Supporting Document
Mandatory Technical Document

Full Drive Encryption: Enterprise Management

January 2018

Version 2.0

CCDB-2018-xxxx

Foreword

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

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

This supporting document has been developed by *Full Drive Encryption iTC* and is designed to be used to support the evaluations of products against products that claim conformance to the collaborative Protection Profile Module for Full Drive Encryption – Enterprise Management.

Technical Editor:

FDE iTC

Document history:

V2.0 January 2018 (Initial release for public review, version set to 2.0 for consistency with other FDE materials)

General Purpose:

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

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

The introduction of an Enterprise Management capability greatly expands the attack surface of an FDE because keys for multiple FDEs are stored, and potentially generated, on a remote server. This means that malicious or careless administrators may adversely affect the behaviour of individual FDEs and insecurely implemented communications or credential recovery processes can lead to compromise of key data that was previously restricted to a single device.

Field of special use:

Full Drive Encryption devices that are deployed and managed underneath a single Enterprise Management server that is responsible for the key lifecycle of the Authorization Acquisition component of these devices.

Acknowledgements:

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

Table of Contents

1 Introduction

1.1 Technology Area and Scope of Supporting Document

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

- Centralized key management/storage for multiple FDEs
- Multi-user access to an endpoint protected by an FDE
- Remote user authentication
- Data recovery in the event of a lost or forgotten credential

2 As an extension of the *FDE cPP – Authorization Acquisition*, the *Enterprise Management Module* extends the Target of Evaluation to a central server where the key data for a number of FDEs (i.e. a number of instances of the same Authorization Acquisition component) are maintained. Depending on the functionality provided by an Enterprise Management server, it may be responsible solely for receiving and maintaining key data generated by an AA, or it may also assume responsibility for key generation in addition to storage.

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

4 a) *collaborative Protection Profile Module for Full Drive Encryption – Enterprise Management, January 4, 2018*

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

1.2 Structure of the Document

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

7 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.

8 In general, if all Evaluation Activities (for both SFRs and SARs) are successfully completed in an evaluation then it would be expected that the overall verdict for the evaluation is a 'pass'. To reach a 'fail' verdict when the Evaluation Activities have been successfully completed would require a specific justification from the evaluator as to why the Evaluation Activities were not sufficient for that TOE.

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

1.3 Terminology

1.3.1 Glossary

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

11 **Supplementary information —** information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP (see description in section 4).

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

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

Term	*Meaning*
Target of Evaluation	A set of software, firmware and/or hardware possibly accompanied by guidance. [CC1]

1.3.2 Acronyms

Acronym	*Meaning*
AA	Authorization Acquisition
AES	Advanced Encryption Standard
BEV	Border Encryption Value
BIOS	Basic Input Output System
CBC	Cipher Block Chaining
CC	Common Criteria
CCM	Counter with CBC-Message Authentication Code
CEM	Common Evaluation Methodology
CPP	Collaborative Protection Profile
DEK	Data Encryption Key
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EE	Encryption Engine
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standards
FDE	Full Drive Encryption
FFC	Finite Field Cryptography
GCM	Galois Counter Mode
HMAC	Keyed-Hash Message Authentication Code
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
ITSEF	IT Security Evaluation Facility
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
IV	Initialization Vector
KEK	Key Encryption Key
KMD	Key Management Description
KRK	Key Release Key
MBR	Master Boot Record
NIST	National Institute of Standards and Technology
OS	Operating System
RBG	Random Bit Generator
RNG	Random Number Generator
RSA	Rivest Shamir Adleman Algorithm
SAR	Security Assurance Requirement
SED	Self Encrypting Drive
SHA	Secure Hash Algorithm
SFR	Security Functional Requirement
SPD	Security Problem Definition
SPI	Serial Peripheral Interface
ST	Security Target

TOE	Target of Evaluation
TPM	Trusted Platform Module
TSF	TOE Security Functionality
TSS	TOE Summary Specification
USB	Universal Serial Bus
XOR	Exclusive or
XTS	XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing

2 Evaluation Activities for SFRs

12 Note that as an extension of the FDE cPP – Authorization Acquisition, all functionality that is mandated by that cPP must be implemented by a TOE claiming conformance with this module. For those functions that may be implemented by either the AA itself or by the Enterprise Management component of the TOE, it is necessary for the evaluator to perform the required assurance activities against the part of the TOE that implements the function in question. In most cases, the actual testing method will not differ based on the component that implements the function; only the tested component will differ.

13 For cases where the Enterprise Management server implements functionality that was originally defined for the AA component, the iteration '/Server' has been appended to the SFR so that the component implementing the function is clearly identified.

14 Note that many of the SFRs defined for this cPP-Module are selection-based or strictly optional based on how the Enterprise Management server satisfies the required functionality and what optional features are provided. The evaluator will only perform the required assurance activities for the SFRs that are claimed in the TOE's Security Target.

15 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.

16 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.

17 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.

