Supporting Document: Evaluation Activities for collaborative Protection Profile for Dedicated Security Component

Mandatory Technical Document

Version 1.0.1, December 13, 2022

Foreword

This is a Supporting Document (SD), intended to complement the Common Criteria Version 3.1 Release 5 and the associated Common Evaluation Methodology for Information Technology Security Evaluation.

SDs 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. SDs may also be "Mandatory Technical Documents," that have a mandatory application for evaluations where the scope is covered by that of the SD. The usage of SDs of the latter class is not only mandatory, but certificates issued as a result of their application are recognized under the CCRA.

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

Technical Editor:

DSC iTC

Document history:

v1.0, September 10, 2020 - initial draft

v1.0.1, December 13, 2022 - conversion to asciidoc

General Purpose:

The purpose of a DSC is to provide seven security services to devices that incorporate them: Parse, Provision, Protect, Process, Prove, Purge, and Propagate. The DSC acts as a secure and separate processing area that stores and manipulates SDOs and SDEs that are isolated from the remainder of the device.

Field of special use:

Vendors and integrators of mobile devices and other hardware devices that use DSCs.

Acknowledgements:

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

Table of Contents

Chapter 1. Introduction

1.1. Technology Area and Scope of Supporting Document

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

a. collaborative Protection Profile for Dedicated Security Component, Version 1.0.1 - December 13, 2022

A Dedicated Security Component (DSC), in the context of this cPP, is the combination of a hardware component and its controlling OS or firmware dedicated to the protection and safe use by the rich OS of Security Data Objects (SDOs) consisting of keys, identities, attributes, and Security Data Elements (SDEs).

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

1.2. Structure of the Document

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

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

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

Many of the test EAs in this Supporting Document require the evaluator to directly exercise low-level interfaces to the DSC to manipulate it in a manner that may not be feasible with a commercially-available model of the DSC and associated tools. In such cases it is acceptable for the TOE developer to produce materials that allow for the required tests to be executed. As such materials may be proprietary to the vendor, it is sufficient for the evaluator to execute these tests (or observe the execution of these tests) at the TOE developer's facility. For any tests that are executed in this manner, the evaluator shall ensure the following:

- The test report shall document the measures the evaluator took to gain assurance that if the TOE itself is modified to allow for certain tests to be performed, the security of the TOE is not reduced in the unmodified TOE (i.e. if the TOE is modified to use a special firmware build, this should not create a situation where the modified build enforces required security functionality that the unmodified build does not).
- Any tools used to conduct the required testing shall produce sufficient evidence to demonstrate that the test was successful (e.g., if a tool is designed to erase a particular key, it should also attempt to perform some operation that requires the use of that key to provide evidence that the key destruction succeeded).
- The evaluator shall ensure that the tool actually performs the intended action and does not create a contrived outcome that imitates the results of a passing test without performing the actual operation (e.g. if a tool is designed to erase a particular key and output its value as proof of this, the tool should be obtaining the actual key value and not simply returning a static result).

The test EAs for individual SFRs identify cases where developer tools may be needed to execute the test as written.

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

Note that certain parts of EAs may or may not be required depending on whether certain selections are made in the Security Target for the corresponding SFR. Underlined text is used to represent specific selection items.

1.3. Terminology

1.3.1. Glossary

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

Table 1. Glossary

Term	Meaning
Access	In the context of SDOs, access to an SDO represents the list of actions permissible with an SDO, including its generation, use, modification, propagation, and destruction.

Term	Meaning
Administrator	A type of user that has special privileges to manage the TSF.
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
Attestation	The process of presenting verifiable evidence describing those characteristics that affect integrity. Examples of these characteristics are boot firmware and boot critical data which, combined, describe the way the DSC booted. [SA]
Attributes	Indications of characteristics or properties of the SDEs bound in an SDO.
Authorization Value	Critical data bound to an action by itself or to action on a subject. Such data, when presented to the TOE, authorizes the action by itself or authorizes the action on or with the subject respectively.
Authorization Data	Collective term for authentication tokens and authorization values.
Authentication Token	Critical data bound to a user. Such data, when presented to the TOE and successfully verified by it, authenticates the user. The TOE may use the successful authentication of a user as an authorization to execute an action on its behalf.
Authenticator	A shortened name for Authentication Token.
Boot Critical Data	Critical data that persists across power cycles and determines characteristics of the DSC. Examples of boot critical data can be DSC configuration settings, certificates, and the results of measurements obtained by the root of trust for measurement.
Boot Firmware	The first firmware that executes during the boot process.
Chain of Trust	A Chain of Trust is anchored in a Root of Trust and extends a trust boundary by verifying the authenticity and integrity of successive components before passing control to those components. [SA]
Client Application	Entity who relies on the services provided by the platform or DSC.
Data Encryption Key	An encryption key, usually for a symmetric algorithm, that encrypts data that is not keying material.
Integrity	Assurance of trustworthiness and accuracy.
Immutable	Unchangeable.
Key Encryption Key	An encryption key that encrypts other keying material. This is sometimes called a key wrapping key. A KEK can be either symmetric or asymmetric.
Known Answer Tests (KATs)	Test vectors or data generated to determine the correctness of an implementation.
Operator	Human being who has physical possession of the platform on which the DSC is located. [GD]
Owner	Human being who controls or manages the platform on which the DSC is located. May be remote. [GD]
Platform	A platform consists of the hardware and firmware of a computing entity.

Term	Meaning
Pre-installed SDO	An SDO installed on the DSC by the manufacturer. The SDO consists of an SDE and attributes, which if not explicitly expressed in a data structure, are implicit based on the functions that have exclusive access to the SDE.
Privileged Function	Functions restricted to the role of administrator, which may include, but are not limited to, provisioning keys, provisioning user authorization values, de-provisioning user authorization values, provisioning administrator authorization values, changing authorization values, disabling key escrow, and configuring cryptography.
Protected Data Blob	Data in an encrypted structure that protects its confidentiality or integrity (as required by the context in which it is used).
Protected Storage	Refers to DSC hardware used to store SDEs or SDOs, and provide integrity protection for all items and confidentiality for those items that require it. Protected Storage may also refer to storage external to the DSC, which is usually encrypted by keys maintained by the DSC's internal protected storage capabilities.
Protections	Mechanisms that ensure components of a DSC (executable firmware code and critical data) remain in a state of integrity and are protected from modification outside of authorized, authenticated processes and entities. [NIST-ROTM]
Remote Secure Channel	Logical channel to the DSC from a remote entity, which cryptographically protects the confidentiality and integrity of the channel content.
Required Supplementary Information	Information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP (see description in Section 6).
Root Encryption Key	An encryption key that serves as the anchor of a hierarchy of keys.
Root of Trust	A root of trust performs one or more security specific functions; establishing the foundation on which all trust in a system is placed. [NIST-ROTM]
Root of Trust for Authorization	(As defined by [GP_ROT]) The Root of Trust for Authorization provides reliable capabilities to assess authorization tokens and determine whether or not they satisfy policies for access control.
Root of Trust for Confidentiality	(As defined by [GP_ROT]) The Root of Trust for Confidentiality maintains shielded locations for the purpose of storing sensitive data, such as secret keys and passwords.
Root of Trust for Integrity	(As defined by [GP_ROT]) The Root of Trust for Integrity maintains shielded locations for the purpose of storing and protecting the integrity of non-secret critical security parameters and platform characteristics. Critical security parameters include, but are not limited to, authorization values, public keys, and public key certificates.

Term	Meaning
Root of Trust for Measurement	(As defined by [GP_ROT]) The Root of Trust for Measurement provides the ability to reliably create platform characteristics.
Root of Trust for Reporting	(As defined by [GP_ROT]) The Root of Trust for Reporting reliably reports platform characteristics. It provides an interface that limits its services to providing reports on its platform characteristics authenticated by a platform identity.
Root of Trust for Storage	A root of trust that acts as the Root of Trust for Confidentiality and the Root of Trust for Integrity.
Root of Trust for Update	A root of trust responsible for updating the firmware.
Root of Trust for Verification	A root of trust responsible for verifying digital signatures.
Security Data Element	A Critical Security Parameter, such as a cryptographic key or authorization token.
Security Data Object	A Security Data Object (SDO) may include one or more SDEs. SDOs bind SDEs with a set of attributes.
Symmetric Encryption Key	A value intend to input as a key to a symmetric encryption algorithm, such as AES.
System	A system consists of the platform hardware and firmware in addition to the higher-level software running on top of it (kernel, user-space processes, etc.).
Target of Evaluation	A set of software, firmware or hardware possibly accompanied by guidance. [CC1]
TOE Security Functionality (TSF)	A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs. [CC1]
Trusted Local Channel	Physical channel to the DSC within the platform of which the DSC is a part, which is protected by the operational environment to ensure confidentiality and integrity.
TSF Data	Data for the operation of the TSF upon which the enforcement of the requirements relies.
User	An administrator or client application.

1.3.2. Acronyms

Table 2. Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
CA	Certificate Authority
CBC	Cipher Block Chaining

Acronym	Meaning
CCM	Counter with CBC-Message Authentication Code
CCMP	CCM Protocol
CPU	Central Processing Unit
CSP	Critical Security Parameter
DAR	Data At Rest
DEK	Data Encryption Key
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standards
FQDN	Fully Qualified Domain Name
GCM	Galois Counter Mode
HMAC	Keyed-Hash Message Authentication Code
HTTPS	Hypertext Transfer Protocol Secure
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPsec	Internet Protocol Security
KEK	Key Encryption Key
KMAC	KECCACK Message Authentication Code
NIST	National Institute of Standards and Technology
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PP	Protection Profile
RA	Registration Authority
RBG	Random Bit Generator
REK	Root Encryption Key
ROM	Read-only memory
RSA	Rivest Shamir Adleman Algorithm
SDE	Security Data Element
SDO	Security Data Object
SFP	Security Function Policy

Acronym	Meaning
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SK	Symmetric Key or Symmetric Encryption Key
SPI	Security Parameter Index
SSH	Secure Shell
ST	Security Target
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
USB	Universal Serial Bus

1.4. Test Environment Note

Many of the test EAs in this Supporting Document require the evaluator to directly exercise low-level interfaces to the DSC to manipulate it in a manner that may not be feasible with a commercially-available model of the DSC and associated tools. In such cases it is acceptable for the TOE developer to produce materials that allow for the required tests to be executed. As such materials may be proprietary to the vendor, it is sufficient for the evaluator to execute these tests (or observe the execution of these tests) at the TOE developer's facility. For any tests that are executed in this manner, the evaluator shall ensure the following:

- The test report shall document the measures the evaluator took to gain assurance that if the TOE itself is modified to allow for certain tests to be performed, the security of the TOE is not reduced in the unmodified TOE (i.e. if the TOE is modified to use a special firmware build, this should not create a situation where the modified build enforces required security functionality that the unmodified build does not).
- Any tools used to conduct the required testing shall produce sufficient evidence to demonstrate that the test was successful (e.g., if a tool is designed to erase a particular key, it should also attempt to perform some operation that requires the use of that key to provide evidence that the key destruction succeeded).
- The evaluator shall ensure that the tool actually performs the intended action and does not create a contrived outcome that imitates the results of a passing test without performing the actual operation (e.g. if a tool is designed to erase a particular key and output its value as proof of this, the tool should be obtaining the actual key value and not simply returning a static result).

The test EAs for individual SFRs identify cases where developer tools may be needed to execute the test as written.

Chapter 2. Evaluation Activities for SFRs

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

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

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

Finally, the subsection labelled Tests is where the iTC has determined that testing of the product in the context of the associated SFR is necessary. While the evaluator is expected to develop tests, there may be instances where it is more practical for the developer to construct tests, or where the developer may have existing tests, as mentioned in section 1.4 above. Therefore, it is acceptable for the evaluator to witness developer-generated tests in lieu of executing the tests. In this case, the evaluator must ensure the developer's tests are executing both in the manner declared by the developer and as mandated by the EA. The CEM work units that are associated with the EAs specified in this section are: ATE_IND.1-3, ATE_IND.1-4, ATE_IND.1-5, ATE_IND.1-6, and ATE_IND.1-7.

2.1. Cryptographic Support (FCS)

2.1.1. Cryptographic Key Management (FCS_CKM)

2.1.1.1. FCS_CKM.1 Cryptographic Key Generation

2.1.1.1.1. TSS

The evaluator shall examine the TSS to verify that it describes how the TOE obtains a cryptographic key through importation of keys from external sources as specified in FDP_ITC_EXT.1 and FDP_ITC_EXT.2. The evaluator shall also examine the TSS to determine whether it describes any supported asymmetric or symmetric key generation functionality consistent with the claims made in FCS_CKM.1.1.

2.1.1.1.2. AGD

The evaluator shall verify that the guidance instructs the administrator how to configure the TOE to use the selected key types for all uses identified in the ST.

2.1.1.1.3. Test

Testing for this function is performed in conjunction with FDP_ITC_EXT.1 and FDP_ITC_EXT.2. If asymmetric or symmetric key generation functionality is claimed, testing for this function is also performed in conjunction with FCS_CKM.1/AK or FCS_CKM.1/SK.

2.1.1.1.4. KMD

The evaluator shall confirm that the KMD describes:

- The parsing interface and how the TSF imports keys for internal use
- The asymmetric key generation interfaces and how the TSF internally creates asymmetric keys, if claimed
- The symmetric key generation interfaces and how the TSF internally creates symmetric keys, if claimed

If the TOE uses the generated key in a key chain/hierarchy then the KMD shall describe how the key is used as part of the key chain/hierarchy.

2.1.1.2. FCS_CKM.1/KEK Cryptographic Key Generation (Key Encryption Key)

2.1.1.2.1. TSS

The evaluator shall examine the key hierarchy section of the TSS to ensure that the formation of all KEKs is described and that the key sizes match that described by the ST author. The evaluator shall examine the key hierarchy section of the TSS to ensure that each KEK encrypts keys of equal or lesser security strength using one of the selected methods.

[conditional] If the KEK is generated according to an asymmetric key scheme, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_CKM.1/AK is invoked. The evaluator uses the description of the key generation functionality in FCS_CKM.1/AK or documentation available for the operational environment to determine that the key strength being requested is greater than or equal to 112 bits.

[conditional] If the KEK is generated according to a symmetric key scheme, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_CKM.1/SK is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1, or the key derivation functionality in either FCS_CKM_EXT.5 or FCS_COP.1/PBKDF, depending on the key generation method claimed, to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.

[conditional] If the KEK is formed from derivation, the evaluator shall verify that the TSS describes the method of derivation and that this method is consistent with FCS_CKM_EXT.5.

2.1.1.2.2. AGD

There are no guidance evaluation activities for this component.

2.1.1.2.3. Test

The evaluator shall iterate through each of the methods selected by the ST and perform all applicable tests from the selected methods.

2.1.1.2.4. KMD

The evaluator shall iterate through each of the methods selected by the ST and confirm that the KMD describes the applicable selected methods.

