read memory from each type of memory listed. The purpose here is to provide a description of how the memory controller works so one can determine exactly how keys are written to memory. The description would include how the data is written to and read from memory (e.g., block level, cell level), mechanisms for copies of the key that could potentially exist (e.g., a copy with parity bits, a copy without parity bits, any mechanisms that are used for redundancy).

160 The evaluator shall examine the TSS to ensure it describes the destruction procedure for each key that has been identified. If different types of memory are used to store the key(s), the evaluator shall check to ensure that the TSS identifies the destruction procedure for each memory type where keys are stored (e.g., key X stored in flash memory is destroyed by overwriting once with zeros, key X' stored in EEPROM is destroyed by a overwrite consisting of a pseudo random pattern – the EEPROM used in the TOE uses a wear-leveling scheme as described).

161 If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

162 The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

163 Upon completion of the TSS examination, the evaluator understands how all the keys (and potential copies) are destroyed.

4.1.1.3.2 Operational Guidance

164 There are a variety of concerns that may prevent or delay key 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 any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

165 For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, it is assumed the drive supports the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

166 Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. It is assumed the operating system and file

system of the OE support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion.

167 It is assumed that if a RAID array is being used, only set-ups that support TRIM are utilized. It is assumed if the drive is connected via PCI-Express, the operating system supports TRIM over that channel. It is assumed the drive is healthy and contains minimal corrupted data and will be end of life before a significant amount of damage to drive health occurs, it is assumed there is a risk small amounts of potentially recoverable data may remain in damaged areas of the drive.

168 Finally, it is assumed the keys are not stored using a method that would be inaccessible to TRIM, such as being contained in a file less than 982 bytes which would be completely contained in the master file table.

169 For destruction on wear-leveled memory, if a time period is required before is processed destruction the ST author shall provide an estimated range.

4.1.1.3.3 Test

170 For these tests the evaluator shall utilize appropriate development environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

171 For destruction on wear-leveled memory, if a time period is required before is evaluator shall wait that amount of time after clearing the key in tests 2 and 3.

172 Test 1: Applied to each key 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). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:

1. Record the value of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Cause the TOE to stop the execution but not exit.
5. Cause the TOE to dump the entire memory of the TOE into a binary file.
6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.
7. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.

173 Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

174 Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a miniscule 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 found the test fails.

175 Test 2: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:

1. Record the value of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Search the non-volatile memory the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.
5. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for test 1 above), and if a fragment is found in the repeated test then the test fails.

176 Test 3: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:

1. Record the storage location of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Read the storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

177 The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

4.1.1.4 FCS_CKM.4(c) Cryptographic Key Destruction (General Hardware)

4.1.1.4.1 TSS + KMD (Key Management Description may be used if necessary details describe proprietary information)

178 The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

179 The evaluator shall check to ensure the TSS lists each type of key that is stored, and identifies the memory type (volatile or non-volatile) where key material is stored.

180 The TSS identifies and describes the interface(s) that is 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.

181 If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

182 The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

4.1.1.4.2 Operational Guidance

183 There are a variety of concerns that may prevent or delay key 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 any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

184 For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, it is assumed the drive supports the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

185 Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. It is assumed the operating system and file system of the OE support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion.

186 It is assumed that if a RAID array is being used, only set-ups that support TRIM are utilized. It is assumed if the drive is connected via PCI-Express, the operating system supports TRIM over that channel. It is assumed the drive is healthy and contains minimal corrupted data and will be end of life before a significant amount of damage to drive health occurs, it is assumed there is a risk small amounts of potentially recoverable data may remain in damaged areas of the drive.

187 Finally, it is assumed the keys are not stored using a method that would be inaccessible to TRIM, such as being contained in a file less than 982 bytes which would be completely contained in the master file table.

4.1.1.4.3 Test

188 For these tests the evaluator shall utilize appropriate development environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

189 Test 1: Applied to each key 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). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Cause the TOE to stop the execution but not exit.

5. Cause the TOE to dump the entire memory of the TOE into a binary file.

6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

7. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.

190 Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

191 Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a miniscule 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 found the test fails.

192 Test 2: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Search the non-volatile memory the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.

5. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for test 1 above), and if a fragment is found in the repeated test then the test fails.

193 Test 3: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:

1. Record the storage location of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Search the storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

194 The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

4.1.1.5 FCS_CKM.4(d) Cryptographic Key Destruction (Software TOE, 3rd Party Storage)

4.1.1.5.1 TSS + KMD (Key Management Description may be used if necessary details describe proprietary information)

195 The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

196 The evaluator shall check to ensure the TSS lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

197 The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) and description in the TSS.

198 The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement. If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

4.1.1.5.2 Operational Guidance

199 There are a variety of concerns that may prevent or delay key 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 any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

200 For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, it is assumed the drive supports the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

201 Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. It is assumed the operating system and file

system of the OE support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion.

202 It is assumed that if a RAID array is being used, only set-ups that support TRIM are utilized. It is assumed if the drive is connected via PCI-Express, the operating system supports TRIM over that channel. It is assumed the drive is healthy and contains minimal corrupted data and will be end of life before a significant amount of damage to drive health occurs, it is assumed there is a risk small amounts of potentially recoverable data may remain in damaged areas of the drive.

