Common Criteria

Supporting Document
Mandatory Technical Document

Full Drive Encryption: Authorization Acquisition

September 2014

Version 0.2

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

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 products against the cPPs identified in section 1.1.

Technical Editor: *FDE iTC*

Document history:

V0.2, September 2014 (Initial Release for Public 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 Authorization Acquisition (AA), it may be difficult to trigger the interface to demonstrate the TSF is properly conditioning a password, or combining multiple submasks. 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 the AA is to gather user input and provide the Encryption Engine with a value that can be used to make the data encryption key available for encryption/decryption functions. In order to ensure comparable, transparent and repeatable evaluation of the implemented mechanisms, methods have to be described that may consist of agreed evaluation approaches, e.g. how to prove that the claimed functionality is really done by the product.

Field of special use: *Full Drive Encryption devices, specifically the set of security functional requirements associated with the Authorization Acquisition 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

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 user or authorization data.

The *FDE cPP - Authorization Acquisition* describes the requirements for the Authorization Acquisition piece and details the necessary security requirements and assurance activities necessary to interact with a user and result in the availability of a data encryption key (DEK).

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

a) *collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition, September 2014.*

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 Cryptographic Support (FCS)

2.1.1 FCS_AFA_EXT.1 Authorization Factor Acquisition

TSS

The evaluators shall first examine the TSS section to ensure that the authorization factors specified in the ST are described. For password-based factors the examination of the TSS section is performed as part of FCS_PCC_EXT.1 Evaluation Activities. Additionally in this case, the evaluator shall verify that the operational guidance discusses the characteristics of external authorization factors (e.g., how the authorization factor must be generated; format(s) or standards that the authorization factor must meet) that are able to be used by the TOE.

If other authorization factors are specified, then for each factor, the TSS specifies how the factors are input into the TOE; how a submask is produced from the authorization factor (including any associated standards to which this process might conform), and verification performed to ensure the length of the submask meets the required size (as specified in this requirement).

Operational Guidance

The evaluator shall verify that the AGD guidance includes instructions on all of the authorization factors. The AGD will discuss the characteristics of external authorization factors (e.g., how the authorization factor is generated; format(s) or standards that the authorization factor must meet, configuration of the TPM device used) that are able to be used by the TOE.

KMD

The evaluator shall examine the Key Management Description to confirm that the initial authorization factors (submasks) directly contribute to the unwrapping of the BEV.

Test

The password authorization factor is tested in FCS_PCC_EXT.1.

The evaluator shall also perform the following tests:

- 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 decrypted plaintext data.

2.1.2 FCS_KYC_EXT.1 (Key Chaining)

TSS

The evaluator shall verify that the TSS describes a high level description of the key hierarchy for all authorizations methods selected in FCS_CKM.EXT.1 that are used to protect the BEV.

The evaluator shall verify the TSS supports BEV outputs of no fewer 128 bits for products that support only AES-128, and no fewer than 256 bits for products that support AES-256.

KMD

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_COP.1(d) and FCS_KDF_EXT.1.

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 the 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 the initial authorization value and the effective strength of the BEV is maintained throughout the Key Chain.

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

2.1.3 FCS_PCC_EXT.1 Cryptographic Password Construct and Conditioning

TSS

The evaluator shall examine the TSS to ensure that the formation of the BEV and intermediary keys is described and that the key sizes match that selected by the ST Author.

The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function and is the same length as the BEV as specified above.

If any manipulation of the key is performed in forming the submask that will be used to form an intermediary key, that process shall be described in the TSS.

Test

The evaluator shall also perform the following tests:

- Test 1: Ensure that the TOE supports passwords/passphrases of 64 characters.

- Test 2: Try entering a password/passphrase less than 64 characters.
- Test 3: If the TOE supports a password/passphrase length up to a maximum number of characters, n (which would be greater than 64), then ensure that the TOE will not accept more than n characters.
- Test 4: Ensure that the TOE supports passwords consisting of all of the upper/lower case letters, numbers, and printable special characters.

2.1.4 FCS_CKM_EXT.3 Cryptographic Key and Key Material Destruction

TSS

The evaluator shall verify the TSS 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.

KMD

The evaluator shall verify the KMD includes a key lifecycle, that includes a description where key material resides, 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.

2.1.5 FCS_CKM.4 Cryptographic key destruction

TSS

The evaluator shall verify that the TSS 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").

KMD

The evaluator shall check to ensure the KMD lists each type of plaintext 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.

- *Test 1:* The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) to test that keys are cleared correctly, including all intermediate 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 intermediate copies of keys that are persisted encrypted by the TOE the evaluator shall:

  1. Load the instrumented TOE build in 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 #4 for instances of the known key value from #1.

  The test succeeds if no copies of the key from #1 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.

2.1.6 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 host platform.

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

2.1.7 FMT_SMF.1 Specification of management functions

TSS

Option A: The evaluator shall ensure the TSS describes how the TOE sends the request to the EE to change the DEK.

Option B: The evaluator shall ensure the TSS describes how the TOE sends the request to the EE to cryptographically erase the DEK.

Option C: The evaluator shall ensure the TSS describes the methods by which users may change their authorization factors.

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

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

Test

Option A and B: The evaluator shall verify that the product has the functionality to forward a command to the EE to change and cryptographically erase the DEK (effectively removing the ability to retrieve previous protected data).

Option C: The evaluator shall initialize the product such that it requires the user to input an authorization factor in order to access encrypted data.