2.1.1.3. FCS_CKM.2 Cryptographic Key Establishment

2.1.1.3.1. TSS

The evaluator shall examine the TSS to ensure that ST supports at least one key establishment scheme. The evaluator also ensures that for each key establishment scheme selected by the ST in FCS_CKM.2.1 it also supports one or more corresponding methods selected in FCS_COP.1/KAT. If the ST selects RSA in FCS_CKM.2.1, then the TOE must support one or more of "KAS1," or "KAS2," "KTS-OAEP," from FCS_COP.1/KAT. If the ST selects elliptic curve-based, then the TOE must support one or more of "ECDH-NIST" or "ECDH-BPC" from FCS_COP.1/KAT. If the ST selects Diffie-Hellman-based key establishment, then the TOE must support "DH" from FCS_COP.1/KAT.

2.1.1.3.2. AGD

The evaluator shall verify that the guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme .

2.1.1.3.3. Test

Testing for this SFR is performed under the corresponding functions in FCS_COP.1/KAT.

2.1.1.3.4. KMD

There are no KMD evaluation activities for this component.

2.1.1.4. FCS_CKM.4 Cryptographic Key Destruction

2.1.1.4.1. TSS

The evaluator shall examine the TSS to ensure it lists all relevant keys and keying material (describing the source of the data, all memory types in which the data is stored (covering storage both during and outside of a session, and both plaintext and non-plaintext forms of the data)), all relevant destruction situations (including the point in time at which the destruction occurs; e.g. factory reset or device wipe function, change of authorization data, change of DEK, completion of use of an intermediate key) and the destruction method used in each case. The evaluator shall confirm that the description of the data and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys in the key chain are accounted for[1]).

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

may be prevented or delayed.

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

2.1.1.4.2. AGD

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

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

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may implement wear-levelling and garbage collection. This may result in additional copies of the data that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command[2] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and guidance documentation).

2.1.1.4.3. 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.

The evaluator shall perform the following tests:

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

The evaluator shall:

1. Record the value of the key or keying material.
2. Cause the TOE to dump the SDO/SDE memory of the TOE into a binary file.
3. Search the content of the binary file created in Step #2. to locate all instances of the known key value from Step #1.

   Note that the primary purpose of Step #3. is to demonstrate that appropriate search commands are being used for Steps #8. and #9.

4. Cause the TOE to perform normal cryptographic processing with the key from Step #1.
5. Cause the TOE to destroy the key.

6. Cause the TOE to stop execution but not exit.
7. Cause the TOE to dump the SDO/SDE memory of the TOE into a binary file.
8. Search the content of the binary file created in Step #7. for instances of the known key value from Step #1.
9. Break the key value from Step #1. into an evaluator-chosen set of fragments and perform a search using each fragment. (Note that the evaluator shall first confirm with the developer how the key is normally stored, in order to choose fragment sizes that are the same or smaller than any fragmentation of the data that may be implemented by the TOE. The endianness or byte-order should also be taken into account in the search.)

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

Step #9 ensures that partial key fragments do not remain in memory. If the evaluator finds a 32-or-greater-consecutive-bit fragment, then fail immediately. Otherwise, there is a chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is also found in this repeated run then the test fails unless the developer provides a reasonable explanation for the collision, then the evaluator may give a pass on this test.

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

1. Record the value of the key or keying material.
2. Cause the TOE to perform normal cryptographic processing with the key from Step #1.
3. Search the non-volatile memory the key was stored in for instances of the known key value from Step #1.

   Note that the primary purpose of Step #3. is to demonstrate that appropriate search commands are being used for Steps #5 and #6.

4. Cause the TOE to clear the key.
5. Search the non-volatile memory in which the key was stored for instances of the known key value from Step #1. If a copy is found, then the test fails.
6. Break the key value from Step #1. into an evaluator-chosen set of fragments and perform a search using each fragment. (Note that the evaluator shall first confirm with the developer how the key is normally stored, in order to choose fragment sizes that are the same or smaller than any fragmentation of the data that may be implemented by the TOE. The endianness or byte-order should also be taken into account in the search).

Step #6 ensures that partial key fragments do not remain in non-volatile memory. If the evaluator finds a 32-or-greater-consecutive-bit fragment, then fail immediately. Otherwise, there is a chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is also found in this repeated run then the test fails unless the developer provides a reasonable explanation for the collision, then the evaluator may give a pass on this test.

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

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

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

2.1.1.4.4. KMD

The evaluator shall examine the KMD to verify that it identifies and describes the interfaces that are used to service commands to read/write memory. The evaluator shall examine the interface description for each different media type to ensure that the interface supports the selections made by the ST author.

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

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

2.1.2. Cryptographic Key Management (Extended - FCS_CKM_EXT)

2.1.2.1. FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction Timing

2.1.2.1.1. 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 this data should be expected to be destroyed.

2.1.2.1.2. AGD

There are no guidance evaluation activities for this component.

2.1.2.1.3. Test

There are no test evaluation activities for this component.

2.1.2.1.4. KMD

The evaluator shall verify that the KMD includes a description of the areas where keys and key material reside and when this data is no longer needed.

The evaluator shall verify that the KMD includes a key lifecycle that includes a description where

key materials reside, how the key materials are used, how it is determined that keys and key material are no longer needed, and how the data is destroyed once it is no longer needed. The evaluator shall also verify that all key destruction operations are performed in a manner specified by FCS_CKM.4.

2.1.3. Cryptographic Operation (FCS_COP)

2.1.3.1. FCS_COP.1/Hash Cryptographic Operation (Hashing)

2.1.3.1.1. 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. The evaluator shall also check that the TSS identifies whether the implementation is bit-oriented or byte-oriented.

2.1.3.1.2. AGD

The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present. The evaluator also checks the AGD documents to confirm that the instructions for establishing the evaluated configuration use only those hash algorithms selected in the ST.

2.1.3.1.3. 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.

SHA-1 and SHA-2 Tests

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

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

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

Short Messages Test, Bit-oriented Implementation

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

messages are provided to the TSF.

Short Messages Test, Byte-oriented Implementation

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

Selected Long Messages Test, Bit-oriented Implementation

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

Selected Long Messages Test, Byte-oriented Implementation

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

Pseudo-randomly Generated Messages Test

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

SHA-3 Tests

The tests below are derived from the The Secure Hash Algorithm-3 Validation System (SHA3VS), Updated: April 7, 2016, from the National Institute of Standards and Technology.

For each SHA-3-XXX implementation, XXX represents d, the digest length in bits. The capacity, c, is equal to 2d bits. The rate is equal to 1600-c bits.

The TSF hashing functions can be implemented with one of two orientations. The first is a bit-oriented mode that hashes messages of arbitrary length. The second is a byte-oriented mode that 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). Separate tests for each orientation are given below.

The evaluator shall perform all of the following tests for each hash algorithm and orientation implemented by the TSF and used to satisfy the requirements of this PP. The evaluator shall compare digest values produced by a known-good SHA-3 implementation against those generated

by running the same values through the TSF.

Short Messages Test, Bit-oriented Mode

The evaluators devise an input set consisting of rate+1 short messages. The length of the messages ranges sequentially from 0 to rate bits. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Short Messages Test, Byte-oriented Mode

The evaluators devise an input set consisting of rate/8+1 short messages. The length of the messages ranges sequentially from 0 to rate/8 bytes, with each message being an integral number of bytes. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Selected Long Messages Test, Bit-oriented Mode

The evaluators devise an input set consisting of 100 long messages ranging in size from rate+(rate+1) to rate+(100*(rate+1)), incrementing by rate+1. (For example, SHA-3-256 has a rate of 1088 bits. Therefore, 100 messages will be generated with lengths 2177, 3266, …, 109988 bits.) The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test, Byte-oriented Mode

The evaluators devise an input set consisting of 100 messages ranging in size from (rate+(rate+8)) to (rate+100*(rate+8)), incrementing by rate+8. (For example, SHA-3-256 has a rate of 1088 bits. Therefore 100 messages will be generated of lengths 2184, 3280, 4376, …, 110688 bits.) The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudo-randomly Generated Messages (Monte Carlo) Test, Byte-oriented Mode

The evaluators supply a seed of d bits (where d is the length of the message digest produced by the hash function to be tested. This seed is used by a pseudorandom function to generate 100,000 message digests. One hundred of the digests (every 1000th digest) are recorded as checkpoints. The TOE then uses the same procedure to generate the same 100,000 message digests and 100 checkpoint values. The evaluators then compare the results generated ensure that the correct result is produced when the messages are generated by the TSF.

2.1.3.1.4. KMD

There are no KMD evaluation activities for this component.

2.1.3.2. FCS_COP.1/HMAC Cryptographic Operation (Keyed Hash)

2.1.3.2.1. TSS

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

2.1.3.2.2. AGD

There are no guidance evaluation activities for this component.

2.1.3.2.3. Test

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

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

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

2.1.3.2.4. KMD

There are no KMD evaluation activities for this component.

2.1.3.3. FCS_COP.1/KAT Cryptographic Operation (Key Agreement/Transport)

2.1.3.3.1. TSS

The evaluator shall ensure that the selected RSA and ECDH key agreement/transport schemes correspond to the key generation schemes selected in FCS_CKM.1/AK, and the key establishment schemes selected in FCS_CKM.2 If the ST selects DH, the TSS shall describe how the implementation meets the relevant sections of RFC 3526 (Section 3-7) and RFC 7919 (Appendices A.1-A.5). If the ST selects ECIES, the TSS shall describe the key sizes and algorithms (e.g. elliptic curve point multiplication, ECDH with either NIST or Brainpool curves, AES in a mode permitted by FCS_COP.1/SKC, a SHA-2 hash algorithm permitted by FCS_COP.1/Hash, and a MAC algorithm permitted by FCS_COP.1/HMAC) that are supported for the ECIES implementation.

The evaluator shall ensure that, for each key agreement/transport scheme, the size of the derived keying material is at least the same as the intended strength of the key agreement/transport scheme, and where feasible this should be twice the intended security strength of the key agreement/transport scheme.

Table 2 of NIST SP 800-57 identifies the key strengths for the different algorithms that can be used for the various key agreement/transport schemes.

2.1.3.3.2. AGD

There are no guidance evaluation activities for this component.

2.1.3.3.3. 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.

The evaluator shall verify the implementation of the key generation routines of the supported schemes using the following tests:

If ECDH-NIST or ECDH-BPC is claimed:

SP800-56A Key Agreement Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role-key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static or ephemeral), the MAC tags, and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, The evaluator shall also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

If KAS1, KAS2, KTS-OAEP, or RSAES-PKCS1-v1_5 is claimed:

SP800-56B and PKCS#1 Key Establishment Schemes

If the TOE acts as a sender, the following evaluation activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following evaluation activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTSOAEP is supported), the tester shall generate 10 sets of test vectors.

Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plain text keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

DH:

The evaluator shall verify the correctness of each TSF implementation of each supported Diffie-Hellman group by comparison with a known good implementation.

Curve25519:

The evaluator shall verify a TOE's implementation of the key agreement scheme using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specification. These components include the calculation of the shared secret K and the hash of K.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement role and hash function combination, the tester shall generate 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the shared secret value K, and the hash of K. The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value K and compare the hash generated from this value.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results. To conduct this test, the evaluator generates a set of 30 test vectors consisting of data sets including the evaluator's public keys and the TOE's public/private key pairs.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value K or the hash of K. At least two of the test vectors shall remain unmodified and therefore should result

in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

ECIES:

The evaluator shall verify the correctness of each TSF implementation of each supported use of ECIES by comparison with a known good implementation.

2.1.3.3.4. KMD

There are no KMD evaluation activities for this component.

2.1.3.4. FCS_COP.1/KeyEnc Cryptographic Operation (Key Encryption)

2.1.3.4.1. TSS

The evaluator shall examine the TSS to ensure that it identifies whether the implementation of this cryptographic operation for key encryption (including key lengths and modes) is an implementation that is tested in FCS_COP.1/SKC.

The evaluator shall check that the TSS includes a description of the key wrap functions and shall check that this uses a key wrap algorithm and key sizes according to the specification selected in the ST out of the table as provided in the [DSC cPP] table.

2.1.3.4.2. AGD

The evaluator checks the AGD documents to confirm that the instructions for establishing the evaluated configuration use only those key wrap functions selected in the ST. If multiple key access modes are supported, the evaluator shall examine the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

2.1.3.4.3. Test

Refer to FCS_COP.1/SKC for the required testing for each symmetric key wrapping method selected from the table and to FCS_COP.1/KAT for the required testing for each asymmetric key wrapping method selected from the table. Each distinct implementation shall be tested separately.

If the implementation of the key encryption operation is the same implementation tested under FCS_COP.1/SKC or FCS_COP.1/KAT, and it has been tested with the same key lengths and modes, then no further testing is required. If key encryption uses a different implementation, (where "different implementation" includes the use of different key lengths or modes), then the evaluator shall additionally test the key encryption implementation using the corresponding tests specified for FCS_COP.1/SKC or FCS_COP.1/KAT.

2.1.3.4.4. KMD

The evaluator shall examine the KMD to ensure that it describes when the key wrapping occurs, that the KMD description is consistent with the description in the TSS, and that for all keys that are

wrapped the TOE uses a method as described in the [DSC cPP] table. No uncertainty should be left over which is the wrapping key and the key to be wrapped and where the wrapping key potentially comes from i.e. is derived from.

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

2.1.3.5. FCS_COP.1/SigGen Cryptographic Operation (Signature Generation)

2.1.3.5.1. TSS

The evaluator shall examine the TSS to ensure that all signature generation functions use the approved algorithms and key sizes.

2.1.3.5.2. AGD

There are no AGD evaluation activities for this component.

2.1.3.5.3. 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.

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

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

If SigGen1: RSASSA-PKCS1-v1_5or SigGen4: RSASSA-PSS is claimed:

The below test is derived from The 186-4 RSA Validation System (RSA2VS). Updated 8 July 2014, Section 6.3, from the National Institute of Standards and Technology.

To test the implementation of RSA signature generation the evaluator uses the system under test to generate signatures for 10 messages for each combination of modulus size and SHA algorithm. The evaluator then uses a known-good implementation and the associated public keys to verify the signatures.

If SigGen2: Digital Signature Scheme 2 (DSS2) or SigGen3: Digital Signature Scheme 3 (DSS3):

To test the implementation of DSS2/3 signature generation the evaluator uses the system under test to generate signatures for 10 messages for each combination of SHA algorithm, hash size and key size. The evaluator them uses a known-good implementation and the associated public keys to verify the signatures.

If SigGen5: ECDSA is claimed:

The below test is derived from The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS). Updated 18 March 2014, Section 6.4, from the National Institute of Standards and Technology.

To test the implementation of ECDSA signature generation the evaluator uses the system under test to generate signatures for 10 messages for each combination of curve, SHA algorithm, hash size, and key size. The evaluator then uses a known-good implementation and the associated public keys to verify the signatures.

2.1.3.5.4. KMD

There are no KMD evaluation activities for this component.

2.1.3.6. FCS_COP.1/SigVer Cryptographic Operation (Signature Verification)

2.1.3.6.1. 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" rather than "memory location 0x00007A4B") of the data to be used in verifying the digital signature; how the data received from the operational environment are brought onto the device; and any processing that is performed that is not part of the digital signature algorithm (for instance, checking of certificate revocation lists).

2.1.3.6.2. AGD

There are no AGD evaluation activities for this component.