203 Finally, it is assumed the keys are not stored using a method that would be inaccessible to TRIM, such as being contained in a file less than 982 bytes which would be completely contained in the master file table.

4.1.1.5.3 Test

204 Test 1: Applied to each key 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). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:

1. Record the value of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Cause the TOE to stop the execution but not exit.
5. Cause the TOE to dump the entire memory of the TOE into a binary file.
6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.
7. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.

205 Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

206 Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a miniscule 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 found the test fails.

207 *The following tests apply only to selection a), since the TOE in this instance has more visibility into what is happening within the underlying platform (e.g., a logical view of the media). In selection b), the TOE has no visibility into the inner workings and completely relies on the underlying platform, so there is no reason to test the TOE beyond test 1.*

208 *For selection a), the following tests are used to determine the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.*

209 Test 2: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):

1. Record the value of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.
5. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for Use Case 1 test 1 above), and if a fragment is found in the repeated test then the test fails.

210 Test 3: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:

1. Record the logical storage location of the key in the TOE subject to clearing.
2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
3. Cause the TOE to clear the key.
4. Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

211 The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

4.1.2 Cryptographic Operation (FCS_COP)

4.1.2.1 FCS_COP.1(a) Cryptographic Operation (Signature Verification)

212 This requirement is used to verify digital signatures attached to updates from the TOE manufacturer before installing those updates on the TOE. Because this component is to be used in the update function, additional Evaluation Activities to those listed below are covered in other evaluation activities sections in this document. The following activities deal only with the implementation for the digital signature algorithm; the evaluator performs the testing appropriate for the algorithm(s) selected in the component.

213 Hash functions and/or random number generation required by these algorithms must be specified in the ST; therefore the Evaluation Activities associated with those functions are contained in the associated Cryptographic Hashing and Random Bit Generation sections. Additionally, the only function required by the TOE is the verification of digital signatures. If the TOE generates digital signatures to support the implementation of any functionality required by this cPP, then the applicable

evaluation and validation scheme must be consulted to determine the required evaluation activities.

4.1.2.1.1 TSS

214 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 on to the device; and any processing that is performed that is not part of the digital signature algorithm (for instance, checking of certificate revocation lists).

4.1.2.1.2 Operational Guidance

215 There are no AGD evaluation activities for this SFR.

4.1.2.1.3 KMD

216 There are no KMD evaluation activities for this SFR.

4.1.2.1.4 Test

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

218 It should be noted that for the schemes given below, there are no key generation/domain parameter generation testing requirements. This is because it is not anticipated that this functionality would be needed in the end device, since the functionality is limited to checking digital signatures in delivered updates. This means that the domain parameters should have already been generated and encapsulated in the hard drive firmware or on-board non-volatile storage. If key generation/domain parameter generation is required, the evaluation and validation scheme must be consulted to ensure the correct specification of the required evaluation activities and any additional components.

219 The following tests are conditional based upon the selections made within the SFR.

220 The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

221 ECDSA Algorithm Tests

222 **ECDSA FIPS 186-4 Signature Verification Test**
For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

223 RSA Signature Algorithm Tests

224 **Signature Verification Test**

225 The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party’s authentic and unauthentic signatures. The evaluator

shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

226 The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

4.1.2.2 FCS_COP.1(b) Cryptographic Operation (Hash Algorithm)

4.1.2.2.1 TSS

227 The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

4.1.2.2.2 Operational Guidance

228 The evaluator checks the operational guidance documents to determine that any system configuration necessary to enable required hash size functionality is provided.

4.1.2.2.3 KMD

229 There are no KMD evaluation activities for this SFR.

4.1.2.2.4 Test

230 The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only 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 mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented test mode.

231 The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this cPP.

232 Short Messages Test Bit-oriented Mode

233 The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly 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.

234 Short Messages Test Byte-oriented Mode

235 The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly 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.

236 Selected Long Messages Test Bit-oriented Mode

237 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 +

99*i, where 1 ≤ i ≤ m. For SHA-384 and SHA-512, the length of the i-th message is 1024 + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly 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.

238 Selected Long Messages Test Byte-oriented Mode

239 The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. For SHA-384 and SHA-512, the length of the i-th message is 1024 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly 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.

240 Pseudorandomly Generated Messages Test

241 This test is for byte-oriented implementations only. 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 the NIST Secure Hash Algorithm Validation System (SHAVS) (https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/shs/SHAVS.pdf). The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

4.1.2.3 FCS_COP.1(c) Cryptographic Operation (Keyed Hash Algorithm)

4.1.2.3.1 TSS

242 If HMAC was selected:

243 The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

244 If CMAC was selected:

245 The evaluator shall examine the TSS to ensure that it specifies the following values used by the CMAC function: key length, block cipher used, block size (of the cipher), and output MAC length used.

4.1.2.3.2 Operational Guidance

246 There are no AGD evaluation activities for this SFR.

4.1.2.3.3 KMD

247 There are no KMD evaluation activities for this SFR.

4.1.2.3.4 Test

248 If HMAC was selected:

249 For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be

compared to the result of generating HMAC tags with the same key using a known good implementation.