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

2.1 Cryptographic Support (FCS)

2.1.1 Key Chaining (FCS_KYC_EXT)

2.1.1.1 FCS_KYC_EXT.1/Server Key Chaining (Initiator) (Management Server)

2.1.1.1.1 TSS

19 The evaluator shall verify the TSS contains a high-level description of the BEV sizes – that it supports BEV outputs of no fewer 128 bits for products that support only AES-128, and no fewer than 256 bits for products that support AES-256.

20 The evaluator shall verify the TSS contains a description of the controls preventing the BEV from being provided to the EE before validation has occurred.

2.1.1.1.2 Operational Guidance

21 If there are configurations to enable or disable use of enterprise server, which modify the key chain, they shall be described.

22 If there are configurations on to enable recovery mechanisms, they shall be described.

2.1.1.1.3 KMD

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

24 The evaluator shall examine the KMD describes a high level description of the key hierarchy for all authorizations methods selected in FCS_AFA_EXT.1 that are used to protect the BEV. The evaluator shall examine the KMD to ensure it describes the key chain in detail. The description of the key chain shall be reviewed to ensure it maintains a chain of keys using key wrap or key derivation methods that meet FCS_COP.1(d) and FCS_KDF_EXT.1. The evaluator shall ensure the chain of keys is maintained from the authorization factor or recovery value to the BEV or from the authorization factor or recovery value to the TOE server and then key chain from the server to the BEV.

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

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

2.1.1.1.4 Test

27 There are no test evaluation activities for this SFR.

2.1.2 Submask Combining (FCS_SMC_EXT)

2.1.2.1 FCS_SMC_EXT.1/Server Submask Combining (Management Server)

2.1.2.1.1 TSS

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

2.1.2.1.2 Operational Guidance

29 There are no AGD assurance activities for this SFR.

2.1.2.1.3 KMD

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

2.1.2.1.4 Test

31 The evaluator shall perform the following test:

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

2.2 Identification and Authentication (FIA)

2.2.1 User Authentication (FIA_UAU)

2.2.1.1 FIA_UAU.1 Timing of Authentication

2.2.1.1.1 TSS

33 The evaluator shall examine the TSS to determine that it describes the list of actions that are performed on behalf of the administrator prior to login of the administrator. The evaluator shall examine the TSS to determine that it describes the list of actions that require administrator authentication.

2.2.1.1.2 Operational Guidance

34 The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps (e.g., establishing credential material such as pre- shared keys, tunnels, certificates, etc.) to logging in are described. For each supported login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the allowed services.

2.2.1.1.3 KMD

35 There are no KMD evaluation activities for this SFR.

2.2.1.1.4 Test

36 The evaluator shall perform the following tests:

37 Test 1: The evaluator shall verify that the list of actions allowed without administrator login completes successfully without requiring administrator login and make sure this list is consistent with the TSS.

38 Test 2: The evaluator shall verify that attempting any other action requires successful entry of an administrator credential.

39 Test 3: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method.

40 For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

41 Test 4: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

2.2.2 User Identification (FIA_UID)

2.2.2.1 FIA_UID.1 Timing of Identification

2.2.2.1.1 TSS

42 The evaluator shall examine the TSS to determine that it describes the list of actions that are performed on behalf of the administrator prior to identification of the administrator.

2.2.2.1.2 Operational Guidance

43 The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps for creating and configuring administrator accounts are described.

2.2.2.1.3 KMD

44 There are no KMD evaluation activities for this SFR.

2.2.2.1.4 Test

45 The evaluator shall perform the following tests:

46 Test 1: The evaluator shall verify that the list of actions allowed without administrator identification completes successfully without requiring the administrative user to be identified and make sure this list is consistent with the TSS.

47 Test 2: The evaluator shall verify that attempting any other action requires successful entry of an administrator account name and successful entry of the administrator account credential.

2.3 Security Management (FMT)

2.3.1 Management of TSF Data (FMT_MTD)

2.3.1.1 FMT_MTD.1 Management of TSF Data

2.3.1.1.1 TSS

48 The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are available to the administrator are identified. For each of these functions, the evaluator shall also confirm that the TSS details when changes may be made to the encryption keys and/or intermediate values.

2.3.1.1.2 Operational Guidance

49 The evaluator shall verify that the guidance document describes what operations on the encryption keys and intermediate values are allowed to the administrator at what times.

2.3.1.1.3 KMD

50 There are no KMD evaluation activities for this SFR.

2.3.1.1.4 Test

51 Test 1: The evaluator shall try to perform at least one of the related actions without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). This test should fail.

52 Test 2: The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This test should pass.

53 Test 3: The evaluator shall try to perform at least one of the actions at the times that are not permitted. This test should fail.

54 Test 4: The evaluator shall try to perform at least one of the actions at the times are permitted. This test should pass.

2.3.2 Specification of Management Functions (FMT_SMF)

2.3.2.1 FMT_SMF.1/Server Specification of Management Functions (Management Server)

2.3.2.1.1 TSS

55 The evaluator shall examine the TSS to ensure that it describes which of the selections are provided by the TOE. Additionally, the TSS shall describe which of the configurable selections can be disabled on the Enterprise Management Server. The evaluator shall examine the TSS to ensure that it describes whether the TOE provides

the ability to initiate key generation, escrow, zeroization and/or recovery or whether it requests the endpoint to perform those functions.