2.1.3.6.3. 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.

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

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

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

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

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

signature verification, and all the signatures made using altered parameters result in unsuccessful signature verification.

SigVer5: ECDSA on NIST and Brainpool Curves

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

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

SigVer2: Digital Signature Scheme 2

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

For each supported modulus size, underlying hash algorithm, and length of the trailer field (1- or 2-byte), the evaluator shall generate NT sets of recoverable message (M1), non-recoverable message (M2), salt, public key and signature (Σ).

1. N_T shall be greater than or equal to 20.
2. The length of salts shall be selected from its supported length range of salt. The typical length of salt is equal to the output block length of underlying hash algorithm (see 9.2.2 of ISO/IEC 9796-2:2010).
3. The length of recoverable messages should be selected by considering modulus size, output block length of underlying hash algorithm, and length of salt (L_S). As described in Annex D of ISO/IEC 9796-2:2010, it is desirable to maximise the length of recoverable message. As Table 3 shows the maximum bit-length of recoverable message that is divisible by 512, for some combinations of modulus size, underlying hash algorithm, and length of salt.

Table 3. Maximum length of recoverable message divisible by 512

Maximum length of recoverable message divisible by 512 (bits)	**Modulus size (bits)**	**Underlying hash algorithm (bits)**	**Length of salt L_S (bits)**
1536	2048	SHA-256	128
1024			256
1024		SHA-512	128
1024			256
512			512

Maximum length of recoverable message divisible by 512 (bits)	Modulus size (bits)	Underlying hash algorithm (bits)	Length of salt L_S (bits)
2560	3072	SHA-256	128
2048			256
2048		SHA-512	128
2048			256
1536			512
None that 2-byte trailer field is assumed in calculating the maximum length of recoverable message			

4. The length of non-recoverable messages should be selected by considering the underlying hash algorithm and usages. If the TSF is used for verifying the authenticity of software/firmware updates, the length of non-recoverable messages should be selected greater than or equal to 2048-bit. With this length range, it means that the underlying hash algorithm is also tested for two or more input blocks.
5. The evaluator shall select approximately one half of N_T sets and shall alter one of the values (non-recoverable message, public key exponent or signature) in the sets. In altering public key exponent, the evaluator shall alter the public key exponent while keeping the exponent odd. In altering signatures, the following ways should be considered:
   i. Altering a signature just by replacing a bit in the bit-string representation of the signature
   ii. Altering a signature so that the trailer in the message representative cannot be interpreted. This can be achieved by following ways:
      - Setting the rightmost four bits of the message representative to the values other than '1100'.
      - In the case when 1-byte trailer is used, setting the rightmost byte of the message representative to the values other than '0xbc', while keeping the rightmost four bits to '1100'.
      - In the case when 2-byte trailer is used, setting the rightmost byte of the message representative to the values other than '0xcc', while keeping the rightmost four bits to '1100'.
   iii. In the case when 2-byte trailer is used, altering a signature so that the hash algorithm identifier in the trailer (i.e. the left most byte of the trailer) does not correspond to hash algorithms identified in the SFR. The hash algorithm identifiers are 0x34 for SHA-256 (see Clause 10 of ISO/IEC 10118-3:2018), and 0x35 for SHA-512 (see Clause 11 of ISO/IEC 10118-3:2018).
   iv. Let L_S be the length of salt, altering a signature so that the intermediate bit string D in the message representative is set to all zeroes except for the rightmost L_S bits of D.
   v. (non-conformant signature length) Altering a signature so that the length of signature Σ is changed to modulus size and the most significant bit of signature Σ is set equal to '1'.
   vi. (non-conformant signature) Altering a signature so that the integer converted from signature Σ is greater than modulus n.

The evaluator shall supply the NT sets to the TSF and obtain in response a set of NT Verification-Success or Verification-Fail values. When the Verification-Success is obtained, the evaluator shall also obtain recovered message (M 1*).

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

For each recovered message, the evaluator shall compare the recovered message (M1*) with the corresponding recoverable message (M 1) in the unaltered sets.

The test passes only if all the signatures made using unaltered sets result in Verification-Success, each recovered message (M 1*) is equal to corresponding M 1 in the unaltered sets, and all the signatures made using altered sets result in Verification-Fail.

SigVer3: Digital Signature Scheme 3

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

2.1.3.6.4. KMD

There are no KMD evaluation activities for this component.

2.1.3.7. FCS_COP.1/SKC Cryptographic Operation (Symmetric Key Cryptography)

2.1.3.7.1. TSS

The evaluator shall check that the TSS includes a description of encryption functions used for symmetric key encryption. The evaluator should check that this description of the selected encryption function includes the key sizes and modes of operations as specified in the [DSC cPP] "Table 9. Supported Methods for Symmetric Key Cryptography Operation."

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

2.1.3.7.2. AGD

If the product supports multiple modes, the evaluator shall examine the vendor’s documentation to determine that the method of choosing a specific mode/key size by the end user is described.

2.1.3.7.3. 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.

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test

environment and TOE execution environment shall be described.

Preconditions for testing:

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

AES-CBC:

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

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

AES-CBC Known Answer Tests

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

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

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

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

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

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

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

KAT-4 (Variable Text): To test the encrypt functionality of AES-CBC, for each selected key size, the

evaluator shall supply a set of 128-bit plaintext values (as described below) and obtain the ciphertext values that result from AES-CBC encryption of each plaintext value using a key of each size and IV consisting of all zeros.

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

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

```
# Input: PT, IV, Key
Key[0] = Key
IV[0] = IV
PT[0] = PT
for i = 0 to 99 {
   Output Key[i], IV[i], PT[0]
   for j = 0 to 999 {
       if (j == 0) {
           CT[j] = AES-CBC-Encrypt(Key[i], IV[i], PT[j])
           PT[j+1] = IV[i]
       } else {
           CT[j] = AES-CBC-Encrypt(Key[i], PT[j])
           PT[j+1] = CT[j-1]
       }
   }
   Output CT[j]
```

```
    If (KeySize == 128) Key[i+1] = Key[i] xor CT[j]
    If (KeySize == 192) Key[i+1] = Key[i] xor (last 64 bits of CT[j-1] ||CT[j])
    If (KeySize == 256) Key[i+1] = Key[i] xor ((CT[j-1] |CT[j])
    IV[i+1] = CT[j]
    PT[0] = CT[j-1]
}
```

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

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

AES-CCM:

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

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

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

- **Keys**: All supported and selected key sizes (e.g., 128, 192, or 256 bits).
- **Associated Data**: Two or three values for associated data length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.
- **Payload**: Two values for payload length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported payload lengths.
- **Nonces**: All supported nonce lengths (e.g., 8, 9, 10, 11, 12, 13) in bytes.
- **Tag**: All supported tag lengths (e.g., 4, 6, 8, 10, 12, 14, 16) in bytes.

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

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

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

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

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

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

AES-GCM:

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

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

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

- **Key size in bits:** Each selected and supported key size (e.g., 128, 192, or 256 bits).
- **Plaintext length in bits**: Up to four values for plaintext length: Two values that are non-zero integer multiples of 128, if supported. And two values that are non-multiples of 128, if supported.
- **AAD length in bits:** Up to five values for AAD length: Zero-length, if supported. Two values that are non-zero integer multiples of 128, if supported. And two values that are integer non-multiples of 128, if supported.
- **IV length in bits**: Up to three values for IV length: 96 bits. Minimum and maximum supported lengths, if different.
- **MAC length in bits:** Each supported length (e.g., 128, 120, 112, 104, 96).

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

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

implementation.

AES-CTR:

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

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

AES-CTR Known Answer Tests

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

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

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

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

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

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

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

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

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

To test the decrypt functionality of AES-CTR, for each selected key size, use the plaintext values

from above as ciphertext input, and AES-CTR decrypt each ciphertext value using key of each size consisting of all zeros.

AES-CTR Multi-Block Message Test

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

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

AES-CTR Monte Carlo Tests

The evaluator shall test the encrypt functionality for each selected key size using 100 2-tuples of pseudo-random values for plaintext and keys.

The evaluator shall supply a single 2-tuple of pseudo-random values for each selected key size. This 2-tuple of plaintext and key is provided as input to the below algorithm to generate the remaining 99 2-tuples, and to run each 2-tuple through 1000 iterations of AES-CTR encryption.

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

The ciphertext computed in the 1000th iteration (CT[999]) is the result for each of the 100 2-tuples for each selected key size. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

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

Note additional design considerations for this mode are addressed in the KMD requirements.

XTS-AES:

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

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

The evaluator shall generate test values as follows:

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

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

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

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

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.

The evaluator shall check that the full length keys are created by methods that ensure that the two halves are different and independent.

AES-KWP:

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

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

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

The evaluator shall test the authenticated-decryption (KWP-AD) functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext

values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption. For the Authenticated Decryption test, 20 out of the 100 trials per plaintext length have ciphertext values that fail authentication.

Additionally, the evaluator shall perform the following negative tests:

Test 1: (invalid plaintext length):

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

Test 2: (invalid ciphertext length):

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

Test 3: (invalid ICV2):

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

Test 4: (invalid padding length):

Generate one ciphertext using algorithm KWP-AE with substring $_{32}$ of S replaced by each of the following 32-bit values, where len(P) is the length of P in bits and $[]_{32}$ denotes the representation of an integer in 32 bits:

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

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

Test 5: (invalid padding bits):

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

for each PAD length [1..7]

for each byte in PAD set a zero PAD value;

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

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

AES-KW:

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

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

- Supported key lengths selected in the ST (e.g. 128 bits, 256 bits)
- Five plaintext lengths:
  - Two lengths that are non-zero multiples of 128 bits (two semi-block lengths)
  - Two lengths that are odd multiples of the semi-block length (64 bits)
  - The largest supported plaintext length less than or equal to 4096 bits.

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

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

Additionally, the evaluator shall perform the following negative tests:

Test 1 (invalid plaintext length):

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

Test 2 (invalid ciphertext length):

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

Test 3 (invalid ICV1):

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

CAM-CBC:

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

CAM-CCM:

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

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

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

CAM-GCM:

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

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

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

XTS-CAM:

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

The evaluator shall generate test values as follows:

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

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

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

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

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

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

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

2.1.3.7.4. KMD

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

Assessment of the complete data path for symmetric key encryption includes confirming that the KMD describes the data flow from the device's host interface to the device's non-volatile memory storing the data, and gives information enabling the user data datapath to be distinguished from those situations in which data bypasses the data encryption engine (e.g. read-write operations to an unencrypted Master Boot Record area). The evaluator shall ensure that the documentation of the data path is detailed enough that it thoroughly describes the parts of the TOE that the data passes through (e.g. different memory types, processors and co-processors), its encryption state (i.e. encrypted or unencrypted) in each part, and any places where the data is stored. For example, any caching or buffering of the data should be identified and distinguished from the final destination in non-volatile memory (the latter represents the location from which the host will expect to retrieve the data in future).

If support for AES-CTR is claimed and the counter value source is internal to the TOE, the evaluator shall verify that the KMD describes the internal counter mechanism used to ensure that it provides unique counter block values.

2.1.4. Random Bit Generation (Extended - FCS_RBG_EXT)

2.1.4.1. FCS_RBG_EXT.1 Random Bit Generation

In addition to the materials below, documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [DSC cPP].

2.1.4.1.1. TSS

The evaluator shall examine the TSS to determine that it specifies the DRBG type, identifies the entropy sources seeding the DRBG, and state the assumed or calculated min-entropy supplied either separately by each source or the min-entropy contained in the combined seed value.

2.1.4.1.2. AGD

There are no AGD evaluation activities for this component.

2.1.4.1.3. 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.

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

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.

2.1.4.1.4. KMD

There are no KMD evaluation activities for this component.

2.1.5. Cryptographic Salt Generation (Extended - FCS_SLT_EXT)

2.1.5.1. FCS_SLT_EXT.1 Cryptographic Salt Generation

2.1.5.1.1. TSS

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

2.1.5.1.2. AGD

There are no AGD evaluation activities for this component.

2.1.5.1.3. Test

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

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

2.1.5.1.4. KMD

There are no KMD evaluation activities for this component.

2.1.6. Cryptographic Key Storage (Extended - FCS_STG_EXT)

2.1.6.1. FCS_STG_EXT.1 Protected Storage

2.1.6.1.1. TSS

The evaluator shall review the TSS to determine that the TOE implements the required protected storage. The evaluator shall ensure that the TSS contains a description of the protected storage mechanism that justifies the selection of mutable hardware-based or software-based.

2.1.6.1.2. AGD

The evaluator shall examine the operational guidance to ensure that it describes the process for generating keys, importing keys, or both, based on what is claimed by the ST. The evaluator shall also examine the operational guidance to ensure that it describes the process for destroying keys that have been imported or generated.

2.1.6.1.3. Test

The evaluator shall test the functionality of each security function as described below. If the TOE supports both import and generation of keys, the evaluator shall repeat the testing as needed to

demonstrate that the keys resulting from both operations are treated in the same manner. The devices used with the tooling may need to be non-production devices in order to enable the execution and gathering of evidence.

Test 1: The evaluator shall import or generate keys/secrets of each supported type according to the operational guidance. The evaluator shall write, or the developer shall provide access to, an application that generates a key/secret of each supported type and calls the import functions. The evaluator shall verify that no errors occur during import.

Test 2: The evaluator shall write, or the developer shall provide access to, an application that uses a generated or imported key/secret:

- For RSA, the secret shall be used to sign data.
- For ECDSA, the secret shall be used to sign data.

The evaluator shall repeat this test with the application-imported or application-generated keys/secrets and a different application's imported keys/secrets or generated keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to use the key/secret imported or generated by the user or by a different application:

- The evaluator shall deny the approvals to verify that the application is not able to use the key/secret as described.
- The evaluator shall repeat the test, allowing the approvals to verify that the application is able to use the key/secret as described.

If the ST author has selected common application developer, this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

Test 3: The evaluator shall destroy keys/secrets of each supported type according to the operational guidance. The evaluator shall write, or the developer shall provide access to, an application that destroys an imported or generated key/secret. The evaluator shall repeat this test with the application-imported or application-generated keys/secrets and a different application's imported or generated keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to destroy the key/secret imported by the administrator or by a different application:

- The evaluator shall deny the approvals and verify that the application is still able to use the key/secret as described.
- The evaluator shall repeat the test, allowing the approvals and verifying that the application is no longer able to use the key/secret as described.

If the ST author has selected common application developer, this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

2.1.6.1.4. KMD

There are no KMD evaluation activities for this component.

2.1.6.2. FCS_STG_EXT.2 Key Storage Encryption

2.1.6.2.1. TSS

The evaluator shall review the TSS to determine that the TSS describes the protection of symmetric keys, KEKs, long-term trusted channel key material, and software-based key storage as claimed in FCS_STG_EXT.2.1.

2.1.6.2.2. AGD

There are no AGD evaluation activities for this component.

2.1.6.2.3. Test

There are no test evaluation activities for this component.

2.1.6.2.4. KMD

There are no KMD evaluation activities for this component.

2.1.6.3. FCS_STG_EXT.3 Key Integrity Protection

2.1.6.3.1. TSS

The evaluator shall examine the TSS and ensure that it contains a description of how the TOE protects the integrity of its keys.