250 If CMAC was selected:

251 For each of the supported parameter sets, the evaluator shall compose at least 15 sets of test data. Each set shall consist of a key and message data. The test data shall include messages of different lengths, some with partial blocks as the last block and some with full blocks as the last block. The test data keys shall include cases for which subkey K1 is generated both with and without using the irreducible polynomial R_b, as well as cases for which subkey K2 is generated from K1 both with and without using the irreducible polynomial R_b. (The subkey generation and polynomial R_b are as defined in SP800-38E.) The evaluator shall have the TSF generate CMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating CMAC tags with the same key using a known good implementation.

4.1.2.4 FCS_COP.1(d) Cryptographic Operation (Key Wrapping)

4.1.2.4.1 TSS

252 The evaluator shall verify the TSS includes a description of the key wrap function(s) and shall verify the key wrap uses an approved key wrap algorithm according to the appropriate specification.

4.1.2.4.2 Operational Guidance

253 There are no AGD evaluation activities for this SFR.

4.1.2.4.3 KMD

254 The evaluator shall review the KMD to ensure that all keys are wrapped using the approved method and a description of when the key wrapping occurs.

4.1.2.4.4 Test

255 There are no test evaluation activities for this SFR.

4.1.2.5 FCS_COP.1(e) Cryptographic Operation (Key Transport)

4.1.2.5.1 TSS

256 The evaluator shall verify the TSS provides a high level description of the RSA scheme and the cryptographic key size that is being used, and that the asymmetric algorithm being used for key transport is RSA. If more than one scheme/key size are allowed, then the evaluator shall make sure and test all combinations of scheme and key size. There may be more than one key size to specify – an RSA modulus size (and/or encryption exponent size), an AES key size, hash sizes, MAC key/MAC tag size.

257 If the KTS-OAEP scheme was selected, the evaluator shall verify that the TSS identifies the hash function, the mask generating function, the random bit generator, the encryption primitive and decryption primitive.

258 If the KTS-KEM-KWS scheme was selected, the evaluator shall verify that the TSS identifies the key derivation method, the AES-based key wrapping method, the secret value encapsulation technique, and the random number generator.

4.1.2.5.2 Operational Guidance

259 There are no AGD evaluation activities for this SFR.

4.1.2.5.3 KMD

260 There are no KMD evaluation activities for this SFR.

4.1.2.5.4 Test

261 For each supported key transport schema, the evaluator shall initiate at least 25 sessions that require key transport with an independently developed remote instance of a key transport entity, using known RSA key-pairs. The evaluator shall observe traffic passed from the sender-side and to the receiver-side of the TOE, and shall perform the following tests, specific to which key transport scheme was employed.

262 If the KTS-OAEP scheme was selected, the evaluator shall perform the following tests:

1. The evaluator shall inspect each cipher text, C, produced by the RSA-OAEP encryption operation of the TOE and make sure it is the correct length, either 256 or 384 bytes depending on RSA key size. The evaluator shall also feed into the TOE's RSA-OEAP decryption operation some cipher texts that are the wrong length and verify that the erroneous input is detected and that the decryption operation exits with an error code.

2. The evaluator shall convert each cipher text, C, produced by the RSA-OAEP encryption operation of the TOE to the correct cipher text integer, c, and use the decryption primitive to compute em = RSADP((n,d),c) and convert em to the encoded message EM. The evaluator shall then check that the first byte of EM is 0x00. The evaluator shall also feed into the TOE's RSA-OEAP decryption operation some cipher texts where the first byte of EM was set to a value other than 0x00, and verify that the erroneous input is detected and that the decryption operation exits with an error code.

3. The evaluator shall decrypt each cipher text, C, produced by the RSA-OAEP encryption operation of the TOE using RSADP, and perform the OAEP decoding operation (described in NIST SP 800-56B section 7.2.2.4) to recover HA' || X. For each HA', the evaluator shall take the corresponding A and the specified hash algorithm and verify that HA' = Hash(A). The evaluator shall also force the TOE to perform some RSA-OAEP decryptions where the A value is passed incorrectly, and the evaluator shall verify that an error is detected.

4. The evaluator shall check the format of the 'X' string recovered in OAEP.Test.3 to ensure that the format is of the form PS || 01 || K, where PS consists of zero or more consecutive 0x00 bytes and K is the transported keying material. The evaluator shall also feed into the TOE's RSA-OEAP decryption operation some cipher texts for which the resulting 'X' strings do not have the correct format (i.e., the leftmost non-zero byte is not 0x01). These incorrectly formatted 'X' variables shall be detected by the RSA-OEAP decrypt function.

5. The evaluator shall trigger all detectable decryption errors and validate that the returned error codes are the same and that no information is given back to the sender on which type of error occurred. The evaluator shall also validate that no intermediate results from the TOE's receiver-side operations are revealed to the sender.

263 If the KTS-KEM-KWS scheme was selected, the evaluator shall perform the following tests:

1. The evaluator shall inspect each cipher text, C, produced by RSA-KEM-KWS encryption operation of the TOE and make sure the length (in bytes) of the cipher text, cLen, is greater than nLen (the length, in bytes, of the modulus of the RSA public key) and that cLen - nLen is consistent with the byte lengths supported by the key wrapping algorithm. The evaluator shall feed into the RSA-KEM-KWS decryption operation a cipher text of unsupported length and verify that an error is detected and that the decryption process stops.

