Australian Government
Department of Defence

ASD-Approved Supporting Document

Mandatory Technical Document for Full Drive Encryption – Encryption Engine

Version:	**1.0**
Technology type:	**Data Protection**
Authored by:	**Common Criteria**
Publication date:	**26 January 2015**
ASD approval date:	**30 May 2016**

The following document is a Supporting Document (SD) authored by Common Criteria. This Supporting document has been approved for use by the Australian Signals Directorate.

This Mandatory Technical Document supports the collaborative Protection Profile for Full Drive Encryption – Encryption Engine.

This SD has been developed by the Full Drive Encryption – Encryption Engine iTC and is designed to be used to support the evaluations of products.

For information relating to the application of Protection Profiles, please refer to the Australian Government Information Security Manual (ISM) or www.asd.gov.au Controls in the ISM take precedence over any requirements contained in this SD where there is a conflict.

Supporting Document
Mandatory Technical Document

Full Drive Encryption: Encryption Engine

January 2015

Version 1.0

CCDB-2015-01-004

Table of Contents

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)

V 1.0 January 2015 (Incorporated changes due to comments received from CCDB review)

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.

Table of Contents

Table of Contents

List of Tables

1 Introduction

1.1 Technology Area and Scope of Supporting Document

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.

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.

The *FDE cPP - Encryption Engine* describes the requirements for the Encryption Engine piece and details the necessary security requirements and assurance 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.

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

a) *collaborative Protection Profile for Full Drive Encryption - Encryption Engine, January 26 2015.*

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

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

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.

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.

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 Glossary

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

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).

2 Evaluation Activities for SFRs

2.1 Class: Cryptographic Support (FCS)

2.1.1 FCS_CKM.1 Cryptographic key generation (Data Encryption Key)

TSS

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).

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.

KMD

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. The evaluator shall verify that the KMD describes how the TOE unwraps the DEK.

Test

The evaluator shall perform the following test activities:

- The evaluator shall configure the TOE to ensure the functionality of all selections.

2.1.2 FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction

TSS

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.

KMD

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

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 for the destruction.

FCS_CKM.4 Cryptographic key destruction

KMD

The evaluator shall check to ensure the KMD lists each type of key material its origin, possible temporary locations (e.g. key register, cache memory, stack, FIFO) and storage location. The evaluator shall verify that the KMD describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, etc.).

The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

The evaluator shall check to ensure the KMD lists each type of key material (software-based key storage, BEVs, passwords, etc.) and its origin, storage location, and the method for destruction for each key.

Test

For each software and firmware key clearing situation the evaluator shall repeat the following tests for Volatile Memory. For the test below, “key” refers to keys and key material.

These tests do not apply to hardware devices, such as Self Encrypting Drives.

- *Test 1:* The evaluator shall utilize appropriate combinations of specialized operational environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared correctly, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

  For each key subject to clearing, including copies of keys that are originally encrypted and stored in non-volatile memory by the TOE, the evaluator shall:

  1. Attach to the TOE software/firmware with a debugger.
  2. Record the value of the key in the TOE subject to clearing.
  3. Cause the TOE to perform a normal cryptographic processing with the key from #1.
  4. Cause the TOE to clear the key.
  5. Cause the TOE to stop the execution but not exit.
  6. Cause the TOE to dump the entire memory footprint of the TOE into a binary file.
  7. Search the content of the binary file created in #6 for instances of the known key value from #2.

  The test succeeds if no copies of the key from #4 are found in step #7 above and fails otherwise.

The evaluator shall perform this test on all keys, including those persisted in encrypted form, to ensure intermediate copies are cleared.

2.1.3 FCS_KYC_EXT.2 (Key Chaining)

KMD

The evaluator shall examine the KMD describes a high level description of the key hierarchy for all authorizations methods selected in FCS_AFA_EXT.1 that are used to protect the BEV. The evaluator shall examine the KMD to ensure it describes the key chain in detail. 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), FCS_COP.1(g).

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.