2.1.6.3.2. AGD

There are no AGD evaluation activities for this component.

2.1.6.3.3. Test

There are no test evaluation activities for this component.

2.1.6.3.4. KMD

There are no KMD evaluation activities for this component.

2.2. User Data Protection (FDP)

2.2.1. Access Control Policy (FDP_ACC)

2.2.1.1. FDP_ACC.1 Subset Access Control

2.2.1.1.1. TSS

The evaluator shall confirm that the TSS contain the access control policy implemented by the TOE. I.e., the ST author lists each object and identifies for each object, which operations the TSF permits for each subject (i.e. what can "admins" do vs "users").

2.2.1.1.2. AGD

There are no guidance evaluation activities for this component.

2.2.1.1.3. Test

Testing for FDP_ACF includes testing this component.

2.2.1.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.2. Access Control Functions (FDP_ACF)

2.2.2.1. FDP_ACF.1 Security Attribute Based Access Control

2.2.2.1.1. TSS

The evaluator shall examine the TSS to verify that it describes the policy rules for the Access Control SFP. Specifically, the evaluator should be able to identify, for any arbitrary subject-object-operation pairing, which of the following is true:

a. The subject can always perform the desired operation.
b. The subject can never perform the desired operation, either because they lack sufficient permission or because the TSF includes no interface to support the operation.
c. The subject can only perform the desired operation under certain conditions (which the evaluator shall verify are described in the TSS). For example, "the S.CA subject may only perform the OP.Destroy operation on an OB.SDO object if it was the subject that originally created or imported the SDO."
d. The subject can only perform the desired operation on one or more attributes of the object as opposed to the entire object itself (which the evaluator shall verify are identified in the TSS).
e. Whether the subject can perform the desired operation depends on TSF configuration (which the evaluator shall verify is described in the TSS as part of the evaluation of FMT_SMF.1).
f. Some combination of c, d, and e.

Given that this SFR requires a large number of potential subject-object-operation pairings to be identified, it is not the expectation that the TSS contain an exhaustive list of these pairings. It is possible that large numbers of pairings are addressed by blanket statements of policy rules, such as "the subjects S.DSC and S.CA are never able to perform any operation on the OB.AntiReplay object." For any rules that are not addressed in this manner, the evaluator shall verify the TSS includes sufficient data for the evaluator to determine how the TSF will evaluate the action. This can be presented in the form of a table, flowchart, list, or other manner that the ST author finds suitable.

Note that the DSC developer may not use the same terminology for its subjects, objects, and operations as the PP. If this is the case, the evaluator shall verify that the TSS includes a mapping that unambiguously shows how the vendor’s preferred terminology corresponds to what the PP defines.

2.2.2.1.2. AGD

For any access control policy enforcement behavior that is configurable, the evaluator shall ensure that the operational guidance describes how to perform the configuration, including any restrictions on permissible configurable settings.

2.2.2.1.3. Test

The following testing may require the TOE developer to make a test harness available to the evaluator that allows the evaluator to interface directly with the DSC. Due to the large volume of potential testing that this requires, this test may require the use of an automated script. If a test script is made available, the evaluator shall verify that it includes sufficient detail to validate the claims made in the TSS.

For each subject/object/operation/attribute combination, the evaluator shall attempt to perform the operation or determine that no interface is present to attempt the operation, consistent with the limitations described in the TSS.

For each case where an operation is always permitted or never permitted, both positive and negative testing will be conducted implicitly by attempting the operation with all possible subjects and determining that the intended results occur in each case.

For each case where the operation succeeds or fails based on the target object attribute, the evaluator shall ensure that both positive and negative testing is performed such that only the correct target attributes can be operated upon.

For each case where the operation succeeds or fails based on one or more specific conditions, the evaluator shall ensure that both positive and negative testing is performed such that the presence of the conditions causes the test to succeed while the absence of the conditions causes the test to fail.

For each case where the operation succeeds or fails based on an administratively configured setting, the evaluator shall ensure that both positive and negative testing is performed such that the configuration setting can be shown to affect whether or not the operation succeeds.

2.2.2.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.3. Export from the TOE (Extended - FDP_ETC_EXT)

2.2.3.1. FDP_ETC_EXT.2 Propagation of SDOs

2.2.3.1.1. TSS

The evaluator shall examine the TSS to ensure that it describes how it protects the SDO references, authorization data, against access from unauthorized entities. If the TSF is selected, then it should describe how it provides confidentiality of the data while it resides outside the TOE.

2.2.3.1.2. AGD

There are no guidance evaluation activities for this component.

2.2.3.1.3. Test

There are no test evaluation activities for this component.

2.2.3.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.4. Factory Reset (Extended - FDP_FRS_EXT)

2.2.4.1. FDP_FRS_EXT.1 Factory Reset

2.2.4.1.1. TSS

The evaluator shall examine the TSS to determine that it describes each of the conditions which will lead to a factory reset.

2.2.4.1.2. AGD

The evaluator shall examine the operational guidance to ensure that it describes the ways the administrator can set the conditions to initiate a factory reset.

2.2.4.1.3. Test

The evaluator shall identify all functions that resets the DSC to factory setting. For each function, the evaluator shall identify all methods for authorizing the factory reset. For each function and for each authorization method, the evaluator shall create an SDE or SDO. The evaluator shall then verify the presence of the item just created. The evaluator shall initiate a factory reset using the selected function and authorization method and verify the item no longer exists.

2.2.4.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.5. Import from Outside the TOE (Extended - FDP_ITC_EXT)

2.2.5.1. FDP_ITC_EXT.1 Parsing of SDEs

2.2.5.1.1. TSS

The evaluator shall confirm the TSS contains descriptions of the supported methods the TSF uses to import SDEs into the TOE. For each import method selected, the TSS shall describe integrity verification schemes employed. The TSS shall also list the ways the TSF generates and binds security attributes to the SDEs.

2.2.5.1.2. AGD

There are no AGD evaluation activities for this component.

2.2.5.1.3. Test

For each supported import method selected in FDP_ITC_EXT.1.1 and for each supported integrity verification method selected in FDP_ITC_EXT.1.2. used by the selected import method, provide one SDE with valid integrity credentials, one with invalid integrity credentials (e.g. hash). The operations with invalid integrity credentials must result in error. The operations with valid integrity credentials must return an SDO with valid security attributes in accordance with FDP_ITC_EXT.1.4.

2.2.5.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.5.2. FDP_ITC_EXT.2 Parsing of SDOs

2.2.5.2.1. TSS

The evaluator shall confirm the TSS contains descriptions of the supported methods the TSF uses to import SDOs into the TOE. For each import method selected, the TSS shall describe integrity verification schemes employed. The TSS shall also list the ways the TSF generates and binds security attributes to the SDOs.

2.2.5.2.2. AGD

There are no AGD evaluation activities for this component.

2.2.5.2.3. Test

For each supported import method selected in FDP_ITC_EXT.2.1 and for each supported integrity verification method selected in FDP_ITC_EXT.2.2 used by the selected import method, provide one SDO with valid integrity credentials, one with invalid integrity credentials (e.g. hash). The operations with invalid integrity credentials must result in error. The operations with valid integrity credentials must return an SDO with valid security attributes in accordance with FDP_ITC_EXT.2.3, FDP_ITC_EXT.2.4, and FDP_ITC_EXT.2.5.

2.2.5.2.4. KMD

There are no KMD evaluation activities for this component.

2.2.6. Mutable/Immutable Firmware (Extended - FDP_MFW_EXT)

2.2.6.1. FDP_MFW_EXT.1 Mutable/Immutable Firmware

2.2.6.1.1. TSS

The evaluator shall examine the TSS and ensure that details of which firmware components are considered mutable and which firmware components are considered immutable, as well as how these firmware components can/cannot be modified or altered, are described. For example, DSC firmware components that are stored in ROM would be considered immutable.

2.2.6.1.2. AGD

If the TOE has mutable firmware, the evaluator shall examine the operational guidance to ensure that it describes how to modify the firmware.

2.2.6.1.3. Test

If the TOE has mutable firmware, the evaluator shall perform the activities described in the operational guidance to modify the firmware.

2.2.6.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.7. Residual Information Protection (FDP_RIP)

2.2.7.1. FDP_RIP.1 Subset Residual Information Protection

2.2.7.1.1. TSS

The evaluator shall check to ensure that the TSS describes resource deallocation to the extent that they can determine that no data will be reused when reallocating resources following the destruction of an SDE or SDO. The evaluator shall ensure that this description at a minimum describes how the previous data is destroyed. The evaluator shall also ensure that this destruction method is consistent with FCS_CKM.4.

2.2.7.1.2. AGD

There are no AGD evaluation activities for this component.

2.2.7.1.3. Test

Testing for FCS_CKM.4 is sufficient to address this component.

2.2.7.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.8. Confidentiality of SDEs (Extended - FDP_SDC_EXT)

2.2.8.1. FDP_SDC_EXT.1 Confidentiality of SDEs

2.2.8.1.1. TSS

The evaluator shall examine the TSS to determine that it describes the protection for SDEs and authorization data and the methods of protection (e.g. protected storage, symmetric encryption, key wrapping, etc.).

The evaluator shall also examine the TSS to determine whether the TSF stores this data inside the TOE boundary or in its operational environment. If the TSF stores this data inside the TOE boundary, the evaluator shall ensure that TSF uses one of the listed methods to provide confidentiality. If the data is stored in the TOE's operational environment, the evaluator shall

ensure that the TSF uses key wrapping to provide confidentiality.

The evaluator shall examine the TSS to confirm is sufficiently describes each method used to provide confidentiality for SDEs. The evaluator shall also confirm that the TOE supports all encryption methods selected.

2.2.8.1.2. AGD

There are no AGD evaluation activities for this component.

2.2.8.1.3. Test

If the TOE stores SDEs and authorization data inside the TSF, the evaluator shall ensure that external interfaces cannot extract this data in plaintext.

In this case, use the evaluation activities of the FPT_PHP.3 if protected storage is selected, FCS_COP.1/SK if symmetric encryption using… is selected, and FCS_COP.1/KAT if key wrapping using… is selected.

If the TOE stores authentication data inside the operational environment, the evaluator shall ensure that plaintext data is not visible on the interface between the TOE and the operational environment.

2.2.8.1.4. KMD

There are no KMD evaluation activities for this component.

2.2.9. Stored Data Integrity (FDP_SDI)

2.2.9.1. FDP_SDI.2 Stored Data Integrity Monitoring and Action

2.2.9.1.1. TSS

The evaluator shall confirm that the ST author describes the methods for protecting the integrity of SDOs stored with the TOE, and shall identify the iteration of FCS_COP.1/Hash or FCS_COP.1/HMAC that covers any cryptographic algorithm used. The evaluator shall also confirm that the TSS describes the response upon the detection of an integrity error.

The evaluator shall confirm that the TSS describes the actions the TSF takes when the integrity verification fails for an SDO, including the circumstances that cause a notification to be sent when this occurs.

The evaluator shall confirm that TSS describes how integrity of SDOs is protected in FMT_MSA.3 during initialization, and how the integrity of SDOs are verified during parsing (import) in FDP_ITC_EXT.

2.2.9.1.2. AGD

The evaluator shall examine the operational guidance to verify that it describes the conditions that cause a notification to be sent when an integrity error is detected, and what the contents of the notification are.

2.2.9.1.3. Test

The tests for FDP_ITC_EXT and FMT_MSA.3 shall suffice for this component.

2.2.9.1.4. KMD

There are no KMD evaluation activities for this component.

2.3. Identification and Authentication (FIA)

2.3.1. Authorization Failure Handling (Extended - FIA_AFL_EXT)

2.3.1.1. FIA_AFL_EXT.1 Authorization Failure Handling

2.3.1.1.1. TSS

The evaluator shall examine the TSS to determine that it contains a description for how successive unsuccessful authorization attempts are detected and tracked. The evaluator shall examine the TSS to determine that is contains a description of the actions in the event that the authorization attempt threshold is met or exceeded.

The evaluator shall also examine the TSS to determine that it describes how the failed authorization attempt counter is incremented before the authorization is verified.

The evaluator shall also examine the TSS to determine the behaviour that will occur if there are excessive failed authorization attempts, specifically whether future attempts are prevented for a static or configurable amount of time, future attempts are prevented indefinitely, or a factory reset is triggered.

2.3.1.1.2. AGD

The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of unlocking the SDOs is described for each "action" specified (if that option is chosen).

The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that access to SDOs can be maintained, unless it is made permanently unavailable due to a factory reset.

2.3.1.1.3. Test

The evaluator shall perform the following tests for each method by which the TSF authorizes access the SDOs (e.g. any passwords entered as part of establishing authorization):

Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authorization attempts allowed by the TOE (and, if the time period selection in FIA_AFL_EXT.1.3 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authorization attempts limit is reached, authorization attempts with valid credentials are no longer

successful.

Test 2: After reaching the limit for unsuccessful authorization attempts as in Test 1 above, the evaluator shall proceed as follows. If the action selected in FIA_AFL_EXT.1.3 is included in the ST then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable access results in successful access. If the time period selection in FIA_AFL_EXT.1.3 is included in the ST, then the evaluator shall wait for just less than the time period configured in Test 1 and show that an authorization attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorization attempt using valid credentials results in successful access.

Test 3 [conditional]: If factory reset the TOE wiping out all non-persistent SDOs, as described by FDP_FRS_EXT.2 is selected in FIA_AFL_EXT.1.3, the evaluator shall perform the test required by FDP_FRS_EXT.2 with step 5 replaced with "The evaluator shall initiate a factory reset by deliberately meeting or surpassing the threshold for unsuccessful authorization attempts, depending on whether meets or surpasses is selected in FIA_AFL_EXT.1.3."

2.3.1.1.4. KMD

There are no KMD evaluation activities for this component.

2.3.2. Specification of Secrets (FIA_SOS)

2.3.2.1. FIA_SOS.2 TSF Generation of Secrets

2.3.2.1.1. TSS

The evaluator shall ensure that the TSS describes for each of the TSF functions listed in FIA_SOS.2.2, if the available key space is configurable, and the size (or range) of the key space employed to generate authorization values.

The evaluator shall ensure that the TSS states that the quality metrics provided is based on the assumption of sufficient entropy being provided in accordance with the information given in [DSC cPP] Annex D.

The evaluator shall ensure that the TSS describes the restrictions implemented in order to restrict consecutive authentication attempts. (Authentication throttling)

The evaluator shall ensure that the TSS describes the mechanism used to generate authorization values and documents the quality metric that the mechanism provides. The information provided in the TSS shall demonstrate that:

a. The probability that a random single authentication attempt will be successful is less than one in 1,000,000; and
b. The probability that random multiple authentication attempts during a one (1) minute period will be successful is less than one in 100,000.

2.3.2.1.2. AGD

The evaluator shall examine the guidance documentation to determine that it describes any configuration necessary to enforce the use of TSF generated authorization values listed in FIA_SOS.2.2.

The evaluator shall ensure that the guidance documentation provides any instructions needed to set parameters affecting the available key spaces.

2.3.2.1.3. Test

The evaluator shall perform the following tests.