2.3.2.1.2 Operational Guidance

56 The evaluator shall examine the Guidance Documents to ensure that, if supported, configuration of the following options is described, including any reliance on the Operational Environment if applicable:

- Register new endpoint
- Revoke registration of an endpoint
- Initiate key generation
- Initiate key escrow
- Initiate key recovery
- Initiate key zeroization
- Set encryption policy (supported algorithms and key sizes)
- Change Administrator passwords
- Change user passwords
- Change Recovery Credentials
- Define Administrators of the TOE
- Enable/Disable the use of recovery credentials (end users)
- Configure the number of failed authentication attempts before issuing a key sanitization of the DEK
- Configure the number of authentication attempts that can be made in a 24 hour period
- Configure the number of failed authentication attempts required to begin blocking subsequent attempts
- The ability to enable/disable one or more functions defined in the base PP
- The ability to authorize whether or not users can perform one or more of the functions in the base PP.

2.3.2.1.3 KMD

57 There are no KMD evaluation activities for this SFR.

2.3.2.1.4 Test

58 The evaluator shall perform the following tests for each claimed management function:

59 Test 1: The evaluator shall configure the management server and two endpoints according to the guidance documents. The evaluator shall register the endpoints with the management server. The evaluator shall verify that the endpoints are identified by the management server as defined in the guidance documents. This test shall pass.

60 Test 2: The evaluator shall disconnect the second endpoint from the network. The evaluator shall revoke the registration of the second endpoint in the management server. The evaluator shall attempt to connect the second endpoint to the network and verify the endpoint fails to connect or is displayed as revoked in the console.

61 Test 3: For each item that is performed by the TOE, the evaluator shall verify that the TOE performs the actions (e.g. generate key) and sends the result to the endpoint. The endpoint shall perform the actions necessary to accept the updated configuration (e.g. encrypt the data with the new key, update the encryption algorithm key size or mode and re-encrypt).

62 Test 4: For each item that is initiated by the TOE but performed on the endpoint, the evaluator shall verify that the TOE requests the endpoint to perform the action (generate a key and encrypt the data, zeroize a key).

63 Test 5: For each method of changing a credential, the evaluator shall first provision the initial authorization factor(s) in the Enterprise Server, and then verify all authorization values supported allow the user access to the encrypted data on configured endpoint. Then the evaluator shall exercise the management functions to change the authorization factor values to a new one on the Enterprise Server. Then he or she will verify that the endpoint denies access to the user's encrypted data when he or she uses the old or original authorization factor values to gain access.

64 Test 6: The evaluator shall add two administrators to the administrator group in the Enterprise Server and provision authorization factor(s) for each administrator. The evaluator shall verify that both administrators can log into the Enterprise Server using the provided…the provided authorization factors. The evaluator shall then exercise the management functions to change the authorization factor values for the first administrator to a new one on the Enterprise Server. Then he or she will verify that the Enterprise Server denies the first administrator access to the Management Console when the first administrator logs in with the old or original authorization factor to gain access. The evaluator shall also verify that the second administrator is still able to log in to the Enterprise Server with their original authorization factor.

65 Test 7: The evaluator shall verify that the second administrator from Test 2 can configure each of the supported authorization attempts configurations and shall verify that the endpoint denies access to the user's encrypted data as in described in the test actions of the AA SD Section 2.1.22 Test 1.

66 Test 8: If the TOE provides the capability to disable management of any capability allowed in the EM Module, the evaluator shall devise a test that ensures that each capability which can be disabled has been or can be disabled following guidance provided by the vendor.

67 Test 9: If the TOE provides the capability to manage capabilities in place of the AA or EE, where those administrative capabilities are then disabled in the AA or EE, the evaluator shall devise a test that ensures that each capability which can be disabled in the AA or EE and can be subsequently managed by the EM is tested as follows:

Disable the administrative capability in the AA/EE and enable it in the EM

Verify that the administration of the capability in the EM is successfully

2.3.3 Security Management Roles (FMT_SMR)

2.3.3.1 FMT_SMR.2 Restrictions on Security Roles

2.3.3.1.1 TSS

68 For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components if not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1. The evaluator shall examine that authentication and identification of Security Administrators cannot be compromised for any TOE component in this case.

2.3.3.1.2 Operational Guidance

69 The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration.

2.3.3.1.3 KMD

70 There are no KMD evaluation activities for this SFR.

2.3.3.1.4 Test

71 In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

72 For distributed TOEs where not every TOE component implements own user management and where authentication and identification of security administrators is not done according to FIA_UIA_EXT.1, the evaluator shall test that at least one component performs authentication and identification of Security Administrators according to FIA_UIA_EXT.1. In addition, the evaluator shall test that all TOE components perform authentication and identification of Security Administrators as described in the TSS.