2. The evaluator shall separate the cipher text, C, produced by the sender-side of the TOE into its C0 and C1 components and use the RSA decryption primitive to recover the secret value, Z, from C0. The evaluator shall check that the unsigned integer represented by Z is greater than 1 and less than n-1, where n is the modulus of the RSA public key. The evaluator shall construct examples where the cipher text is created with a secret value Z = 1 and make sure the RSA-KEM-KWS decryption process detects the error. Similarly, the evaluator shall construct examples where the cipher text is created with a secret value Z = n – 1 and make sure the RSA-KEM-KWS decryption process detects the error.

3. The evaluator shall attempt to successfully recover the secret value Z, derive the key wrapping key, KWK, and unwrap the KWA-cipher text following the RSA-KEM-KWS decryption process given in NISP SP 800-56B section 7.2.3.4. If the key-wrapping algorithm is AES-CCM, the evaluator shall verify that the value of any (unwrapped) associated data, A, that was passed with the wrapped keying material is correct The evaluator shall feed into the TOE's RSA-KEM-KWS decryption operation examples of incorrect cipher text and verify that a decryption error is detected. If the key-wrapping algorithm is AES-CCM, the evaluator shall attempt at least one decryption where the wrong value of A is given to the RSA-KEM-KWS decryption operation and verify that a decryption error is detected. Similarly, if the key-wrapping algorithm is AES-CCM, the evaluator shall attempt at least one decryption where the wrong nonce is given to the RSA-KEM-KWS decryption operation and verify that a decryption error is detected.

4. The evaluator shall trigger all detectable decryption errors and validate that the resulting error codes are the same and that no information is given back to the sender on which type of error occurred. The evaluator shall also validate that no intermediate results from the TOE's receiver-side operations (in particular, no Z values) are revealed to the sender.

4.1.2.6 FCS_COP.1(f) Cryptographic Operation (AES Data Encryption/Decryption)

4.1.2.6.1 TSS

264 The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for encryption.

4.1.2.6.2 Operational Guidance

265 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.

4.1.2.6.3 KMD

266 There are no KMD evaluation activities for this SFR.

4.1.2.6.4 Test

267 The following tests are conditional based upon the selections made in the SFR.

268 **AES-CBC Tests**

269 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.

270 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). Known answer values tailored to exercise the AES-CBC implementation can be obtained using NIST's CAVS Algorithm Validation Tool or from NIST's ACPV service for automated algorithm tests (acvp.nist.gov), when available. 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.

271 **AES-CBC Known Answer Tests**

272 KAT-1 (GFSBox):

273 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.

274 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.

275 KAT-2 (KeySBox):

276 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.

277 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.

278 KAT-3 (Variable Key):

279 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.

280 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.

281 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.

282 KAT-4 (Variable Text):

283 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.

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

285 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.

286 **AES-CBC Multi-Block Message Test**

287 The evaluator shall test the encrypt functionality by encrypting nine i-block messages for each selected key size, for $2 \leq i \leq 10$. 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.

288 The evaluator shall test the decrypt functionality by decrypting nine i-block messages for each selected key size, for $2 \leq i \leq 10$. 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.

289 **AES-CBC Monte Carlo Tests**

290 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.

291 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.

292
```
# Input: PT, IV, Key
Key[0] = Key
IV[0] = IV
PT[0] = PT

for i = 1 to 100  {
```

```
        Output Key[i], IV[i], PT[0]
        for j = 1 to 1000 {
        if j == 1 {
                CT[1] = AES-CBC-Encrypt(Key[i], IV[i], PT[1])
                PT[2] = IV[i]
        } else {
                CT[j] = AES-CBC-Encrypt(Key[i], PT[j])
                PT[j+1] = CT[j-1]
        }
  }
     Output CT[1000]

     If KeySize == 128 { Key[i+1] = Key[i] xor CT[1000] }
     If KeySize == 256 { Key[i+1] = Key[i] xor ((CT[999] << 128) | CT[1000]) }

     IV[i+1] = CT[1000]
     PT[0] = CT[999]
}
```

293 The ciphertext computed in the 1000th iteration (CT[1000]) 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.

294 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.

295 **AES-GCM Test**

296 The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

297 The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

298 The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

299 The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

300 **XTS-AES Test**

301 The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

302 using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

303 The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

304 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.

4.1.2.7 FCS_COP.1(g) Cryptographic Operation (Key Encryption)

4.1.2.7.1 TSS

305 The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for the key encryption.

4.1.2.7.2 Operational Guidance

306 If multiple key 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.

4.1.2.7.3 KMD

307 The evaluator shall examine the vendor's KMD to verify that it includes a description of how key encryption will be used as part of the key chain.