Test 1: The evaluator shall compose a set of 50 authorization values that meet the requirements, and 50 authorization values that fail to meet the requirements.

a. For each authentication value that meets the requirements, the evaluator shall verify that the TOE supports the authentication value.
b. For each authentication value that does not meet the requirements, the evaluator shall verify that the TOE does not support the authentication value.

While the evaluator is not required (nor is it feasible) to test all possible compositions of authentication values, the evaluator shall ensure that the key space identified in the TSS is valid.

Test 2: For each TSF function listed in FIA_SOS.2.2 the TOE shall be configured to generate the authentication values; the evaluator shall check that the TOE produces the authentication values.

2.3.2.1.4. KMD

There are no KMD evaluation activities for this component.

2.3.3. User Authentication (FIA_UAU)

2.3.3.1. FIA_UAU.2 User Authentication before Any Action

2.3.3.1.1. TSS

The evaluator shall examine the TSS to determine that it describes the identification and authentication process for each supported method (PIN/try-PIN, salted hash, etc.), the circumstances in which each supported method is used, and constitutes "successful authentication."

The evaluator shall examine the TSS to determine that it describes which actions are allowed before user identification and authentication. The evaluator shall also determine that the TSS describes, for each action that does require identification and authentication, the method and circumstances by which the authentication is performed (e.g., as per the application note, the TSF may authenticate a user once rather than each time access to an SDO is attempted; the TSS shall describe when authentication is or is not required in order to perform a TSF-mediated action).

2.3.3.1.2. AGD

The evaluator shall examine the guidance documentation to determine that any necessary

preparatory steps (e.g., establishing valid credential material such as PIN) to logging in are described. For each supported the login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on.

2.3.3.1.3. Test

The evaluator shall use the guidance documentation to configure the appropriate credentials supported for each authentication method. For that authentication method, the evaluator shall attempt to perform TSF-mediated actions that require successful use of that authentication method and subsequently show that providing correct I&A information results in the ability to perform the requested action, while providing incorrect information results in denial of access.

2.3.3.1.4. KMD

There are no KMD evaluation activities for this component.

2.3.3.2. FIA_UAU.5 Multiple Authentication Mechanisms

2.3.3.2.1. TSS

The evaluator shall examine the TSS and ensure that it describes the authentication mechanisms used to support user authentication for the Prove service as well as how each authentication mechanism provides authentication for the Prove service.

2.3.3.2.2. AGD

If the supported authentication mechanisms are configurable, the evaluator shall examine the operational guidance to verify that it describes how to configure the authentication mechanisms used to provide authentication for the Prove service.

2.3.3.2.3. Test

For each supported authentication mechanism, the evaluator shall verify that valid credentials result in successful authentication and invalid credentials result in a rejected authentication attempt. If the supported authentication mechanisms are configurable, the evaluator shall follow the operational guidance to enable/disable the various mechanisms and ensure that valid credentials do not result in successful authentication if that mechanism is disabled, or that there is no interface to provide authentication credentials over an external interface when that mechanism is disabled.

2.3.3.2.4. KMD

There are no KMD evaluation activities for this component.

2.3.3.3. FIA_UAU.6 Re-Authenticating

2.3.3.3.1. TSS

The evaluator shall examine the TSS to determine that it describes each of the options for reauthorization.

2.3.3.3.2. AGD

There are no AGD evaluation activities for this component.

2.3.3.3.3. Test

The evaluator shall use the configuration guidance to create an SDO with each of the options for reauthorization, then identify functions to exercise each of these options, then execute these options providing the correct authorization confirming that the operation succeeded with respect to the reauthorization option chosen. The evaluator shall then attempt to execute these functions while providing the incorrect authorization and confirming that the operation fails.

2.3.3.3.4. KMD

There are no KMD evaluation activities for this component.

2.4. Security Management (FMT)

2.4.1. Management of Functions in TSF (Extended - FMT_MOF_EXT)

2.4.1.1. FMT_MOF_EXT.1 Management of Security Functions Behavior

2.4.1.1.1. TSS

The evaluator shall verify that the TSS describes those management functions that may be performed by the Administrator, to include how the client applications are prevented from accessing, performing, or relaxing the function (if applicable), and how they are prevented from modifying the Administrator configuration. The TSS also describes any functionality that is affected by administrator-configured policy and how. This activity will be performed in conjunction with FMT_SMF_EXT.1.

2.4.1.1.2. AGD

There are no AGD evaluation activities for this component.

2.4.1.1.3. Test

For each management function described in FMT_SMF_EXT.1.1, the evaluator shall perform the function with administrator authorization data and confirm it succeeds, and again with client application authorization data and confirm that it fails.

2.4.1.1.4. KMD

There are no KMD evaluation activities for this component.

2.4.2. Management of Security Attributes (FMT_MSA)

2.4.2.1. FMT_MSA.1 Management of Security Attributes

2.4.2.1.1. TSS

The evaluator shall confirm that the TSS describes the modification constraints for each SDO security attribute.

2.4.2.1.2. AGD

There are no AGD evaluation activities for this component.

2.4.2.1.3. Test

The evaluator shall confirm that the evaluation activities for FDP_ACF.1 contains tests for the OP.Modify operation on objects OB.P_SDO, OB.T_SDO.

2.4.2.1.4. KMD

There are no KMD evaluation activities for this component.

2.4.2.2. FMT_MSA.3 Static Attribute Initialization

2.4.2.2.1. TSS

The evaluator shall confirm that the TSS describes the initialization process for importing and generating SDOs. The TSS shall describe each type of SDO.Type and any additional attributes that are beyond the ones listed. Additionally, list any further restrictions of the allowed values for the minimum list of attributes.

The evaluator shall confirm that the TSS describes the allowed values for each of the attributes.

2.4.2.2.2. AGD

There are no AGD evaluation activities for this component.

2.4.2.2.3. Test

The evaluator shall confirm that the evaluation activities for FDP_ACF.1 contains tests for the OP.Import and OP.Create operations on objects OB.P_SDO, OB.T_SDO.

2.4.2.2.4. KMD

There are no KMD evaluation activities for this component.

2.4.3. Specification of Management Functions (FMT_SMF)

2.4.3.1. FMT_SMF.1 Specification of Management Functions

2.4.3.1.1. TSS

The evaluator shall verify that the TSS describes all management functions.

2.4.3.1.2. AGD

The evaluator shall verify that the AGD describes how the administrator configures the

management functions.

2.4.3.1.3. Test

Testing for this component is performed through evaluation of FMT_MOF_EXT.1.

2.4.3.1.4. KMD

There are no KMD evaluation activities for this component.

2.4.4. Security Management Roles (FMT_SMR)

2.4.4.1. FMT_SMR.2 Restrictions on Security Roles

2.4.4.1.1. TSS

The evaluator shall confirm that the TSS describes the mechanisms by which client applications can exclusively access their own encrypted data and administrators cannot access client application encrypted data. The evaluator shall also confirm the TSS describes the mechanisms that allow only administrators to perform privileged functions.

2.4.4.1.2. AGD

The evaluator shall verify that the AGD describes how the administrator configures the management functions.

2.4.4.1.3. Test

Testing for this component is performed through evaluation of FMT_MOF_EXT.1.

2.4.4.1.4. KMD

There are no KMD evaluation activities for this component.

2.5. Protection of the TSF (FPT)

2.5.1. Fail Secure (FPT_FLS)

2.5.1.1. FPT_FLS.1/FI Failure with Preservation of Secure State (Fault Injection)

2.5.1.1.1. TSS

The evaluator shall examine the TSS to verify that it describes the actions taken when the TOE experiences fault injection and how the DSC preserves a secure state.

The evaluator shall verify that the TSS describes the state of the DSC when the firmware validity checks fail, including the various failure modes assumed.

2.5.1.1.2. AGD

The evaluator shall examine the operational guidance to verify that it describes what actions

should be taken to attempt to resolve the failed state.

2.5.1.1.3. Test

The evaluator shall perform fault injection on the DSC and attempt to extract a known SDO/SDE.

The evaluator shall cause the DSC to parse or generate an SDO/SDE with a known value. The evaluator will then cause the TOE to process the SDO/SDE, possibly multiple times, while injecting faults on the TOE.

If the evaluator is able to acquire the original SDO/SDE or a known result from the DSC processing the SDO/SDE, the test is a 'Fail', otherwise, the test is a 'Pass'.

2.5.1.1.4. KMD

There are no KMD evaluation activities for this component.

2.5.2. Debug Modes (Extended - FPT_MOD_EXT)

2.5.2.1. FPT_MOD_EXT.1 Debug Modes

2.5.2.1.1. TSS

The evaluator shall examine the TSS to ensure it describes the mechanisms the TSF employs to prevent access to debug modes with a brief description of each debug mode supported.

2.5.2.1.2. AGD

There are no AGD evaluation activities for this component.

2.5.2.1.3. Test

The evaluator shall attempt to exercise any single function from each supported debug mode. If the evaluator is able to exercise any function from any of the supported debug modes, the test is a 'Fail', otherwise, the test is a 'Pass'.

2.5.2.1.4. KMD

There are no KMD evaluation activities for this component.

2.5.3. TSF Physical Protection (FPT_PHP)

2.5.3.1. FPT_PHP.3 Resistance to Physical Attack

2.5.3.1.1. TSS

The evaluator shall examine the TSS to ensure it describes the methods used by the TOE to detect physical tampering and how the TOE will respond when physical tampering has been detected.

The evaluator shall also examine the TSS to ensure that it documents the temperature and voltage ranges in which the TSF is assured to operate properly.

2.5.3.1.2. AGD

There are no AGD evaluation activities for this component.

2.5.3.1.3. Test

The evaluator shall perform the following tests:

Test 1: Fault Injection

Refer to the testing for FPT_FLS.1/FI.

Test 2: Temperature and Power Analysis

The following testing is derived from [ISO 24759] test procedures TE07.77.01 through TE07.77.03:

The evaluator shall configure the ambient temperature and voltage close to the approximate extreme of the normal operating ranges specified in the TSS and verify that the TSF continues to function as expected. The evaluator shall determine 'expected functionality' based on how the TSS describes the TOE's reaction to an environmental failure. For example, if the TSS states that the TOE's response is to shut down, it can be assumed that the TOE functions as expected if it does not shut down. If the TSS states that the TOE's response is to zeroize certain data, it can be assumed that the TOE functions as expected if the evaluator performs functions that rely on known data values and obtain results that indicate non-zero values.

The evaluator shall then extend the temperature and voltage outside of the specified normal range and verify that the TOE responds in the manner specified in the ST. If the TOE's response is to zeroize known data, the evaluator shall return the ambient temperature and voltage to a normal range, perform functions that rely on known data values, and observe that the results of these functions are consistent with known values of zero.

2.5.3.1.4. KMD

There are no KMD evaluation activities for this component.

2.5.4. Root of Trust (Extended - FPT_PRO_EXT)

2.5.4.1. FPT_PRO_EXT.1 Root of Trust

2.5.4.1.1. TSS

The evaluator shall ensure that the TSS describes either a pre-installed identity (contained within an SDO), or a process on how the TOE creates an identity. IEEE 802.1ar is one example of a standard which a device can use to create such an identity.

The evaluator shall additionally examine the TSS to ensure that it describes how the Root of Trust is immutable or otherwise mutable if and only if controlled by a unique identifiable owner, the roles this owner assumes in doing so (manufacturer administrator, owner administrator, etc.), as well as the circumstances in which the Root of Trust is mutable.

[conditional] For an immutable Root of Trust, the evaluator shall ensure there are no RoT update

functions.

[conditional] For a mutable Root of Trust, the evaluator shall ensure the Root of Trust update mechanism uses an approved method for authenticating the source of the update.

2.5.4.1.2. AGD

For mutable Root of Trust data, the evaluator shall confirm the AGD contains an approved authenticated method for modifying the Root of Trust identity.

2.5.4.1.3. Test

Immutability

For immutable Root of Trust identity, the evaluator shall confirm a successful evaluation of FPT_PHP.1 (Physical Protection).

Mutability

For a mutable Root of Trust identity, the evaluator shall perform the following tests:

1. Create or use an authenticated Root of Trust identity, confirm the authenticated method for modifying the Root of Trust identity succeeds.
2. Create or use an unauthenticated Root of Trust identity, confirm the target fails to modify the Root of Trust identity.

2.5.4.1.4. KMD

There are no KMD evaluation activities for this component.

2.5.5. Root of Trust Services (Extended - FPT_ROT_EXT)

2.5.5.1. FPT_ROT_EXT.1 Root of Trust Services

2.5.5.1.1. TSS

The evaluator shall ensure that the TSS identifies the Roots of Trust it uses (including but not limited to the Roots of Trust identified in the selections in this requirement) and describes their function in the context of the TOE.

2.5.5.1.2. AGD

There are no AGD evaluation activities for this component.

2.5.5.1.3. Test

Root of Trust for Storage

The evaluator shall confirm a successful evaluation of FCS_CKM.1/KEK, FCS_STG_EXT.1, FCS_STG_EXT.2, FCS_STG_EXT.3, FPT_PHP.3.

Root of Trust for Authorization

The evaluator shall confirm a successful evaluation of FIA_AFL_EXT.1.

Root of Trust for Measurement

The evaluator shall confirm a successful evaluation of FCS_COP.1/Hash.

Root of Trust for Reporting

The evaluator shall confirm a successful evaluation of FCS_COP.1/SigGen.

2.5.5.1.4. KMD

There are no KMD evaluation activities for this component.

2.5.5.2. FPT_ROT_EXT.2 Root of Trust for Storage

2.5.5.2.1. TSS

The evaluator shall ensure that the TSS describes how the Root of Trust for Storage prevents unauthorized access to SDOs. The evaluator shall also examine the TSS to verify that it uses approved mechanisms to protect the integrity of SDOs.

2.5.5.2.2. AGD

There are no AGD evaluation activities for this component.

2.5.5.2.3. Test

Testing for this component is completed through evaluation of FCS_CKM.1/KEK, FCS_STG_EXT.1, FCS_STG_EXT.2, FCS_STG_EXT.3, and FPT_PHP.3.

2.5.5.2.4. KMD

There are no KMD evaluation activities for this component.

2.5.6. Replay Prevention (Extended - FPT_RPL_EXT)

2.5.6.1. FPT_RPL_EXT.1 Replay Prevention

2.5.6.1.1. TSS

The evaluator shall examine the TSS to verify that it describes the mechanism employed for preventing replay of user authorization of operations on SDOs and that access is denied when replay is detected.

2.5.6.1.2. AGD

The evaluator shall examine the operational guidance to verify that it describes how to enforce Replay Prevention if configuration is necessary.

2.5.6.1.3. Test

The evaluator shall perform an authorization of an operation on an SDO and capture or retain that authorization for reuse. The evaluator shall then attempt to replay that same authorization and ensure that the DSC does not allow the authorization to take place. If the replay of the authorization is allowed to take place for an operation on SDOs, the test is a 'Fail', otherwise, the test is a 'Pass'.

2.5.6.1.4. KMD

There are no KMD evaluation activities for this component.