2.4 Protection of the TSF (FPT)

2.4.1 Internal TOE TSF Data Transfer (FPT_ITT)

2.4.1.1 FPT_ITT.1 Basic Internal TSF Data Transfer Protection

2.4.1.1.1 TSS

73 The evaluator shall examine the TSS to determine that, for all communications between components of a distributed TOE, each communications mechanism is identified in terms of the allowed protocols and intra-TOE configurations for that IT entity. The

evaluator shall also confirm that all protocols listed in the TSS for these inter-component communications are specified and included in the requirements in the ST.

74 If the TOE is not a distributed TOE then no evaluator action is necessary. For a distributed TOE the evaluator carries out the activities below.

2.4.1.1.2 Operational Guidance

75 The evaluator shall confirm that the guidance documentation contains instructions for establishing the relevant allowed communication channels and protocols between each pair of authorized TOE components, and that it contains recovery instructions should a connection be unintentionally broken.

2.4.1.1.3 KMD

76 There are no KMD evaluation activities for this SFR.

2.4.1.1.4 Test

77 The evaluator shall perform the following tests:

78 Test 1: The evaluator shall ensure that communications using each supported protocol between each pair of authorized TOE components is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

79 Test 2: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

80 Test 3: The evaluator shall, for each protocol associated with each authorized IT entity tested during Test 1, physically interrupt the connection. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

81 Further assurance activities are associated with the specific protocols.

2.4.2 Key and Key Material Protection (FPT_KYP_EXT)

2.4.2.1 FPT_KYP_EXT.2 Storage of Protected Key and Key Material

2.4.2.1.1 TSS

82 The evaluator shall examine the TSS to verify that it describes the storage locations key material may be stored.

2.4.2.1.2 Operational Guidance

83 The evaluator shall verify that guidance documentation lists any configuration information associated with key storage. If any configuration changes the storage location of keys, it must be described.

2.4.2.1.3 KMD

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

85 The evaluator shall verify the KMD describes that all keys have an associated storage location(s) with them.

2.4.2.1.4 Test

86 There are no test evaluation activities for this SFR.

2.4.2.2 FPT_KYP_EXT.3 Attribution of Protected Key and Key Material

2.4.2.2.1 TSS

87 The evaluator shall examine the TSS to verify that it describes the method by which an association is maintained and verify it matches the selections.

2.4.2.2.2 Operational Guidance

88 The evaluator shall verify the guidance documentation provides instructions on how to configure the association, if any configuration is necessary.

2.4.2.2.3 KMD

89 There are no KMD evaluation activities for this SFR.

2.4.2.2.4 Test

90 For each method of association, the evaluator shall change the configuration so that the associate is broken and verify that enterprise functions do not work.

2.5 Trusted Path/Channels (FTP)

2.5.1 Inter-TSF Trusted Channel (FTP_ITC)

2.5.1.1 FTP_TRP.1 Inter-TSF Trusted Channel

2.5.1.1.1 TSS

91 The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

2.5.1.1.2 Operational Guidance

92 The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

2.5.1.1.3 KMD

93 There are no KMD evaluation activities for this SFR.

2.5.1.1.4 Test

94 The evaluator shall perform the following tests:

95 Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method are tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

96 Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

97 Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

98 Test 4: The evaluators shall ensure that, for each protocol associated with each authorized IT entity tested during test 1, the connection is physically interrupted. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

99 Further assurance activities are associated with the specific protocols.

3 Evaluation Activities for Optional Requirements

3.1 Cryptographic Support (FCS)

3.1.1 Cryptographic Key Management (FCS_CKM)

3.1.1.1 FCS_CKM.1(a)/Server Cryptographic Key Generation (Asymmetric Keys) (Server Communications

3.1.1.1.1 TSS

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

3.1.1.1.2 Operational Guidance

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

3.1.1.1.3 KMD

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

3.1.1.1.4 Test

103 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.

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

105 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.

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

107 1. Random Primes:
- Provable primes
- Probable primes

108 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

109 To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

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

111 *FIPS 186-4 ECC Key Generation Test*

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

113 *FIPS 186-4 Public Key Verification (PKV) Test*

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

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

115 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.

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

117 Cryptographic and Field Primes:

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

and two ways to generate the cryptographic group generator g:

118 Cryptographic Group Generator:

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

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

120 Private Key:

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

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

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

123 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.

3.1.1.2 FCS_CKM.2 Cryptographic Key Distribution

3.1.1.2.1 TSS

124 The evaluator shall ensure that the supported key distribution methods correspond to the key generation schemes identified in FCS_CKM.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each method.

3.1.1.2.2 Operational Guidance

125 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

3.1.1.2.3 KMD

126 There are no KMD activities for this SFR.

3.1.1.2.4 Test

127 The evaluator shall verify the implementation of the key distribution methods supported by the TOE using the applicable tests below.

128 SP800-56A Key Establishment Schemes

129 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.

130 Function Test

131 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.

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

133 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.

134 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.

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

136 Validity Test

137 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.

138 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 will 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 and/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).