4.1.2.7.4 Test

308 The AES test should be followed in FCS_COP.1(f) Cryptographic Operation (AES Data Encryption/Decryption)

4.1.3 Cryptographic Key Derivation (FCS_KDF_EXT)

4.1.3.1 FCS_KDF_EXT.1 Cryptographic Key Derivation

4.1.3.1.1 TSS

309 The evaluator shall verify the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 and SP 800-132.

4.1.3.1.2 Operational Guidance

310 There are no AGD evaluation activities for this SFR.

4.1.3.1.3 KMD

311 The evaluator shall examine the vendor's KMD to ensure that all keys used are derived using an approved method and a description of how and when the keys are derived.

4.1.3.1.4 Test

312 There are no test evaluation activities for this SFR.

4.1.4 Random Bit Generation (FCS_RBG_EXT)

4.1.4.1 FCS_RBG_EXT.1 Random Bit Generation

4.1.4.1.1 TSS

313 For any RBG services provided by a third party, the evaluator shall ensure the TSS includes a statement about the expected amount of entropy received from such a source, and a full description of the processing of the output of the third-party source. The evaluator shall verify that this statement is consistent with the selection made in FCS_RBG_EXT.1.2 for the seeding of the DRBG. If the ST specifies more than one DRBG, the evaluator shall examine the TSS to verify that it identifies the usage of each DRBG mechanism.

4.1.4.1.2 Operational Guidance

314 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected DRBG mechanism(s), if necessary, and provides information regarding how to instantiate/call the DRBG for RBG services needed in this cPP.

4.1.4.1.3 KMD

315 There are no KMD evaluation activities for this SFR.

4.1.4.1.4 Test

316 The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable by the TOE, the evaluator shall perform 15 trials for each configuration. The evaluator shall verify that the instructions in the operational guidance for configuration of the RNG are valid.

317 If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization

string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. “generate one block of random bits” means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

318 If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

319 The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

320 Entropy input: the length of the entropy input value must equal the seed length.

321 Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

322 Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

323 Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

4.1.5 Submask Combining (FCS_SMC_EXT)

4.1.5.1 FCS_SMC_EXT.1 Submask Combining

4.1.5.1.1 TSS

324 If the submasks produced from the authorization factors are XORed together to form the BEV or intermediate key, the TSS section shall identify how this is performed (e.g., if there are ordering requirements, checks performed, etc.). The evaluator shall also confirm that the TSS describes how the length of the output produced is at least the same as that of the BEV.

4.1.5.1.2 Operational Guidance

325 There are no AGD evaluation activities for this SFR.

4.1.5.1.3 KMD

326 The evaluator shall review the KMD to ensure that an approved combination is used and does not result in the weakening or exposure of key material.

4.1.5.1.4 Test

327 The evaluator shall perform the following test:

328 Test 1 (conditional): If there is more than one authorization factor, ensure that failure to supply a required authorization factor does not result in access to the encrypted data.

4.2 Protection of the TSF (FPT)

4.2.1 Firmware Update Authentication (FPT_FUA_EXT)

4.2.1.1 FPT_FUA_EXT.1 Firmware Update Authentication

4.2.1.1.1 TSS

329 The evaluator shall examine the TSS to ensure that it describes how the TOE uses the RTU, what type of key or hash value, and where the value is stored on the RTU. The evaluator shall also verify that the TSS contains a description (storage location) of where the original firmware exists.

4.2.1.1.2 Operational Guidance

330 There are no AGD evaluation activities for this SFR.

4.2.1.1.3 KMD

331 There are no KMD evaluation activities for this SFR.

4.2.1.1.4 Test

332 There are no test evaluation activities for this SFR.

5 Evaluation Activities for SARs

333 The sections below specify Evaluation Activities for the Security Assurance Requirements included in the related cPPs (see section 1.1 above). The Evaluation Activities are an interpretation of the more general CEM assurance requirements as they apply to the specific technology area of the TOE.

334 Note that in order to meet the claimed SARs, all CEM work units must be performed by the evaluators. The Evaluation Activities discussed in sections 2, 3, and 4 as well as the refinements and Evaluation Activities in this section all serve to clarify and supplement the SARs; they do not exempt the SARs from evaluation.

5.1 ASE: Security Target Evaluation

335 An evaluation activity is defined here for evaluation of Exact Conformance claims against a cPP in a Security Target. Other aspects of ASE remain as defined in the CEM.

5.1.1 Conformance Claims (ASE_CCL.1)

336 The table below indicates the actions to be taken for particular ASE_CCL.1 elements in order to determine exact conformance with a cPP.

Table 1: ASE_CCL.1 Exact Conformance Actions

ASE_CCL.1 element	***Evaluator Action***
ASE_CCL.1.8C	The evaluator shall check that the statements of security problem definition in the PP and ST are identical.
ASE_CCL.1.9C	The evaluator shall check that the statements of security objectives in the PP and ST are identical.
ASE_CCL.1.10C	The evaluator shall check that the statements of security requirements in the ST include all the mandatory SFRs in the cPP, and all of the selection-based SFRs that are entailed by selections made in other SFRs (including any SFR iterations added in the ST). The evaluator shall check that if any other SFRs are present in the ST (apart from iterations of SFRs in the cPP) then these are taken only from the list of optional SFRs specified in the cPP (the cPP will not *necessarily* include optional SFRs, but may do so). If optional SFRs from the cPP are included in the ST then the evaluator shall check that any selection-based SFRs entailed by the optional SFRs adopted are also included in the ST.