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

2.1.4 FCS_SMV.EXT.1 Validation

TSS

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

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).

KMD

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.

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.

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

Operational Guidance

[conditional] If 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.

Test

The evaluator shall perform the following tests:

- 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 user 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.

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

TSS

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.

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.

2.2 Class: User Data Protection (FDP)

2.2.1 FDP_DSK_EXT.1 Extended: Protection of Data on Disk

TSS

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.

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.

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.

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.

Operational Guidance

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.

Test

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.

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).

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.

The evaluator shall perform the following tests:

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;

- 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.

- 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;

  - 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 Class: Security Management (FMT)

2.3.1 FMT_SMF.1 Specification of management functions

TSS

Option A: The evaluator shall ensure the TSS describes how the TOE changes the DEK.

Option B: The evaluator shall ensure the TSS describes how the TOE cryptographically erases the DEK.

Option C: The evaluator shall ensure the TSS describes the process to initiate TOE firmware/software updates.

Option D: If additional management functions are claimed in the ST, the evaluator shall verify that the TSS describes those functions.

KMD

Option D: 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.

Operational Guidance

Option A: 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.

Option C: The evaluator shall examine the operational guidance to ensure that it describes how to initiate TOE firmware/software updates.

Option D: 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.

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

Test

Option A and B: 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).

Option C: The evaluator shall verify that the TOE has the functionality to initiate TOE firmware/software updates.

Option D: If additional management functions are claimed, the evaluator shall verify that the additional features function as described.

2.4 Class: Protection of the TSF (FPT)

FPT_KYP_EXT.1 Extended: Protection of Key and Key Material

KMD

The evaluator shall verify the KMD for a high level description of method used to protect keys stored in non-volatile memory.

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 FCS_COP.1(d) or FCS_COP.1(g) 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 FPT_TUD_EXT.1 Trusted Update

TSS

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.

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.

Operational Guidance

The evaluator ensures that the Operational Guidance describes how the vendor provides updates for 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.

Test

The evaluators shall perform the following tests (if the TOE supports multiple signature each using a different hash algorithm, then the evaluator performs tests 2 and 3 for different combinations of authentic and unauthentic digital signatures and hashes, as well as for digital signature alone):

> 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.

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 assurance activity tests to demonstrate that the update functions as expected. FPT_TST_EXT.1 TSF Testing

TSS

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

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, for 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.

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.

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

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.

Optional Requirements

2.5 Cryptographic Support (FCS)

2.5.1 FCS_KDF_EXT.1 Cryptographic Key Derivation

TSS

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 SP800-132.

KMD

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.

2.5.2 FCS_CKM.1(b) Cryptographic Key Generation (Asymmetric Keys)

TSS

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.

Operational Guidance

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 defined in this cPP.

Test

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.

Key Generation for FIPS PUB 186-4 RSA Schemes

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*.

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

1. Random Primes:
   - Provable primes
   - Probable primes
2. Primes with Conditions:
   - Primes *p1, p2, q1,q2, p* and *q* shall all be provable primes
   - Primes *p1, p2, q1,* and *q2* shall be provable primes and *p* and *q* shall be probable primes
   - Primes *p1, p2, q1,q2, p* and *q* shall all be probable primes

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. The public key exponent shall be odd integer in the range (2^16, 2^256). 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.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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.

Key Generation for Finite-Field Cryptography (FFC)

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

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:

Cryptographic Group Generator:

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

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

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 where 1<= x<=q-1.

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

2.5.3 FCS_COP.1(f) Cryptographic operation (AES Data Encryption/Decryption)

TSS

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

Guidance

If multiple encryption modes are supported, the evaluator examines the guidance documentation to determine how a specific mode/key-size is chosen by the end user.

Test

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

AES-CBC Tests

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. 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.