139 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.

140 SP800-56B Key Establishment Schemes

141 The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

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

143 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.

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

145 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 KTS- OAEP 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 plaintext 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.

146 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. If KTS-KEM- KWS 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 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

3.1.1.3 FCS_CKM.2/Server Cryptographic Key Establishment (Server Communications)

3.1.1.3.1 TSS

147 The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1/Server. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. If Diffie-Hellman group 14 is selected from

FCS_CKM.2.1/Server, the TSS shall describe how the implementation meets RFC 3526 Section 3.

3.1.1.3.2 Operational Guidance

148 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

3.1.1.3.3 KMD

149 There are no KMD activities for this SFR.

3.1.1.3.4 Test

150 The evaluator shall verify the implementation of the key establishment schemes supported by the TOE using the applicable tests below.

151 SP800-56A Key Establishment Schemes

152 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.

153 Function Test

154 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.

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

156 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.

157 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.

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

159 Validity Test

160 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.

161 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 will 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 and/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).

162 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.

163 SP800-56B Key Establishment Schemes

164 The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

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

166 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.

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

168 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 KTS- OAEP 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 plaintext 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.

169 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. If KTS-KEM- KWS 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 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

170 Diffie-Hellman Group 14

171 The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_ITT.1 and FTP_TRP.1 that uses Diffie-Hellman group 14.

3.1.1.4 FCS_CKM.4(a)/Server Cryptographic Key Destruction (Server Communications)

3.1.1.4.1 TSS

172 (Key Management Description may be used if necessary details describe proprietary information)

173 The evaluator shall check to ensure the TSS (KMD) lists each type of key material, its origin, possible temporary locations (e.g. memory), and storage location (e.g. SQL database).

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

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

description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs, API's between EM and AA to transfer key material).

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

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

3.1.1.4.2 Operational Guidance

178 There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

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

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

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

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

3.1.1.4.3 KMD

183 There are no requirements for a KMD, however, vendors may provide any information required by the TSS which they deem proprietary as part of a KMD.

3.1.1.4.4 Test

184 Test 1: Applied to each key encrypting key or BEV 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:

- Record the value of the key in the TOE subject to clearing.
- Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
- Cause the TOE to clear the key.
- Cause the TOE to stop the execution but not exit.
- Cause the TOE to dump the entire memory of the TOE into a binary file.
- Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.
- Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.
- Ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.
- Ensure that partial key fragments do not remain in memory. If a fragment is found, there is a miniscule chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is found the test fails.

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

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

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

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

described for Use Case 1 test 1 above), and if a fragment is found in the repeated test then the test fails.

188 Test 3: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g. MBR file system):

- Record the logical (e.g. LBA) storage location of the key in the TOE subject to clearing.
- Cause the TOE to perform a normal cryptographic processing with the key from the previous step.
- Cause the TOE to clear the key.
- Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

3.1.2 Cryptographic Operation (FCS_COP)

3.1.2.1 FCS_COP.1(a)/Server Cryptographic Operation (Signature Generation and Verification) (Server Communications)

3.1.2.1.1 TSS

189 The evaluator shall verify the TSS includes a description of the cryptographic algorithms and corresponding parameters (key size, etc.) used to generate signatures. The evaluator should ensure that the specified algorithms are listed in SFRs and described properly in TSS.

3.1.2.1.2 Operational Guidance

190 There are no AGD evaluation activities for this SFR.

3.1.2.1.3 KMD

191 There are no KMD evaluation activities for this SFR.

3.1.2.1.4 Test

192 The evaluator shall perform the following tests, depending on the selections made by the ST author:

ECDSA Algorithm Tests

193 ECDSA FIPS 186-4 Signature Generation Test

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

195 ECDSA FIPS 186-4 Signature Verification Test

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

197 RSA Signature Algorithm Tests

198 Signature Generation Test

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

200 The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

201 Signature Verification Test

202 The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

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

3.1.2.2 FCS_COP.1(b)/Server Cryptographic Operation (Hash Algorithm) (Server Communications)

3.1.2.2.1 TSS

204 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.

3.1.2.2.2 Operational Guidance

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

3.1.2.2.3 KMD

206 There are no KMD evaluation activities for this SFR.

3.1.2.2.4 Test

207 The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an

indication is given in the following sections for the bit-oriented vs. the byte-oriented test mode.