5.2 Development (ADV)

5.2.1 Basic Functional Specification (ADV_FSP.1)

337 The EAs for this assurance component focus on understanding the interfaces (e.g., application programing 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 sections 2, 3, and 4, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

338 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.

339 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.

340 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.

Table 2: Mapping of ADV_FSP.1 CEM Work Units to Evaluation Activities

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.	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.	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.	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.	5.2.1.3 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 a complete 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 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.

5.2.1.1 Evaluation Activity

341 *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.*

342 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.

343 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

344 *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

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

346 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 (*Evaluation Activities for SFRs*) as well as any applicable EAs in Sections 3 and 4, including the EAs associated with testing of the interfaces.

347 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.

348 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 Guidance Documents (AGD)

349 It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the Evaluation Activities in this section are described under the traditionally separate AGD families, the mapping between real TOE documents and 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)

350 Specific requirements and checks on the user guidance documentation are identified (where relevant) in the individual Evaluation Activities for each SFR, and for some other SARs (e.g. ALC_CMC.1).

351 *Evaluation Activity:*

352 The evaluator shall check the requirements below are met by the operational guidance. It should be noted that operational guidance may take the form of an "integrator's guide", where the TOE developer provides a description of the interface (e.g., commands that the Host Platform may invoke to configure a SED).

353 Operational guidance documentation shall be 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.

354 Operational guidance must be provided for every Operational Environment that the TOE supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target. This may be contained all in one document.

355 The contents of the operational guidance will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in sections 2, 3, and 4 above.

356 In addition to SFR-related Evaluation Activities, the following information is also required.

- The operational guidance shall contain 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.

- The operational guidance shall describe how to configure the IT environments that are supported to shut down after an administratively defined period of inactivity.

- The operational guidance shall identify system "sleeping" states for all supported operating environments and for each environment, provide administrative guidance on how to disable the sleep state. As stated above, the TOE developer may be providing an integrator's guide and "power states" may be an abstraction that SEDs provide at various levels – e.g., may simply provide a command that the Host Platform issues to manage the state of the device, and the Host Platform is responsible for providing a more sophisticated power management scheme.

- The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The operational guidance shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

5.3.2 Preparative Procedures (AGD_PRE.1)

357 As for the operational guidance, specific requirements and checks on the preparative procedures are identified (where relevant) in the individual Evaluation Activities for each SFR.

358 *Evaluation Activity:*

359 The evaluator shall check the requirements below are met by the preparative procedures.

360 The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in sections 2, 3, and 4 above.

361 Preparative procedures shall be 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.

362 The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in sections 2, 3, and 4 above.

363 In addition to SFR-related Evaluation Activities, the following information is also required.

364 Preparative procedures must 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). 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 itself).

365 Preparative procedures must be provided for every Operational Environment that the TOE supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target. . This may be contained all in one document.

366 The preparative procedures must include

- instructions to successfully install the TSF in each Operational Environment; and
- instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
- instructions to provide a protected administrative capability.

5.4 Life-cycle Support (ALC)

5.4.1 Labelling of the TOE (ALC_CMC.1)

367 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)

368 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 Tests (ATE)

5.5.1 Independent Testing – Conformance (ATE_IND.1)

369 Testing is performed to confirm the functionality described in the TSS as well as the operational guidance documentation. The focus of the testing is to confirm that the requirements specified in the SFRs are being met.

370 The evaluator should consult Appendix B, FDE Equivalency Considerations when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

371 The SFR-related Evaluation Activities in the SD identify the specific testing activities necessary to verify compliance with the SFRs. The tests identified in these other Evaluation Activities constitute a sufficient set of tests for the purposes of meeting ATE_IND.1.2E. It is important to note that while the Evaluation Activities identify the testing that is necessary to be performed, the evaluator is responsible for ensuring that the interfaces are adequately tested for the security functionality specified for each SFR.

372 *Evaluation Activity:*

373 The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

374 *Evaluation Activity:*

375 The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

376 *Evaluation Activity:*

377 The evaluator shall prepare a test plan that covers all of the testing actions for ATE_IND.1 in the CEM and in the SFR-related Evaluation Activities. While it is not necessary to have one test case per test listed in an Evaluation Activity, the evaluator must show in the test plan that each applicable testing requirement in the SFR-related Evaluation Activities is covered.

378 The test plan identifies the platforms to be tested, and for any platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

379 The test plan describes the composition and configuration of each platform to be tested, and any setup actions that are necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of any cryptographic engine to be used (e.g. for cryptographic protocols being evaluated).

380 The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives, and the expected results.

381 The test report (which could just be an updated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure, so that a fix was then installed and then a successful re-run of the test was carried out, then the report would show a “fail” result followed by a “pass” result (and the supporting details), and not just the “pass” result[1].

5.6 Vulnerability Assessment (AVA)

5.6.1 Vulnerability Survey (AVA_VAN.1)

382 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

[1] It is not necessary to capture failures that were due to errors on the part of the tester or test environment. The intention here is to make absolutely clear when a planned test resulted in a change being required to the originally specified test configuration in the test plan, to the evaluated configuration identified in the ST and operational guidance, or to the TOE itself.