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values 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. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values 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. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key *i* in each set shall have the leftmost *i* bits be ones and the rightmost *N-i* bits be zeros, for *i* in [1,N].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall

be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an *i*-block message where 1 < *i* <=10. The evaluator shall choose a key, an IV and plaintext message of length *i* blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an *i*-block message where 1 < *i* <=10. The evaluator shall choose a key, an IV and a ciphertext message of length *i* blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
        if i == 1:
                CT[1] = AES-CBC-Encrypt(Key, IV, PT)
                PT = IV
        else:
                CT[i] = AES-CBC-Encrypt(Key, PT)
                PT = CT[i-1]
```

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

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

AES-GCM Test

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.

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.

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.

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.

XTS-AES Test

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 2^{16} bits, whichever is smaller.

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.

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.

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.

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

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 assurance activities section 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.

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

TSS

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" vice “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).

Test

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.

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 assurance activities and any additional components.

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

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.

ECDSA Algorithm Tests

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.

RSA Signature Algorithm Tests

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

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.

2.5.5 FCS_COP.1(d) Cryptographic operation (Key Wrapping)

TSS

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.

KMD

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.

2.5.6 FCS_COP.1(b) Cryptographic operation (Hash Algorithm)

TSS
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.

Operational Guidance
The evaluator checks the operational guidance documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present.

Test
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.

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.

Short Messages Test - Bit-oriented Mode
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.

Short Messages Test - Byte-oriented Mode
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.

Selected Long Messages Test - Bit-oriented Mode
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 $1 \leq i \leq m$. For SHA-512, the length of the i-th message is $1024 + 99*i$, where $1 \leq i \leq m$. 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.

Selected Long Messages Test - Byte-oriented Mode
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 $1 \leq i \leq m/8$. For SHA-512, the length of the i-th message is $1024 + 8*99*i$, where $1 \leq i \leq m/8$. 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.

Pseudorandomly Generated Messages Test
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 [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

2.5.7 FCS_COP.1(c) Cryptographic operation (Keyed Hash Algorithm)

TSS

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.

Test

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.

Selection-Based Requirements

2.6 Cryptographic Support (FCS)

2.6.1 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

TSS

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.

Entropy Documentation

The evaluator shall ensure the Entropy Essay provides all of the required information as described in Appendix D of the cPP. The evaluator assesses the information provided and ensures the TOE is providing sufficient entropy when it is generating a Random Bit String.

Operational Guidance

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.

Test

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.

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).

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.

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

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

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.

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.

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

3 Evaluation Activities for SARs

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.

In cases where the requirements are not technology dependent, the evaluator is expected to perform the CEM work units (e.g., ASE, ALC_CMC.1, ALC_CMS.1), those activities are not repeated here, rather they are expressed as part of the cPP.

3.1 ASE: Security Target Evaluation

1 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.

3.1.1 Conformance claims (ASE_CCL.1)

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

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.

3.2 ADV: Development

3.2.1 Basic Functional Specification (ADV_FSP.1)

The Evaluation Activities for this assurance component focus on understanding the interfaces presented in the TOE Summary Specification (TSS) in response to the functional requirements, and on the interfaces presented in the AGD documentation. Specific requirements on this documentation are identified (where relevant) for each SFR in section 2 above, and in Evaluation Activities for AGD, ATE and AVA SARs in other parts of section 3 in this Supporting Document.

Evaluation Activity:

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

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., perform 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.

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.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any supplementary information required by the cPP for aspects such as entropy analysis or cryptographic key management architecture[1]: no additional "functional specification" documentation is necessary to satisfy the Evaluation Activities. The interfaces that need to be evaluated are also identified by reference to the assurance activities listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any supplementary information required by 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 Evaluation Activities for each SFR also means that the tracing required in ADV_FSP.1.2D is treated as implicit, and no separate mapping information is required for this element.

However, if the evaluator is unable to perform some other required Evaluation Activity 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'.