2.5.7. Time Stamps (FPT_STM)

2.5.7.1. FPT_STM.1 Reliable Time Stamps

2.5.7.1.1. TSS

The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time. The TSS provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

2.5.7.1.2. AGD

The evaluator shall examine the guidance documentation to ensure it instructs the administrator how to set the time or indicates any configuration steps required for the TSF to receive time data from an external source.

2.5.7.1.3. Test

The evaluator shall perform the following tests:

Test 1 [conditional]: If the TSF provides a mechanism to manually set the time, the evaluator shall use the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

Test 2 [conditional]: If the TSF receives time data from some source outside the TOE, the evaluator shall use the guidance documentation to configure the external time source (if applicable). The evaluator shall observe that the time has been set to the expected value.

2.5.7.1.4. KMD

There are no KMD evaluation activities for this component.

2.5.8. TSF Self Test (FPT_TST)

2.5.8.1. FPT_TST.1 TSF Testing

2.5.8.1.1. TSS

The evaluator shall examine the TSS and other vendor documentation and ensure they describe the methods used to verify integrity of the TSF and TSF data. The evaluator shall also verify that the TSS

describes how the tests are performed automatically and autonomously (without intervention).

2.5.8.1.2. AGD

The evaluator shall examine the operational guidance to ensure it provides authorized users with the capability to verify the integrity of the TSF and its data.

2.5.8.1.3. Test

Test 1: The evaluator shall verify that the DSC performs an integrity check of all TSF, including data, as well as performing KATs for those functions. The evaluator shall verify failures using malformed known answer test data (for example, unexpected input or output values).

Test 2: The evaluator shall ensure that when an integrity check failure occurs specific to failing KATs and failure to verify the integrity of the TSF, the TOE will prevent any further processing of the current TSF and user data.

2.5.8.1.4. KMD

There are no KMD evaluation activities for this component.

2.6. Resource Utilization (FRU)

2.6.1. Fault Tolerance (FRU_FLT)

2.6.1.1. FRU_FLT.1 Degraded Fault Tolerance

2.6.1.1.1. TSS

The evaluator shall examine the TSS and other vendor documentation and ensure they describe the response and state of TSF data to each type of fault injection into the TOE.

2.6.1.1.2. AGD

There are no AGD evaluation activities for this component.

2.6.1.1.3. Test

The evaluator shall process SDOs/SDEs while applying each type of identified Fault Injection into the TSF. The evaluator will note whether the TSF response is as noted in the TSS and whether the state can be confirmed. If the response and state are as documented, the test is a 'Pass', otherwise, the test is a 'Fail'.

2.6.1.1.4. KMD

There are no KMD evaluation activities for this component.

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

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

Chapter 3. Evaluation Activities for Optional Requirements

3.1. Cryptographic Support (FCS)

3.1.1. Entropy for External IT Entities (Extended - FCS_ENT_EXT)

3.1.1.1. FCS_ENT_EXT.1 Entropy for External IT Entities

3.1.1.1.1. TSS

The evaluator shall verify that the TSS identifies one or more APIs to obtain entropy data from the TOE.

3.1.1.1.2. AGD

There are no AGD evaluation activities for this component.

3.1.1.1.3. Test

The evaluator shall develop and execute or verify and observe the developer tooling which requests entropy data from the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. The evaluator shall also test the entropy data using the Entropy Analysis Report requirements to determine that the entropy is of comparable quality (in terms of min-entropy estimate) to what is documented in the report.

3.1.1.1.4. KMD

There are no KMD evaluation activities for this component.

3.1.2. Random Bit Generation (Extended - FCS_RBG_EXT)

3.1.2.1. FCS_RBG_EXT.2 External Seeding for Random Bit Generation

3.1.2.1.1. TSS

The evaluator shall verify that this function is included as an interface to the RBG in the TSS or the proprietary Entropy Analysis Report and that the behavior of the RBG following a call to this interface is described. The evaluator shall also verify that the documentation of the DRBG describes the conditions of use and possible values for the Personalization String input to the SP 800-90A specified DRBG.

3.1.2.1.2. AGD

The evaluator shall verify that the operational guidance describes the process for supplying an external seed to the TOE's DRBG.

3.1.2.1.3. Test

The evaluator shall develop and execute or verify and observe the developer tooling which adds data to the RBG via the Personalization String. The evaluator shall verify that the request succeeds.

3.1.2.1.4. KMD

There are no KMD evaluation activities for this component.

3.2. Protection of the TSF (FPT)

3.2.1. Internal TOE TSF Data Transfer (FPT_ITT)

3.2.1.1. FPT_ITT.1 Basic Internal TSF Data Transfer Protection

3.2.1.1.1. TSS

The evaluator shall examine the TSS to determine that it describes the TOE in terms of its distributed components and identifies the communications that take place between these components. The evaluator shall also examine the TSS to ensure that it describes the methods by which each of these communications are protected.

3.2.1.1.2. AGD

If applicable, the evaluator shall confirm that the guidance documentation contains instructions for establishing the relevant allowed communication channels between TOE components, and that it contains recovery instructions should a connection be unintentionally broken.

3.2.1.1.3. Test

The evaluator shall perform the following tests:

Test 1: The evaluator shall ensure that all communications between distributed TOE components is tested during the course of the evaluation, setting up the connections as described in the guidance documentation (if applicable) and ensuring that communication is successful.

Test 2: The evaluator shall ensure, for each communication channel, that the channel data is not sent in plaintext or that there is no interface by which it is possible to extract the channel data.

Test 3: The evaluator shall, where possible, physically interrupt communications between TOE components. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected. Note that this test shall only be performed for those interfaces where physical interruption of communications is typical operational behavior. If the TOE consists of multiple components embedded within the same physical chassis such that interrupting communications requires physical destruction of the device (as opposed to disconnecting a re-connectable wire or temporarily disabling a radio), this test is not applicable.

3.2.1.1.4. KMD

There are no KMD evaluation activities for this component.

3.2.2. Root of Trust (Extended - FPT_PRO_EXT)

3.2.2.1. FPT_PRO_EXT.2 Data Integrity Measurements

3.2.2.1.1. TSS

The evaluator shall examine the TSS and ensure that it describes the contents and structure of any quantifiable measures of integrity (including but not limited to cryptographic self-tests) as well as the contents and structure of any reports generated, if any, that are used to quantify the integrity of the data integrity measurements stored in the DSC and establish trust. The evaluator shall also ensure that the TSS describes the order and importance of integrity measurements and records that comprise attestation to prove the integrity of the SDOs in the DSC. In addition, the evaluator shall ensure the TSS gives justification as to how the process in accumulating measurements is consistent between restarts.

3.2.2.1.2. AGD

There are no AGD evaluation activities for this component.

3.2.2.1.3. Test

The tests for FDP_ITC_EXT and FMT_MSA.3 shall suffice for this component.

3.2.2.1.4. KMD

There are no KMD evaluation activities for this component.

3.2.3. Root of Trust Services (Extended - FPT_ROT_EXT)

3.2.3.1. FPT_ROT_EXT.3 Root of Trust for Reporting Mechanisms

3.2.3.1.1. TSS

The evaluator shall examine the TSS and ensure that it describes the cryptographically verifiable identities for the Root of Trust for Reporting and how they are used, verifying that they are not visible outside the subset of DSC scope corresponding to the Roots of Trust.

[conditional] If these cryptographically verifiable identities come in the form of resident keys or aliases, the evaluator shall verify that the possession of such aliases or keys can only be proved indirectly by using them to decrypt a value that has been encrypted with a corresponding public key. The evaluator shall also verify that such keys or aliases are not used for producing digital signatures.

In addition, the evaluator shall examine the TSS and ensure that it describes how these cryptographically verifiable identities are unique between individual DSCs. The evaluator shall also ensure that the TSS describes how the Root of Trust for Reporting reports on the state of the platform and how the values reported satisfy the scope indicated by the report.

3.2.3.1.2. AGD

The evaluator shall examine the guidance documentation to confirm it describes how to enable the

root of trust for reporting mechanism on the TOE, how to generate a report, and how reports are verified as genuine.

3.2.3.1.3. Test

Note that the following test may require the developer to provide access to a test platform that provides the evaluator with tools that are not typically available to end users.

The evaluator shall perform the following tests:

Test 1: The evaluator shall enable the root of trust for reporting mechanism on the TOE and generate a report.

Test 2: The evaluator shall confirm the generated report is valid.

Test 3: The evaluator shall modify the signature of a valid report and confirm the report is invalid.

Test 4: The evaluator shall modify the contents of a valid report and confirm the report is invalid.

3.2.3.1.4. KMD

There are no KMD evaluation activities for this component.

Chapter 4. Evaluation Activities for Selection-Based Requirements

4.1. Cryptographic Support (FCS)

4.1.1. Cryptographic Key Generation (FCS_CKM)

4.1.1.1. FCS_CKM.1/AK Cryptographic Key Generation (Asymmetric Keys)

4.1.1.1.1. TSS

The evaluator shall examine the TSS to verify that it describes how the TOE generates an asymmetric key based on the methods selected from [DSC cPP] Table 13: "Supported Methods for Asymmetric Key Generation". The evaluator shall examine the TSS to verify that it describes how the TOE invokes the methods selected in the ST from the same table. The evaluator shall examine the TSS to verify that it identifies the usage for each row identifier (key type, key size, and list of standards) selected in the ST.

4.1.1.1.2. AGD

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

4.1.1.1.3. 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.

AK1: RSA Key Generation

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

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.

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

These are:

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

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

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.

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

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

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

AK2 & AK3: ECC Key Generation with NIST and Brainpool Curves

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

ECC Key Generation Test

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

Public Key Validation (PKV) Test

For each supported curve, the evaluator shall generate 12 private/public key pairs using the key

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

AK4: DSA Key Generation using 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:

- 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:

- 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:

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

AK5: Curve25519 Key Generation

The evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated as specified in RFC 7748 using an approved random bit generator (RBG) and shall be written in little-endian order (least significant byte first. To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

Note: Assuming the PKV function of the good implementation will (using little-endian order):

- Confirm the private and public keys are 32-byte values
- Confirm the three least significant bits of the first byte of the private key are zero
- Confirm the most significant bit of the last byte is zero
- Confirm the second most significant bit of the last byte is one
- Calculate the expected public key from the private key and confirm it matches the supplied public key

The evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify 5 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.

4.1.1.1.4. KMD

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

- If AK1 is selected, then the KMD describes which methods for generating p and q are used
- How the key is used as part of the key chain/hierarchy.

4.1.1.2. FCS_CKM.1/SK Cryptographic Key Generation (Symmetric Encryption Key)

4.1.1.2.1. TSS

The evaluator shall examine the TSS to verify that it describes how the TOE obtains an SK through direct generation as specified in FCS_RBG_EXT.1, FCS_COP.1/KDF, or FCS_COP.1/PBKDF. The evaluator shall review the TSS to verify that it describes how the ST invokes the functionality described by FCS_RBG_EXT.1 and FCS_COP.1/PBKDF where applicable.

[conditional] If the symmetric key is generated by an RBG, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.

4.1.1.2.2. AGD

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

4.1.1.2.3. Test

For each selected key generation method, the evaluator shall configure the selected generation capability. The evaluator shall use the description of the RBG interface to verify that the TOE requests and receives an amount of RBG output greater than or equal to the requested key size. The evaluator shall perform the tests as described for FCS_COP.1/KDF and FCS_COP.1/PBKDF.

4.1.1.2.4. KMD

The evaluator shall confirm that the KMD describes, as applicable:

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

4.1.2. Cryptographic Key Management (Extended - FCS_CKM_EXT)

4.1.2.1. FCS_CKM_EXT.5 Cryptographic Key Derivation

4.1.2.1.1. TSS

The evaluator shall check that the TSS includes a description of the key derivation functions and shall check that this uses a key derivation algorithm and key sizes according to the specification selected in the ST out of the table as provided in the [DSC cPP] table per row. The evaluator shall confirm that the TSS supports the selected methods.

If KeyDrv5 is selected, the evaluator shall verify that the TSS shows that the total length of the concatenated keys used as input to the KDF is greater than or equal to the length of the output from the KDF.

[conditional] If key combination is used to form a KEK, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR, a KDF, or encryption.

[conditional] If a KDF is used to form a KEK, the evaluator shall ensure that 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.

[conditional] If key concatenation is used to derive KEKs (KeyDrv5), the evaluator shall ensure the TSS includes a description of the randomness extraction step, including the following:

- The description must include how an approved untruncated MAC function is being used for the randomness extraction step and the evaluator must verify the TSS describes that the output length (in bits) of the MAC function is at least as large as the targeted security strength (in bits) of the parameter set employed by the key establishment scheme (see Tables 1-3 of SP 800-56C).
- The description must include how the MAC function being used for the randomness extraction step is related to the PRF used in the key expansion and verify the TSS description includes the correct MAC function:

    - If an HMAC-hash is used in the randomness extraction step, then the same HMAC-hash (with the same hash function hash) is used as the PRF in the key expansion step.
    - If an AES-CMAC (with key length 128, 192, or 256 bits) is used in the randomness extraction step, then AES-CMAC with a 128-bit key is used as the PRF in the key expansion step.
- The description must include the lengths of the salt values being used in the randomness extraction step and the evaluator shall verify the TSS description includes correct salt lengths:
    - If an HMAC-hash is being used as the MAC, the salt length can be any value up to the maximum bit length permitted for input to the hash function hash.
    - If an AES-CMAC is being used as the MAC, the salt length shall be the same length as the AES key (i.e. 128, 192, or 256 bits).

4.1.2.1.2. AGD

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

4.1.2.1.3. 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.

The evaluator shall perform one or more of the following tests to verify the correctness of the key derivation function, depending on the specific functions that are supported:

Preconditions for testing:

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

The following table maps the data fields in the tests below to the notations used in SP 800-108 and SP 800-56C

Data Fields	**Notations**	
	SP 800-108	SP 800-56C
Pseudorandom function	PRF	PRF
Counter length	r	r
Length of output of PRF	h	h
Length of derived keying material	L	L
Length of input values	I_length	I_length
Pseudorandom input values I	K_1 (key derivation key)	Z (shared secret)
Pseudorandom salt values		S
Randomness extraction MAC	n/a	MAC

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

KeyDrv1: Counter Mode Tests:

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

- One or more pseudorandom functions that are supported by the implementation (PRF).
- One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).
- The length (in bits) of the output of the PRF (h).
- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.
- Up to two values of L that are NOT evenly divisible by h.
- Location of the counter relative to fixed input data: before, after, or in the middle.
    - Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.
    - Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.
    - Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.
- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location, value of r, and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it. For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output.

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.

KeyDrv2: Feedback Mode Tests:

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

- One or more pseudorandom functions that are supported by the implementation (PRF).
- The length (in bits) of the output of the PRF (h).
- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.
- Up to two values of L that are NOT evenly divisible by h.

- Whether or not zero-length IVs are supported.
- Whether or not a counter is used, and if so:
  - One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).
  - Location of the counter relative to fixed input data: before, after, or in the middle.
    - Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.
    - Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.
    - Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.
- The length (I_length) of the input values L.

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I and pseudorandom salt values. If the KDF supports zero-length IVs, five of these test vectors will be accompanied by pseudorandom IVs and the other five will use zero-length IVs. If zero-length IVs are not supported, each test vector will be accompanied by an pseudorandom IV. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the *TOE* in order to produce the keying material output. 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.