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

209 Short Messages Test - Bit-oriented Mode

210 The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

211 Short Messages Test - Byte-oriented Mode

212 The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

213 Selected Long Messages Test - Bit-oriented Mode

214 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 8*99*i, where $1 \leq i \leq m/8$. For SHA-512, the length of the i-th message is 1024 + 8*99*i, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

215 Selected Long Messages Test - Byte-oriented Mode

216 The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 99*i, where $1 \leq i \leq m$. For SHA-512, the length of the i-th message is 1024 + 99*i, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

217 Pseudorandomly Generated Messages Test

218 This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

3.1.2.3 FCS_COP.1(c)/Server Cryptographic Operation (Keyed Hash Algorithm) (Server Communications)

3.1.2.3.1 TSS

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

3.1.2.3.2 Operational Guidance

220 There are no AGD evaluation activities for this SFR.

3.1.2.3.3 KMD

221 There are no KMD evaluation activities for this SFR.

3.1.2.3.4 Test

222 For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key using a known good implementation.

3.1.2.4 FCS_COP.1(d)/Server Cryptographic Operation (Key Wrapping) (Server Communications)

3.1.2.4.1 TSS

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

3.1.2.4.2 Operational Guidance

224 There are no AGD evaluation activities for this SFR.

3.1.2.4.3 KMD

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

3.1.2.4.4 Test

226 There are no test evaluation activities for this SFR.

3.1.2.5 FCS_COP.1(e)/Server Cryptographic Operation (Key Transport) (Server Communications)

3.1.2.5.1 TSS

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

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

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

3.1.2.5.2 Operational Guidance

230 There are no AGD evaluation activities for this SFR.

3.1.2.5.3 KMD

231 There are no KMD evaluation activities for this SFR.

3.1.2.5.4 Test

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

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

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

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

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

4. The evaluator shall check the format of the 'X' string recovered in OAEP.Test.3 to ensure that the format is of the form PS || 01 || K, where PS consists of zero or more consecutive 0x00 bytes and K is the transported keying material. The evaluator should[shall?] also feed into the TOE's RSA-OEAP decryption operation some cipher texts for which the resulting 'X' strings do not have the

correct format (i.e., the leftmost non-zero byte is not 0x01). These incorrectly formatted ‘X’ variables should[shall?] be detected by the RSA-OEAP decrypt function.

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

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

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

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

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

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

3.1.2.6 FCS_COP.1(f)/Server Cryptographic Operation (AES Data Encryption/Decryption) (Server Communications)

3.1.2.6.1 TSS

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

3.1.2.6.2 Operational Guidance

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

3.1.2.6.3 KMD

237 There are no KMD evaluation activities for this SFR.

3.1.2.6.4 Test

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

239 AES-CBC Tests

240 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.

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

242 AES-CBC Known Answer Tests

243 KAT-1 (GFSBox):

244 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.

245 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.

246 KAT-2 (KeySBox):

247 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.

248 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.

249 KAT-3 (Variable Key):

250 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.

251 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.

252 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.

253 KAT-4 (Variable Text):

254 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.

255 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.

256 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.

257 AES-CBC Multi-Block Message Test

258 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.

259 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.

260 AES-CBC Monte Carlo Tests

261 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.