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.

383 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.

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

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 A.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 A.1, and documentation as specified in Appendix A, Section A.3.
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 A.2.
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:	The CEM work unit is captured in Appendix A, Section A.3; there are no substantive differences.

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 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 A.3 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 A.3.
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 A.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 A.3.
AVA_VAN.1-11 The evaluator ***shall report*** in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each:	Replace the CEM work unit with the reporting called for in Appendix A, Section A.3.

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

384 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):

385 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.

386 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.

387 In addition to the activities specified by the CEM in accordance with Table 3 above, the evaluator shall perform the following activities.

5.6.1.2 Evaluation Activity

388 The evaluator formulates hypotheses in accordance with process defined in Appendix A.1. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

6 Required Supplementary Information

389 This Supporting Document refers in various places to the possibility that '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. 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.

390 The FDE cPP for Encryption Engine requires a key management description and an entropy analysis if the TOE is providing the RNG. The EAs the evaluator is to perform with those documents are captured under the appropriate SFRs in Section 2.

7 References

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

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

[CC3] Common Criteria for Information Technology Security Evaluation,
Part 3: Security Assurance Components,
CCMB-2017-09-004, Version 3.1 Revision 5, April 2017

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

[FDE–EE] collaborative Protection Profile for Full Disk Encryption – Encryption Engine, Version 2.0 + Errata 20190201, 1 February 2019

Appendixes

A. Vulnerability Analysis

A.1 Sources of Vulnerability Information

391 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), 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

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

a. Search Common Vulnerabilities and Exposures: http://cve.mitre.org/cve/
b. Search the National Vulnerability Database: https://nvd.nist.gov/
c. Search US-CERT http://www.kb.cert.org/vuls/html/search

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

- General (for all)
  - Product name
  - underlying components (e.g., OS, software libraries (crypto libraries), chipsets)
  - drive encryption, disk encryption
  - key destruction/sanitization
- EE:

        - Underlying components (e.g., chipsets, firmware)

    - For SEDs (for EE):

        - Self Encrypting Drive (SED), OPAL

    - For Software FDE (AA or EE):

        - Key caching

394 In order to successfully complete this activity, the evaluator will use the developer provided list of all of 3rd party library information that is used as part of their product, along with the version and any other identifying information (this is required in the cPP as part of the ASE_TSS.1.1C requirement). This applies to hardware (including chipsets, etc.) that a vendor utilizes as part of their TOE. This TOE-unique information will be used in the search terms the evaluator uses in addition to those listed above.

395 The evaluator will also consider the requirements that are chosen and the appropriate guidance that is tied to each requirement.

396 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.

397 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).

A.1.2 Type 2 Hypotheses—iTC-Sourced

398 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:

399 *General:*

400 EE:

- Software FDE:

    - During the software encryption installation process, it is possible that that encryption is interrupted (e.g., power is removed, etc.). The evaluator should verify that when the software encryption resumes and completes, that all of the user data is encrypted.

401 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.

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

402 The iTC has leveraged the expertise of the developers and the evaluation labs to diligently develop the appropriate search terms and vulnerability databases. They have also thoughtfully considered the iTC-sourced hypotheses the evaluators should use based upon the applicable use case and the threats to be mitigated by the SFRs. Therefore, it is the intent of the iTC, for the evaluation to focus all effort on the Type 1 and Type 2 Hypotheses and has decided that Type 3 Hypotheses are not necessary.

403 However, if the evaluators discover a Type 3 potential flaw that they believe should be considered, they should work with their Certification Body to determine the feasibility of pursuing the hypothesis. The Certification Body may determine whether the potential flaw hypotheses is worth submitting to the iTC for consideration as Type 2 hypotheses in future drafts of the cPP/SD.

A.1.4 Type 4 Hypotheses—Tool-Generated

404 The iTC has called out several tools that should be used during the Type 2 hypotheses process. Therefore, the use of any tools is covered within the Type 2 construct and the iTC does not see any additional tools that are necessary. The use case for Version 2 of this cPP is rather straightforward – the device is found in a powered down state and has not been subjected to revisit/evil maid attacks. Since that is the use case, the iTC has also assumed there is a trusted channel between the AA and EE. Since the use case is so narrow, and is not a typical model for penetration or fuzzing testing, the normal types of testing do not apply. Therefore, the relevant types of tools are referenced in Type 2.

A.2 Process for Evaluator Vulnerability Analysis

405 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.

406 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:

- the source documents used in formulating the hypothesis, and why it represents a potential compromise against a specific TOE function;
- an argument why the flaw hypothesis could not be proven or disproved by the evidence provided so far; and
- the type of information required to investigate the flaw hypothesis further.

407 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).

408 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 A.3 below.

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

410 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.

411 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).

412 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.

413 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.

414 Any testing performed by the evaluator shall be documented in the test report as outlined in Section A.3 below.

415 As indicated in Section A.3, 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 A.1) flaw hypotheses only. The fact that the iTC generates these candidate hypotheses indicates these must be addressed.

A.3 Reporting

416 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.

417 The public-facing report contains:

418 * 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;

419 * 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.