KeyDrv3: Double Pipeline Iteration Mode Tests:

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

- One or more pseudorandom functions that are supported by the implementation (PRF).
- The length (in bits) of the output of the PRF (h).
- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.
- Up to two values of L that are NOT evenly divisible by h.
- Whether or not a counter is used, and if so:
  - One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).
  - Location of the counter relative to fixed input data: before, after, or in the middle.
    - Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.

    - Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.
    - Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.
- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. 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.

KeyDrv4: Intermediate Keys Method

If the selected algorithm is a hash then the testing of the hash primitive is the only required Evaluation Activity. If the selected algorithm is XOR then no separate primitive testing is necessary.

KeyDrv5: Concatenated Keys Method

The evaluator should confirm that the combined length of the concatenated keys should be at least as long as the keysize of the selected methods. There are no other tests other than for the methods selected for this row performed for KeyDrv1, KeyDrv2, and KeyDrv3.

KeyDrv6: Two Keys Method

The evaluator should confirm that the combined length of the two keys should be at least as long as the keysize of the selected methods. There are no other tests other than for the methods selected for this row from FCD_COP.1/SK.

KeyDrv7: Shared Secret, Salt, Output Length, Fixed Information Method

For each supported selection of PRF, length of shared secret (Z) [selection: 128, 256] bits, length of salt (S) [selection: length of input block of PRF, one-half length of input block of PRF, 0] bits, output length (L) [selection: 128, 256] bits, and length of fixed information (FixedInfo) [selection: length of on input block of PRF, one-half length of input block of PRF, 0] bits, the evaluator shall generate 10 test vectors that include pseudorandom input values for Z, salt values (for non-zero lengths, otherwise, omit) and fixed information (for non-zero lengths, otherwise, omit).

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. 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.

KeyDrv8: Shared Secret, Salt, IV, Output Length, Fixed Information Method

For each supported selection of PRF, length of shared secret (Z), length of salt, length of initialization vector (IV), output length (L), and length of fixed information (FixedInfo), the evaluator shall generate 10 test vectors that include pseudorandom input values for Z, salt values (for non-zero lengths, otherwise, omit), IV (for non-zero lengths, otherwise, use a vector of length equal to length of input block of PRF and fill with zeros), and fixed information (for non-zero lengths, otherwise, omit).

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. 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.

4.1.2.1.4. KMD

The evaluator shall examine the KMD to ensure that:

- The KMD describes the complete key derivation chain and the description must be consistent with the description in the TSS. For all key derivations the TOE must use a method as described in the [DSC cPP] table. There should be no uncertainty about how a key is derived from another in the chain.
- The length of the key derivation key is defined by the PRF. The evaluator should check whether the key derivation key length is consistent with the length provided by the selected PRF.
- If a key is used as an input to several KDFs, each invocation must use a distinct context string. If the output of a KDF execution is used for multiple cryptographic keys, those keys must be disjoint segments of the output.

4.1.3. Cryptographic Operation (FCS_COP)

4.1.3.1. FCS_COP.1/PBKDF Cryptographic Operation (Password-Based Key Derivation Functions)

4.1.3.1.1. TSS

The evaluator shall review the TSS to verify that it contains a description of the PBKDF. The evaluator shall also confirm the ST supports the selected hash function itself. The evaluator shall confirm that the TSS contains a description of how the TOE ensures that the output of the PBKDF is at least the same length as that specified in FCS_CKM.1/SK and for the KeyDrv4, KeyDrv5, or KeyDrv6 in FCS_CKM_EXT.5.

If the ST performs additional conditioning, whitening, or manipulation of the password or passphrase before applying the PBKDF, or to the output of the PBKDF, the evaluator shall ensure that the TSS describes the actions and provides assurance that the TSF does not negatively impact the entropy of the PBKDF output.

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

4.1.3.1.2. AGD

There are no AGD evaluation activities for this component.

4.1.3.1.3. Test

No explicit testing of the formation of the submask from the input password is required.

For the NIST SP 800-132-based conditioning of the passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1/HMAC).

The evaluator shall verify that the iteration count for PBKDFs performed by the TOE comply with NIST SP 800-132 by ensuring that the TSS contains a description of the estimated time required to derive key material from passwords and how the TOE increases the computation time for password-based key derivation (including but not limited to increasing the iteration count).

4.1.3.1.4. KMD

There are no KMD evaluation activities for this component.

4.2. User Data Protection (FDP)

4.2.1. Data Authentication (FDP_DAU)

4.2.1.1. FDP_DAU.1/Prove Basic Data Authentication (for Use with the Prove Service)

4.2.1.1.1. TSS

The evaluator shall examine the TSS and ensure it describes the data that is validity-stamped and where applicable, authenticity-stamped to the level of understanding the DSC has about the data or its origin (from the user providing it to the Prove service). The evaluator shall also ensure that the TSS describes how the evidence of validity or authenticity is generated, including the subjects who perform the verification, and the form the validity or authenticity stamp is represented (i.e. a cryptographic signature, MAC using a symmetric key shared with the receiver, etc.).

4.2.1.1.2. AGD

The evaluator shall ensure that the operational guidance describes how to configure validity-stamping on the TOE.

4.2.1.1.3. Test

The following testing may require the TOE developer to make a test harness available to the evaluator that allows the evaluator to interface directly with the DSC. Tests may also require the use of an automated script provided by either the vendor or the evaluator. If a test script is made available, the evaluator shall verify that it includes sufficient detail to validate the claims made in the TSS.

Test 1: Demonstrate the TOE can validity-stamp data.

For each configurable option to validity-stamp data, the evaluator shall configure the TOE to create a data object or import a data object which has not-yet been validity-stamped. The evaluator shall then instruct the TOE to validity-stamp this data object. The evaluator shall then export each data object and demonstrate it has been validity-stamped in accordance with the configured options.

Test 2: Demonstrate the TOE can disable validity-stamping of data objects.

The evaluator shall disable, or ensure validity-stamping has been disabled on the TOE.

The evaluator shall create a data object on the TOE or import an already-created data object which has not been validity-signed.

The evaluator shall export that data object and verify it has not been validity-stamped.

4.2.1.1.4. KMD

There are no KMD evaluation activities for this component.

4.2.2. Factory Reset (Extended - FDP_FRS_EXT)

4.2.2.1. FDP_FRS_EXT.2 Factory Reset Behavior

4.2.2.1.1. TSS

The evaluator shall examine the TSS to determine the pre-installed SDOs that are reverted to their factory settings when a factory reset occurs, what the factory settings are for those SDOs, and that the TSS states that all non-persistent SDOs are destroyed.

4.2.2.1.2. AGD

The evaluator shall examine the operational guidance and verify that it identifies the pre-installed SDOs that are reverted to their initial values when a factory reset has been performed.

4.2.2.1.3. Test

The evaluator shall perform the following test:

1. The evaluator shall use each supported role to create or import an SDE or SDO that has known data.
2. The evaluator shall then verify that the created SDE/SDO resides either within the DSC, or under the control of the DSC.
3. The evaluator shall perform some action for each created or imported SDE/SDO in step 1 that demonstrates that the SDE/SDO has been set to the indicated value.
4. For each pre-installed SDO that is identified in FDP_FRS_EXT.2.1, the evaluator shall perform some action to verify what the current value of that SDO is.
5. The evaluator shall initiate a factory reset.
6. For each operation in step 3, the evaluator shall re-attempt the operation and verify that it no longer completes successfully because the SDE/SDO data has been erased.

7. For each pre-installed SDO that is identified in step 4, the evaluator shall re-attempt the operation and verify that the SDOs have been set to their factory default values.

4.2.2.1.4. KMD

There are no KMD evaluation activities for this component.

4.2.3. Mutable/Immutable Firmware (Extended - FDP_MFW_EXT)

4.2.3.1. FDP_MFW_EXT.2 Basic Firmware Integrity

4.2.3.1.1. TSS

The evaluator shall verify that the TSS describes which critical memory is measured for these integrity values and how the measurement is performed (including which TOE software measures the memory integrity values, how that software accesses the critical memory, and which algorithms are used).

4.2.3.1.2. AGD

If the integrity values are provided to the administrator, the evaluator shall verify that the AGD guidance contains instructions for retrieving these values and information for interpreting them. For example, if multiple measurements are taken, what those measurements are and how changes to those values relate to changes in the device state.

4.2.3.1.3. Test

Note that the following test may require the developer to provide access to a test platform that provides the evaluator with tools that are not typically available to end users.

The evaluator shall repeat the following test for each measurement:

The evaluator shall boot the TOE in an approved state and record the measurement taken. The evaluator shall modify the critical memory or value that is measured. The evaluator shall reboot the TOE and verify that the measurement changed.

4.2.3.1.4. KMD

There are no KMD evaluation activities for this component.

4.2.3.2. FDP_MFW_EXT.3 Firmware Authentication with Identity of Guarantor

4.2.3.2.1. TSS

The evaluator shall examine the TSS to ensure it describes the methods and identities used to verify integrity and authenticity of the firmware. The TSS shall identify the Guarantor and how to verify its identity.

4.2.3.2.2. AGD

There are no guidance activities for this component.

4.2.3.2.3. Test

The TOE guarantees the authenticity of the firmware using the identity of the Guarantor. This prevents impersonating a Guarantor when sending firmware to a device or modifying the firmware in transit.

Test 1: Verify Authentic Firmware

The evaluator shall trigger the TOE to load and evaluate the authenticity of authentic firmware according the methods described in the TSS. The evaluator shall ensure that the TOE provides a clear indication of the success of the evaluation to consider the test a 'Pass', otherwise, the test is a 'Fail'.

Test 2: Verify Unauthentic Firmware

The evaluator shall deliberately modify authentic firmware.

The evaluator shall trigger the TOE to load and evaluate the authenticity of the deliberately modified firmware according the methods described in the TSS. The evaluator shall ensure that the TOE provides a clear indication of the failure of the evaluation to consider the test a 'Pass', otherwise, the test is a 'Fail'.

4.2.3.2.4. KMD

There are no KMD evaluation activities for this component.

4.3. Identification and Authentication (FIA)

4.3.1. Authorization Failure Handling (Extended - FIA_AFL_EXT)

4.3.1.1. FIA_AFL_EXT.2 Authorization Failure Response

4.3.1.1.1. TSS

The evaluator shall examine the TSS to determine that it describes the method by which access to an SDO is restored following a lockout that results from excessive authentication failures.

4.3.1.1.2. AGD

The evaluator shall examine the guidance to ensure that it describes the method by which an administrator unlocks access to an SDO following a lockout that results from excessive authentication failures.

4.3.1.1.3. Test

The evaluator shall intentionally fail authentication attempts to access an SDO until they are locked out from interacting with it. The evaluator shall then follow the operational guidance to unlock access to the SDO and verify that it was successful by subsequently using valid credentials to access the SDO.

4.3.1.1.4. KMD

There are no KMD evaluation activities for this component.

4.4. Protection of the TSF (FPT)

4.4.1. Fail Secure (FPT_FLS)

4.4.1.1. FPT_FLS.1/FW Failure with Preservation of Secure State (Firmware)

4.4.1.1.1. TSS

The evaluator shall examine the TSS to verify that it describes the actions taken when the TOE experiences each of the stated failures and how these actions ensure the DSC preserves a secure state.

The evaluator shall verify that the TSS describes the state of the DSC when the firmware validity checks fail, including the various failure modes assumed.

4.4.1.1.2. AGD

For each failure state, the evaluator shall examine the operational guidance to verify that it describes what actions should be taken to attempt to resolve the failure state.

4.4.1.1.3. Test

Note that this test requires firmware builds that are deliberately invalidated to cause authenticity, integrity, and rollback violation failures.

The evaluator shall examine the TOE's behavior when it is loaded with a firmware build that causes a firmware failure. The evaluator shall ensure that when the failure occurs, the TOE prevents further processing of TSF and user data and performs any actions consistent with maintaining a secure state as described in the TSS.

The evaluator shall repeat this test as necessary to observe each of the specific firmware failures identified in the SFR.

4.4.1.1.4. KMD

There are no KMD evaluation activities for this component.

4.4.2. Replay Detection (FPT_RPL)

4.4.2.1. FPT_RPL.1/Rollback Replay Detection (Rollback)

4.4.2.1.1. TSS

The evaluator shall examine the TSS and other vendor documentation and ensure that they describe the methods used to guarantee the validity of firmware identifiers and prevents the TSF from executing older instances than that which is currently authorized.

4.4.2.1.2. AGD

There are no AGD evaluation activities for this component.

4.4.2.1.3. Test

The evaluator shall repeat the following tests to cover all allowed firmware verification mechanisms as described in the TSS. For example, if the firmware verification mechanism replaces an entire partition or subset of the DSC scope containing many separate code files, the evaluator does not need to repeat the test for each individual file.

Test 1: The evaluator shall attempt to execute an earlier instance or build of the software (as determined by the vendor). The evaluator shall verify that this attempt fails by checking the version identifiers or cryptographic hashes of the firmware against those previously recorded and checking that the values do not correspond to an unauthorized build.

Test 2: The evaluator shall attempt to execute a current or later instance and shall verify that the firmware execution succeeds.

4.4.2.1.4. KMD

There are no KMD evaluation activities for this component.

4.5. Trusted Path/Channels (FTP)

4.5.1. CCM Protocol (Extended - FTP_CCMP_EXT)

4.5.1.1. FTP_CCMP_EXT.1 CCM Protocol

4.5.1.1.1. TSS

The evaluator shall verify that the TSS includes a description of the TOE's expected responses to CCMP authentication failures and malformed or invalid CCMP data units.

4.5.1.1.2. AGD

There are no AGD evaluation activities for this component.

4.5.1.1.3. Test

The evaluator shall perform the following tests:

Test 1: The evaluator shall attempt to establish a CCMP connection to the TOE from an external entity, observe the return traffic with a network traffic analyzer, and verify that the connection succeeds and that the traffic is identified as properly constructed CCMP data units.

Test 2: The evaluator shall attempt to establish a CCMP connection to the TOE using five messages with incorrect or invalid authentication factors and verify that an authentication failure or error status is returned.

Test 3: The evaluator shall attempt to establish a CCMP connection to the TOE using five different

messages that are malformed or invalid due to noncompliance with the CCMP standard and observe that all connection attempts are unsuccessful.

Test 4: The evaluator shall establish a valid CCMP connection to the TOE. Once this has been established, the evaluator shall send ten different messages that are malformed or invalid due to noncompliance with the CCMP standard and observe that each of these messages are rejected.

4.5.1.1.4. KMD

There are no KMD evaluation activities for this component.

4.5.2. GCM Protocol (Extended - FTP_GCMP_EXT)

4.5.2.1. FTP_GCMP_EXT.1 GCM Protocol

4.5.2.1.1. TSS

The evaluator shall verify that the TSS includes a description of the TOE's expected responses to GCMP authentication failures and malformed or invalid GCMP data units.

4.5.2.1.2. AGD

There are no AGD evaluation activities for this component.