262 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 = 1 to 100 {
Output Key[i], IV[i], PT[0]
for j = 1 to 1000 {
if j == 1 {
                CT[1] = AES-CBC-Encrypt(Key[i], IV[i], PT[1])
                PT[2] = IV[i]
        } else {
                CT[j] = AES-CBC-Encrypt(Key[i], PT[j])
                PT[j+1] = CT[j-1]
        }
 }
Output CT[1000]
If KeySize == 128 { Key[i+1] = Key[i] xor CT[1000] }
If KeySize == 256 { Key[i+1] = Key[i] xor ((CT[999] << 128) | CT[1000]) }
IV[i+1] = CT[1000]
PT[0] = CT[999]
 }
```

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

264 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.

265 AES-GCM Test

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

267 128 bit and 256 bit keys

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

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

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

271 The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the

ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

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

273 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.

274 XTS-AES Test

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

276 256 bit (for AES-128) and 512 bit (for AES-256) keys

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

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

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

280 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.

3.1.2.7 FCS_COP.1(g)/Server Cryptographic Operation (Key Encryption) (Server Communications)

3.1.2.7.1 TSS

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

3.1.2.7.2 Operational Guidance

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

3.1.2.7.3 KMD

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

3.1.2.7.4 Test

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

3.1.3 Random Bit Generation (FCS_RBG_EXT)

3.1.3.1 FCS_RBG_EXT.1/Server Random Bit Generation (Server Communications)

3.1.3.1.1 TSS

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

3.1.3.1.2 Operational Guidance

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

3.1.3.1.3 KMD

287 There are no KMD evaluation activities for this SFR.

3.1.3.1.4 Test

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

289 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).

290 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.

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

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

293 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.

294 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.

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

3.1.4 Salt, Nonce, and Initialization Vector Generation (FCS_SNI_EXT)

3.1.4.1 FCS_SNI_EXT.1/Server Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) (Server Communications)

3.1.4.1.1 TSS

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

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

3.1.4.1.2 Operational Guidance

298 There are no AGD evaluation activities for this SFR.

3.1.4.1.3 KMD

299 There are no KMD evaluation activities for this SFR.

3.1.4.1.4 Test

300 There are no ATE evaluation activities for this SFR.

3.2 Identification and Authentication (FIA)

3.2.1 Authentication Using X.509 Certificates

3.2.1.1 FIA_X509_EXT.1/Server X.509 Certificate Validation (Server Communications)

3.2.1.1.1 TSS

301 The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

3.2.1.1.2 Operational Guidance

302 There are no AGD evaluation activities for this SFR.

3.2.1.1.3 KMD

303 There are no KMD evaluation activities for this SFR.

3.2.1.1.4 Test

304 The evaluator shall perform the following tests:

305 Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.

306 Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

307 Test 3: The evaluator shall test that the TOE can properly handle revoked certificates--–conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the TOE certificate and revocation of the TOE intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

308 Test 4: If OCSP is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the CRLsign key usage bit set, and verify that validation of the CRL fails.

309 Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

310 Test 6: The evaluator shall modify any bit in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

311 Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

312 The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Server. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Server. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

313 The evaluator shall create a chain of at least three certificates: the node certificate to be tested, one intermediate CAs, and the self-signed Root CA.

314 Test 1: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. Verify the validation of the certificate path fails.

315 Test 2: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to FALSE. Verify the validation of the certificate path fails.

316 Test 3: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

3.2.1.2 FIA_X509_EXT.2/Server X.509 Certificate Authentication (Server Communications)

3.2.1.2.1 TSS

317 The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use.

318 The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described.

3.2.1.2.2 Operational Guidance

319 The evaluator shall verify the instructions in the AGD describe how to configure the operating environment so that the TOE can use the certificates. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed.

3.2.1.2.3 KMD

320 There are no KMD evaluation activities for this SFR.

3.2.1.2.4 Test

321 The evaluator shall run the following test for each trusted channel:

322 The evaluator shall demonstrate that using a valid certificate requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2/Server is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

3.2.1.3 FIA_X509_EXT.3/Server X.509 Certificate Requests (Server Communications)

3.2.1.3.1 TSS

323 If the ST author selects "device-specific information", the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests

3.2.1.3.2 Operational Guidance

324 The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request Message. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the certificate request message.

3.2.1.3.3 KMD

325 There are no KMD evaluation activities for this SFR.

3.2.1.3.4 Test

326 The evaluator shall perform the following tests:

327 Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a certificate request message. The evaluator shall capture the generated message and ensure that it conforms to the format specified. The evaluator shall confirm that the certificate request provides the public key and other required information, including any necessary user-input information.

328 Test 2: The evaluator shall demonstrate that validating a certificate response message without a valid certification path results in the function failing. The evaluator shall then load the Trusted Root CA certificate and certificates of all intermediate CAs comprising the validation path for the certificate received in the response message, and demonstrate that the function succeeds. The evaluator shall then delete or invalidate one of loaded the certificates, and show that the function fails.

3.3 Security Management (FMT)

3.3.1 Management of Functions in TSF (FMT_MOF)

3.3.1.1 FMT_MOF.1/Server Management of Functions Behavior (Management Server)

3.3.1.1.1 TSS

329 If support for configuring the encryption algorithms and/or key sizes are claimed in the ST, the evaluator shall ensure the TSS describes how these are configured and shall ensure that TSS describes how only privileged users (administrators) are allowed to manage the states.

3.3.1.1.2 Operational Guidance

330 The evaluator to check if guidance documentation describes which authorization factors are required to change encryption algorithms and/or key sizes.

3.3.1.1.3 KMD

331 There are no KMD evaluation activities for this SFR.

3.3.1.1.4 Test

332 The evaluator shall perform the following tests:

333 Test 1: The evaluator presents a privileged authorization credential to the TSF and validates that changes to encryption algorithm or key sizes are allowed.

334 Test 2: The evaluator presents a non-privileged authorization credential to the TSF and validates that changes to encryption algorithms are not allowed.

4 Evaluation Activities for Selection-Based Requirements