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

421 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:

  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 identified vulnerabilities (cf. AVA_VAN.1-11);

  f) 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

  g) 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. FDE Equivalency Considerations

422 **Introduction**

423 This appendix provides a foundation for evaluators to determine whether a vendor's request for equivalency of products for different OSs/platforms wishing to claim conformance to the FDE collaborative Protection Profiles.

424 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 OS/platform the product is tested (e.g., the testing environment**): The method a TOE provides functionality (or the functionality itself) may vary depending upon the OS on which it is installed. If there are no difference in the TOE provided functionality or in the manner in which the TOE provides the functionality, the models may be considered equivalent.

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

426 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.

427 If it is determined that a TOE operates the same regardless of the platform/OS it is installed within, testing may be performed on a single OS/platform combination 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.

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

429 **Evaluator guidance for determining equivalence**

430 The following table provides a description of how an evaluator should consider each of the factors that affect equivalency between TOE model variations and across operating environments. Additionally, the table also identifies scenarios that will result in additional separate testing across models/platforms.

Factor	Same/Not Same	Evaluator Guidance
Platform/Hardware Dependencies	Independent	If there are no identified platform/hardware dependencies, the evaluator shall consider testing on multiple hardware platforms to be equivalent.
	Dependencies	If there are specified differences between platforms/hardware, the evaluator must identify if the differences affect the cPP specified security functionality or if they apply to non-PP specified functionality. If functionality specified in the cPP is dependent upon platform/hardware provided services, the TOE must be tested on each of the different platform to be considered validated on that particular hardware combination. In these cases, the evaluator has the option of only re-testing the functionality dependent upon the platform/hardware provided functionality. If the differences only affect non-PP specified functionality, the variations may still be considered equivalent. For each difference the evaluator must provide an explanation of why the difference does or does not affect cPP specified functionality.
Software/OS Dependencies	Independent	If there are no identified software/OS dependencies, the evaluator shall consider testing on multiple OSs to be equivalent.
	Dependencies	If there are specified differences between OSs, the evaluator must identify if the differences affect the cPP specified security functionality or if they apply to non-PP specified functionality. If functionality specified in the cPP is dependent upon OS provided services, the TOE must be tested on each of the different OSs. In these cases, the evaluator has the option of only re-testing the functionality dependent upon the OS provided functionality. If the differences only affect non-PP specified functionality, the model variations may still be considered equivalent. For each difference the evaluator must provide an explanation of why the difference does or does not affect cPP specified functionality.
Differences in TOE Software Binaries	Identical	If the model binaries are identical, the model variations shall be considered equivalent.
	Different	If there are differences between model software binaries, a determination must be made if the differences affect cPP-specified security functionality. If cPP-specified functionality is affected, the models are not considered equivalent and must be tested separately. The evaluator has the option of only retesting the functionality that was affected by the software differences. If the differences only affect non-PP specified functionality, the models may still be considered equivalent. For each difference the evaluator must provide an explanation of why the difference does or does not affect cPP specified functionality.

Factor	Same/Not Same	Evaluator Guidance
Different in Libraries Used to Provide TOE Functionality	Same	If there are no differences between the libraries used in various TOE models, the model variations shall be considered equivalent.
	Different	If the separate libraries are used between model variations, a determination if the functionality provided by the library affects cPP-specified functionality must be made. If cPP-specified functionality is affected, the models are not considered equivalent and must be tested separately. The evaluator has the option of only retesting the functionality that was affected by the differences in the included libraries. If the different libraries only affect non-PP specified functionality, the models may still be considered equivalent. For each different library, the evaluator must provide an explanation of why the different libraries do or do not affect cPP specified functionality.
TOE Management Interface Differences	Consistent	If there are no differences in the management interfaces between various TOE models, the models variations shall be considered equivalent.
	Differences	If the TOE provides separate interfaces based on either the OS it is installed on or the model variation, a determination must be made if cPP-specified functionality can be configured by the different interfaces. If the interface differences affect cPP-specified functionality, the variations/OS installations are not considered equivalent and must be separately tested. The evaluator has the option of only retesting the functionality that can be configured by the different interfaces (and the configuration of said functionality). If the different management interfaces only affect non-PP specified functionality, the models may still be considered equivalent. For each management interface difference, the evaluator must provide an explanation of why the different management interfaces do or do not affect cPP specified functionality.
TOE Functional Differences	Identical	If the functionality provided by different TOE model variation is identical, the models variations shall be considered equivalent.
	Different	If the functionality provided by different TOE model variations differ, a determination must be made if the functional differences affect cPP-specified functionality. If cPP-specific functionality differs between models, the models are not considered equivalent and must be tested separately. In these cases, the evaluator has the option of only retesting the functionality that differs model-to-model. If the functional differences only affect non-cPP specified functionality, the model variations may still be considered equivalent. For each difference the evaluator must provide an explanation of why the difference does or does not affect cPP specified functionality.

431 **Strategy**

432 When performing the equivalency analysis, the evaluator should consider each factor independently. 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 TOE has been identified to require separate testing to ensure that it operates identically to all other equivalent TOE. The analysis would identify the specific combinations of models/testing environments that needed to be tested.

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

434 **Test presentation/Truth in advertising**

435 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 publically included.