> Test 1: The evaluator shall first provision user authorization factors, and then verify the authorization factors allow the user access to the encrypted data. Then the evaluator shall exercise the management functions to change a user's authorization factor to a new one. Then he or she will verify that the product denies access to the user's encrypted data when he or she uses the old or original authorization factor to gain access.

Option D: The evaluator shall verify that the product has the functionality to initiate system firmware/software updates.

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

Test 2: [conditional] If the TOE provides default authorization factors, the evaluator shall change these factors in the course of taking ownership of the device as described in the operational guidance. The evaluator shall then confirm that the (old) authorization factors are no longer valid for data access.
Test 3 [conditional] If the TOE provides key recovery capability whose effects are visible at the TOE interface, then the evaluator shall devise a test that ensures that the key recovery capability has been or can be disabled following the guidance provided by the vendor.

Operational Guidance

Option C: The evaluator shall examine the operational guidance to ensure that it describes how selected authorization factors are changed.

Option E: Default Auth Factors: It may be the case that the TOE arrives with default authorization factors in place. If it does, then the selection in section E 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.

2.1.8 FPT_KYP_EXT.1 Extended: Protection of Key and Key Material

TSS

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

KMD

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(c) is followed when storing keys in non-volatile memory.

2.1.9 FPT_TUD_EXT.1 Trusted Update

TSS

The evaluator shall verify the TSS to ensure that it describes information stating that an authorized source signs product updates and will have an associated signed hash. The evaluator shall verify the TSS contains a definition of an authorized source along with a description of how the product uses public keys for the update verification mechanism in the Operational Environment. The evaluator ensures the TSS contains this information and details any instructions dealing with the installation of the update credentials. The evaluator also ensures that the operational guidance describes how the product obtains candidate updates; the processing associated with verifying the digital signature of the updates; and the actions that take place for successful and unsuccessful cases. If the Operational Environment performs the digital hashing and signature verification, then the evaluator shall verify the TSS to ensure it describes -- for each platform identified in the TSS --the interface(s) used by the product to invoke this cryptographic functionality.

Test

The evaluators shall perform the following tests (if the products supports an optional hash, then the evaluator performs tests 2 and 3 for different combinations of valid and invalid 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 product. 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 it an update successfully installs it on the product. The evaluator shall perform a subset of other assurance activity tests to demonstrate that the update functions as expected.

• Test 3: The evaluator obtains or produces an illegitimate update, and attempts to install it on the product. The evaluator verifies that the product rejects the update.

Optional Requirements

2.1.10 FCS_CKM.1 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. 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.

2.1.11 FCS_SMC_EXT.2 Submask Combining

TSS

If the submasks produced from the authorization factors are XORed together to form the BEV, 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.

KMD

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.

Test

- 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.

2.1.12 FCS_VAL.EXT.1 Validation

TSS

The evaluator shall examine the TSS to determine how the TOE's brute-force attack protection ensures that only 300 attempts can be made within a 24 hour period. The TSS also identifies the method used to validate the submasks, and describes the validation process at a high level. If multiple submasks are used within the product, the TSS describes how the submasks are validated (e.g., each submask validated before combining, once combined validation takes place).

The evaluator shall examine the TSS to verify that it describes the methods the product employs to limit the number of consecutively failed authorization attempts.

KMD

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 submasks.

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

Operational Guidance

[conditional] 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 product 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 product behaves as described in the TSS.
- Test 2: For each supported authorization factor, ensure that when the user provides an incorrect authorization factor, the TOE prevents the BEV from being forwarded outside the TOE (e.g., to the EE).

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

Test

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

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

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.

<u>RSA Signature Algorithm Tests</u>

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.
The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid 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.

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.

2.1.14 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 testmacs.

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.

<u>Short Messages Test - Bit-oriented Mode</u>

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. The length of the ith message is 512 + 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. The length of the ith message is 512 + 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.

2.1.15 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 and IV using a known good implementation.

2.1.16 FPT_TST_EXT.1 Extended: 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 product, the method by which the product tests those functions. The evaluator shall verify that the TSS includes each of these functions, the method by which the product verifies the correct operation of the function.

Selection-Based Requirements

2.1.17 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.

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

Test

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

<u>AES-CBC Tests</u>

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.

<u>XTS-AES Monte Carlo Test</u>

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.

2.1.19 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.

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.

2.1.20 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.

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, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

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.

3.1 ADV: Development

3.1.1 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.

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’.

3.2 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.

[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.

3.2.1 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.

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 product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

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 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.

3.2.2 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 product itself).

Preparative procedures must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

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.

3.3 ALC: Life-cycle Support

3.3.1 Labelling of the TOE (ALC_CMC.1)

Evaluation Activity:

The evaluator shall check the ST and any deliverables needed to provide required supplementary information to ensure that they contain an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the identifier value specified there is consistent with that in the ST.

If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the certified version of the product.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Evaluation Activity:

The "evaluation evidence required by the SARs" in ALC_CMS.1.1C is limited to the information in the ST, the AGD documentation, and any deliverables needed to provide the required supplementary information. By ensuring that the TOE is specifically identified and that this identification is consistent in these documents (as checked in the Evaluation

Activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component.

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.

Evaluation Activity:

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.

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 the Authorization Acquisition requires an entropy analysis, and key management description. 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

A Vulnerability Analysis

This document provides the 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.

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 product 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 product 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 product 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 product 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 product 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.