[1] The Security Target and AGD documentation are public documents. Supplementary information may be public or proprietary: the cPP and/or Evaluation Activity descriptions will identify where such supplementary documentation is permitted to be proprietary and non-public.

3.3 AGD: Guidance Documents

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.

3.3.1 Operational User Guidance (AGD_OPE.1)

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).

Evaluation Activity:

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).

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.

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.

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

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

a) 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.

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

c) 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.

d) 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.

3.3.2 Preparative Procedures (AGD_PRE.1)

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.

Evaluation Activity:

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

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

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.

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

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

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).

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.

The preparative procedures must include

a) instructions to successfully install the TSF in each Operational Environment; and

b) instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and

c) instructions to provide a protected administrative capability.

3.4 ATE: Tests

3.4.1 Independent Testing – Conformance (ATE_IND.1)

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.

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.

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.

Evaluation Activity:

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

Evaluation Activity:

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

Evaluation Activity:

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.

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.

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).

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.

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[2].

3.5 AVA: Vulnerability Assessment

3.5.1 Vulnerability Survey (AVA_VAN.1)

Evaluation Activity:

The evaluator shall document their analysis and testing of potential vulnerabilities with respect to this requirement. This report could be included as part of the test report for ATE_IND, or could be a separate document.

The evaluator performs a search of public information to determine the vulnerabilities that have been found in products representing the relevant TOE type (including vulnerabilities related to aspects such as components used in the TOE and the communication protocols that it uses) as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided for ATE_IND) to confirm the vulnerability, if suitable.

See Appendix A for more information on vulnerability assessment.

[2] 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.

4 Required Supplementary Information

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.

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.

5 References

[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model

CCMB-2012-09-001, Version 3.1 Revision 4, September 2012

Appendix A Vulnerability Analysis

This provides supplemental guidance for the AVA activities for the Authorization Acquisition (AA) and Encryption Engine (EE) cPPs. This guidance is based on version 0.2 of the SPD and v0.2 of the ESR, and the Draft cPP Vulnerability Analysis Whitepaper [VAWP].

Introduction

In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities and documents his findings such that others can follow his arguments and come to the same conclusion as the evaluator in his report. Consequently, assurance activities were created for the cPP based on the threat model and known vulnerabilities for the type of product being assessed.

This supplemental guidance process used by the evaluator to propose an additional assurance activity to the iTC that needs to be incorporated into the cPP based on their additional understanding achieved by evaluating a particular product. This process can also be used to propose additional activities as other vulnerabilities are discovered and made public.

Sources of vulnerability information

It is critical to remember that the use case for the FDE AA and EE Version 1 is rather straightforward – the device is found in a powered down situation and has not been subjected to revisit/evil maid attacks. 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 definition of a basic attack is limited to a very narrow threat window. For example, if a vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this cPP.

Process for Proposal of New Activities

The evaluation lab proposes to the validator that the Scheme propose a new assurance activity based on a type of vulnerability that the evaluator believes is not suitable addressed by following these steps:

1. The evaluator describes the type of vulnerability and how it applies to the threat model in the cPP.
2. The evaluator performs that activity for that product if approved by the validator. (The evaluator can of course always perform an activity that the vendor, evaluator, and the Certifier agrees is useful).
3. The evaluator and validator document a proposed assurance activity (or a revision to an assurance activity) based on the type of vulnerability that they determine should be incorporated into the cPP.

The iTC reads the document and determines whether to make a revision to the cPP based on whatever document or evidence is provided by the Scheme.

In the event a vulnerability that applies to the threat model is ever discovered in a product and is not mitigated to the satisfaction of the validator, the Scheme shall fail the product and report the vulnerability in a CVE.

Appendix B FDE Equivalency Considerations

Introduction

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.

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.

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

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.

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.

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

Evaluator guidance for determining equivalence

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

Factor	Same/Not Same	Evaluator guidance
		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.
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

Factor	Same/Not Same	Evaluator guidance
		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.

Table 1 - Evaluation Equivalency Analysis

Strategy

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.

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

Test presentation/Truth in advertising

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.

Appendix C: Glossary

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.
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.

Term	Meaning
Non-Volatile Memory	A type of computer memory that will retain information without power.
Powered-Off State	The device has been shutdown.
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]

See [CC1] for other Common Criteria abbreviations and terminology.

Appendix D: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