4.5.2.1.3. Test

The evaluator shall perform the following tests:

Test 1: The evaluator shall attempt to establish a GCMP connection to the TOE from an external entity, observe the return traffic with a network traffic analyzer, and verify that the connection succeeds and that the traffic is identified as properly constructed GCMP data units.

Test 2: The evaluator shall attempt to establish a GCMP connection to the TOE using five messages with incorrect or invalid authentication factors and verify that an authentication failure or error status is returned.

Test 3: The evaluator shall attempt to establish a GCMP connection to the TOE using five different messages that are malformed or invalid due to noncompliance with the GCMP standard and observe that all connection attempts are unsuccessful.

Test 4: The evaluator shall establish a valid GCMP connection to the TOE. Once this has been established, the evaluator shall send ten different messages that are malformed or invalid due to noncompliance with the GCMP standard and observe that each of these messages are rejected.

4.5.2.1.4. KMD

There are no KMD evaluation activities for this component.

4.5.3. Inter-TSF Trusted Channel (Extended - FTP_ITC_EXT)

4.5.3.1. FTP_ITC_EXT.1 Cryptographically Protected Communications Channels

4.5.3.1.1. TSS

The evaluator shall review the TSS to determine that it lists all trusted channels the TOE uses for remote communications, including both the external entities and remote users used for the channel as well as the protocol that is used for each.

4.5.3.1.2. AGD

There are no AGD evaluation activities for this component.

4.5.3.1.3. Test

The evaluator shall configure the TOE to communicate with each external IT entity or type of remote user identified in the TSS. The evaluator shall monitor network traffic while the TSF performs communication with each of these destinations. The evaluator shall ensure that for each session a trusted channel was established in conformance with the protocols identified in the selection.

4.5.3.1.4. KMD

There are no KMD evaluation activities for this component.

4.5.4. Encrypted Data Communications (Extended - FTP_ITE_EXT)

4.5.4.1. FTP_ITE_EXT.1 Encrypted Data Communications

4.5.4.1.1. TSS

The evaluator shall review the TSS to determine that it lists all encryption mechanisms the TOE uses for protected external communications, along with the types of communications protected using each mechanism.

4.5.4.1.2. AGD

There are no AGD evaluation activities for this component.

4.5.4.1.3. Test

The evaluator shall configure the TOE to communicate with each external entity identified in the TSS. The evaluator shall initiate a transaction that will result in data being transferred to the TOE through the mechanism and other data returned to the initiating entity through the mechanism. The evaluator must verify that the data returned to the entity was encrypted using the documented mechanism when received.

4.5.4.1.4. 4KMD

There are no KMD evaluation activities for this component.

4.5.5. Physically Protected Channel (Extended - FTP_ITP_EXT)

4.5.5.1. FTP_ITP_EXT.1 Physically Protected Channel

4.5.5.1.1. TSS

The evaluator shall review the TSS to determine that it lists all mechanisms the TOE uses for physically protected external communications, along with the types of communications protected using each mechanism.

4.5.5.1.2. AGD

There are no AGD evaluation activities for this component.

4.5.5.1.3. Test

There are no test activities for this component.

4.5.5.1.4. KMD

There are no KMD evaluation activities for this component.

Chapter 5. Evaluation Activities for SARs

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

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

5.1. ASE: Security Target Evaluation

When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2 as well as the EAs for the optional and selection-based SFRs claimed by the ST and specified in Section 3 and Section 4.

5.2. ADV: Development

5.2.1. Basic Functional Specification (ADV_FSP.1)

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

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

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

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the [DSC cPP]: no additional "functional specification" documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the [DSC cPP] rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7 is treated as

implicit and no separate mapping information is required for this element.

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

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

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

5.2.1.1. Evaluation Activity

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

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or if they are used by the TSF to send or receive security-relevant data or to invoke an API or other service that supports the execution of a security function. 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.

The set of TSFI that are provided as evaluation evidence are contained in the operational guidance.

5.2.1.2. 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.

5.2.1.3. Evaluation Activity

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

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, *Evaluation Activities for SFRs*, including the EAs associated with testing of the interfaces.

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

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

5.2.1.4. Evaluation Activity for Specifying DSC Interface for Use in Composite Evaluations

Table 5 below lists all potentially exportable DSC services supported by this [DSC cPP] along with the SFRs that implement the services. The ST author should include this table in the ST after removing the rows for services not supported by the DSC or whose interfaces are not available to dependent components. SFRs surrounded in brackets ("[SFR]") are optional or selection-based. All the SFRs listed in the resulting table must be selected in the ST.

Table 5. Mapping between DSC Exportable Services and SFRs

Service	**DSC SFR**
Asymmetric-Key Generation	[FCS_CKM.1/AK]
Symmetric-Key Generation	[FCS_CKM.1/SK]
Private-Key Cryptography	FCS_CKM.2
Public-Key Cryptography	FCS_CKM.2
Symmetric-Key Cryptography	FCS_COP.1/SKC, [FTP_CCMP_EXT.1, FTP_GCMP_EXT.1]
KEK Creation	FCS_CKM.1/KEK
Random-Bit Generation	FCS_RBG_EXT.1, [FCS_ENT_EXT.1]
RNG Seeding	FCS_RBG_EXT.1, [FCS_RBG_EXT.2]
Key Wrapping	FCS_COP.1/KeyEnc
Cryptographic Hashing	FCS_COP.1/Hash
Cryptographic Keyed Hash	FCS_COP.1/HMAC
Password Hashing (PBKDF)	[FCS_COP.1/PBKDF]
Signature Generation	FCS_COP.1/SigGen
Signature Verification	FCS_COP.1/SigVer
Factory Reset	FDP_FRS_EXT.1, [FDP_FRS_EXT.2]
Destroy Persistent Secrets	FCS_CKM.4
Set Root Encryption Key (mutable H/W)	FCS_STG_EXT.1
Store Persistent Secrets	FCS_STG_EXT.1-3
Wrap Transient Secrets	FCS_STG_EXT.1-3
Get Time Stamp	FPT_STM.1
Cryptographic Self-Tests	FPT_TST.1

The ST must include a description of the interface for each exported service. Interface descriptions must include the purpose of the interface, method of use, parameters, and parameter descriptions. The descriptions must provide sufficient detail to allow the ST author of a composite evaluation to align the DSC interfaces with the dependent component SFRs that they satisfy. Multiple DSC operations may be combined to satisfy dependent component SFRs.

It is not required that a DSC expose interfaces to all the Table 1 services to a dependent component.

Nor is a DSC limited to providing only the above services. But all such services that are provided must be documented and the interfaces described.

5.3. 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 EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and the AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to administrators and users (as appropriate) as part of the TOE.

Note that a DSC is not a direct-to-consumer product and is more typically integrated with a larger system that is distributed to end customers. The guidance documentation may reflect this. For example, information on how to administer the TOE may not necessarily be user-facing instructions; rather, they may be presented as APIs or other information that could be used by a third-party integrator to develop user-facing functions that interface with the DSC to perform the required behavior.

Note that unique aspects of the DSC's design or architecture may mean that some of these EAs below are not applicable to the TOE because they may be enforced by default. If a given EA is not applicable, the evaluator shall ensure that adequate rationale is provided to justify the lack of applicability.

5.3.1. Operational User Guidance (AGD_OPE.1)

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

In addition, the evaluator performs the EAs specified below.

5.3.1.1. Evaluation Activity

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

5.3.1.2. Evaluation Activity

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

5.3.1.3. Evaluation Activity

The evaluator *shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a*

warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

5.3.1.4. Evaluation Activity

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

5.3.2. Preparative Procedures (AGD_PRE.1)

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

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

In addition, the evaluator performs the EAs specified below. Note that unique aspects of the DSC's design or architecture may mean that some of these EAs are not applicable to the TOE because they may be enforced by default. If a given EA is not applicable, the evaluator shall ensure that adequate rationale is provided to justify the lack of applicability.

5.3.2.1. Evaluation Activity

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

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 product developers/engineers that are responsible for integrating the DSC into a larger system).

5.3.2.2. Evaluation Activity

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

5.3.2.3. Evaluation Activity

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

5.3.2.4. Evaluation Activity

The evaluator shall examine the preparative procedures to ensure they include instructions to manage

the security of the TSF as a product and as a component of the larger operational environment.

5.4. ALC: Life-cycle Support

5.4.1. Labelling of the TOE (ALC_CMC.1)

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

5.4.2. TOE CM coverage (ALC_CMS.1)

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

5.5. ATE: Tests

5.5.1. Independent Testing - Conformance (ATE_IND.1)

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

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Section 2, Section 3 and Section 4.

5.6. AVA: Vulnerability Assessment

5.6.1. Vulnerability Survey (AVA_VAN.1)

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

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. As Table 6 indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Table 6. Mapping of AVA_VAN.1 CEM Work Units to Evaluation Activitiess

CEM AVA_VAN.1 Work Units	Evaluation Activities
AVA_VAN.1-1 The evaluator ***shall examine*** the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.	The evaluator shall perform the CEM activity as specified.
AVA_VAN.1-2 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state	The evaluator shall perform the CEM activity as specified.
AVA_VAN.1-3 The evaluator ***shall examine*** sources of information publicly available to identify potential vulnerabilities in the TOE.	Replace CEM work unit with activities outlined in Section A.1, “Sources of vulnerability information”
AVA_VAN.1-4 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.	Replace the CEM work unit with the analysis activities on the list of potential vulnerabilities in Section A.1, “Sources of vulnerability information”, and documentation as specified in Section A.3, “Reporting”.
AVA_VAN.1-5 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.	Replace the CEM work unit with the activities specified in Section A.2, “Process for Evaluator Vulnerability Analysis”.
AVA_VAN.1-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include: a. identification of the potential vulnerability the TOE is being tested for; b. instructions to connect and setup all required test equipment as required to conduct the penetration test; c. instructions to establish all penetration test prerequisite initialconditions; d. instructions to stimulate the TSF; e. instructions for observing the behaviour of the TSF; f. descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison againstexpected results; g. instructions to conclude the test and establish the necessary post-test state for the TOE.	The CEM work unit is captured in Section A.3, “Reporting”; there are no substantive differences.

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

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

5.6.1.1. Evaluation Activity (Documentation):

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

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

In addition to the activities specified by the CEM in accordance with Table 6, “Mapping of AVA_VAN.1 CEM Work Units to Evaluation Activitiess” above, the evaluator shall perform the following activities.

5.6.1.2. Evaluation Activity

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

Chapter 6. Required Supplementary Information

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

- Entropy Documentation, which is evaluated against the guidance specified in Appendix D of the [DSC cPP].
- A Key Management Document (KMD), which is evaluated against all relevant KMD Evaluation Activities in this SD.

Chapter 7. References

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

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

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

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

[DSC cPP] collaborative Protection Profile for Dedicated Security Component, Version 1.0.1, December 13, 2022

[ISO-TR] ISO/IEC. *ISO/IEC 24759-3:2017 Information Technology - Security Techniques - Test Requirements for Cryptographic Modules*, ISO/IEC, 2017

Appendix A: Vulnerability Analysis

A.1. Sources of vulnerability information

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

1. A list of flaw hypotheses applicable to the technology described by the [DSC cPP] derived from public sources as documented in Section A.1.1—this fixed set has been agreed to by the iTC. Additionally, this will be supplemented with entries for a set of public sources (as indicated below) that are directly applicable to the TOE or its identified components (as defined by the process in Section A.1.1 below); this is to ensure that the evaluators include in their assessment applicable entries that have been discovered since the [DSC cPP] was published;
2. A list of flaw hypotheses contained in this document that are derived from lessons learned specific to that technology and other iTC input (that might be derived from other open sources and vulnerability databases, for example) as documented in Section A.1.2;
3. A list of flaw hypotheses derived from information available to the evaluators; this includes the baseline evidence provided by the vendor described in this Supporting Document (documentation associated with EAs, documentation described in Section 5.6.1.1, documentation described in Section 6), as well as other information (public or based on evaluator experience) as documented in Section A.1.3; and
4. A list of flaw hypotheses that are generated through the use of iTC-defined tools (e.g. nmap, protocol testers) and their application is specified in Section A.1.4.

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

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

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

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

- Vendor name
- Product name
- Third-party or OEM components, if known
- If the DSC is embedded in specific commercially-available products, any terms relevant to a sample of those products

In order to successfully complete this activity, the evaluator will use the developer provided list of

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

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

In order to supplement this list, the evaluators shall also perform a search on the sources listed above to determine a list of potential flaw hypotheses that are more recent that the publication date of the [DSC cPP], and those that are specific to the TOE and its components as specified by the additional documentation mentioned above. Any duplicates - either in a specific entry, or in the flaw hypothesis that is generated from an entry from the same or a different source - can be noted and removed from consideration by the evaluation team.

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

A.1.2. Type 2 Hypotheses—iTC-Sourced

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

There are currently no iTC-sourced flaw hypotheses. A future revision of the [DSC cPP] may update this section for relevant findings made by evaluation laboratories.

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

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

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

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

A.1.4. Type 4 Hypotheses—Tool-Generated

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

A.2. Process for Evaluator Vulnerability Analysis

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

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

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

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

For each hypothesis, the evaluator shall note whether the flaw hypothesis has been successfully disproved, successfully proven to have identified a flaw, or requires further investigation. It is important to have the results documented as outlined in Section A.3 below.

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

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

If the developer, the evaluator, and the CB agree that the flaw is exploitable only above Basic Attack

Potential and does not require resolution for any other reason, then no change is made and the flaw is noted as a residual vulnerability in the CB-internal report (ETR).

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

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

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

As indicated in Section A.3, the public statement with respect to vulnerability analysis that is performed on TOEs conformant to the [DSC cPP] is constrained to coverage of flaws associated with Types 1 and 2 (defined in Section A.1) flaw hypotheses only. The fact that the iTC generates these candidate hypotheses indicates these must be addressed.

A.3. Reporting

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

The public-facing report contains:

- The flaw identifiers returned when the procedures for searching public sources were followed according to instructions in the Supporting Document per Section A.1.1;
- A statement that the evaluators have examined the Type 1 flaw hypotheses specified in this Supporting Document in Section A.1.1 (i.e. the flaws listed in the previous bullet) and the Type 2 flaw hypotheses specified in this Supporting Document by the iTC in Section A.1.2.

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

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

- A list of all of the flaw hypotheses generated (cf. AVA_VAN.1-4);
- The evaluator penetration testing effort, outlining the testing approach, configuration, depth and results (cf. AVA_VAN.1-9);
- All documentation used to generate the flaw hypotheses (in identifying the documentation used in coming up with the flaw hypotheses, the evaluation team must characterize the documentation so that a reader can determine whether it is strictly required by this Supporting Document, and the nature of the documentation (design information, developer engineering notebooks, etc.));
- The evaluator shall report all exploitable vulnerabilities and residual vulnerabilities, detailing for each:
    - Its source (e.g. CEM activity being undertaken when it was conceived, known to the

evaluator, read in a publication);

- The SFRs not met;

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

Appendix B: Equivalency Considerations

B.1. Introduction

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

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

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

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

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

B.2. Evaluator guidance for determining equivalence

B.2.1. Strategy

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

ability to enforce the SFRs. Each analysis of an individual factor will result in one of two outcomes:

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

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

B.3. Test presentation/Truth in advertising

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