4.1 Cryptographic Support (FCS)

4.1.1 Cryptographic Key Management (FCS_CKM)

4.1.1.1 FCS_CKM.1(b)/Server Cryptographic Key Generation (Symmetric Keys) (Server Communications)

4.1.1.1.1 TSS

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

4.1.1.1.2 Operational Guidance

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

4.1.1.1.3 KMD

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

4.1.1.1.4 Test

338 There are no test evaluation activities for this SFR.

4.1.2 HTTPS Protocol (FCS_HTTPS_EXT)

4.1.2.1 FCS_HTTPS_EXT.1 HTTPS Protocol

4.1.2.1.1 TSS

339 There are no TSS evaluation activities for this SFR.

4.1.2.1.2 Operational Guidance

340 There are no AGD evaluation activities for this SFR.

4.1.2.1.3 KMD

341 There are no KMD evaluation activities for this SFR.

4.1.2.1.4 Test

342 The evaluator shall perform the following tests:

343 Test 1: The evaluator shall attempt to establish an HTTPS connection with the EM server, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

344 Other tests are performed in conjunction with the evaluation activities performed for FCS_TLSC_EXT.1.

345 Validity of the server certificate shall be tested in accordance with testing performed for FIA_X509_EXT.1/Server.

4.1.3 IPsec Protocol (FCS_IPSEC_EXT)

4.1.3.1 FCS_IPSEC_EXT.1 IPsec Protocol

4.1.3.1.1 TSS

346 FCS_IPSEC_EXT.1.1

347 The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

348 As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

349 FCS_IPSEC_EXT.1.3

350 The evaluator checks the TSS to ensure it states that the VPN can be established to operate in transport mode and/or tunnel mode (as identified in FCS_IPSEC_EXT.1.3).

351 FCS_IPSEC_EXT.1.4

352 The evaluator shall examine the TSS to verify that the algorithms AES-CBC-128 and AES-CBC-256 are implemented. If the ST author has selected either AES-GCM-128 or AES-GCM-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(c) Cryptographic Operations (for keyed-hash message authentication).

353 FCS_IPSEC_EXT.1.5

354 The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

355 For IKEv1 implementations, the evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

356 FCS_IPSEC_EXT.1.6

357 The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms AESCBC- 128, AES-CBC-256 are specified, and if others are chosen in the selection of the requirement, those are included in the TSS discussion.

358 FCS_IPSEC_EXT.1.7

359 The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 1 SA lifetime and/or the IKEv2 SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

360 FCS_IPSEC_EXT.1.8

361 The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 2 SA lifetime and/or the IKEv2 Child SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

362 FCS_IPSEC_EXT.1.9

363 The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x". The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" meets the stipulations in the requirement.

364 FCS_IPSEC_EXT.1.11

365 The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

366 FCS_IPSEC_EXT.1.12

367 The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

368 FCS_IPSEC_EXT.1.13

369 The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(a) Cryptographic Operations (for cryptographic signature).

370 If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

371 FCS_IPSEC_EXT.1.14

372 The evaluator shall verify that the TSS describes how the DN in the certificate is compared to the expected DN.

4.1.3.1.2 Operational Guidance

373 FCS_IPSEC_EXT.1.1

374 The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases – a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

375 FCS_IPSEC_EXT.1.3

376 The evaluator shall confirm that the guidance documentation contains instructions on how to configure the connection in each mode selected.

377 FCS_IPSEC_EXT.1.4

378 The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms, and if either AES-GCM-128 or AES-GCM-256 have been selected the guidance instructs how to use these as well.

379 FCS_IPSEC_EXT.1.5

380 The evaluator shall check the guidance documentation to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the following test (if selected).

381 If the IKEv1 Phase 1 mode requires configuration of the TOE prior to its operation, the evaluator shall check the guidance documentation to ensure that instructions for this configuration are contained within that guidance.

382 FCS_IPSEC_EXT.1.6

383 The evaluator ensures that the guidance documentation describes the configuration of the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE to perform the following test for each ciphersuite selected.

384 FCS_IPSEC_EXT.1.7

385 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 1 SA values for 24 hours.

386 Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

387 FCS_IPSEC_EXT.1.8

388 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 2 SA values for 8 hours.

389 Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

390 FCS_IPSEC_EXT.1.11

391 The evaluator ensures that the guidance documentation describes the configuration of the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE to perform the following test for each ciphersuite selected.

392 FCS_IPSEC_EXT.1.13

393 The evaluator ensures the guidance documentation describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

394 The evaluator shall check that the guidance documentation describes how pre-shared keys are to be generated and established. The description in the guidance documentation shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

395 In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE and marked “trusted”.

396 FCS_IPSEC_EXT.1.14

397 The evaluator shall ensure that the guidance documentation includes configuration of the expected DN for the connection.

4.1.3.1.3 KMD

398 There are no KMD evaluation activities for this SFR.

4.1.3.1.4 Test

399 FCS_IPSEC_EXT.1.1

400 The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

401 Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and guidance documentation. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation.

402 FCS_IPSEC_EXT.1.2

403 The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

404 The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

405 Test 1 The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator created entries (there may be a “TOE created” final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was dropped.

406 FCS_IPSEC_EXT.1.3

407 The evaluator shall perform the following test(s) based on the selections chosen:

408 Test 1 (conditional): If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

409 Test 2: The evaluator uses the guidance documentation to configure the TOE to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

410 FCS_IPSEC_EXT.1.4

411 The evaluator shall configure the TOE as indicated in the guidance documentation configuring the TOE to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

412 FCS_IPSEC_EXT.1.5

413 Tests are performed in conjunction with the other IPsec evaluation activities.

414 Test 1 (conditional): The evaluator shall configure the TOE as indicated in the guidance documentation, and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.

415 Test 2 (conditional): The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

416 FCS_IPSEC_EXT.1.6

417 The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

418 FCS_IPSEC_EXT.1.7

419 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

420 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

421 Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

422 Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that once 24 hours has elapsed, a new Phase 1 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

423 FCS_IPSEC_EXT.1.8

424 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

425 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

426 Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

427 Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once 8 hours has elapsed, a new Phase 2 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

428 FCS_IPSEC_EXT.1.10

429 Test 1 (conditional): If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

430 Test 2 (conditional): If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

431 FCS_IPSEC_EXT.1.11

432 For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

433 FCS_IPSEC_EXT.1.12

434 The evaluator simply follows the guidance to configure the TOE to perform the following tests:

435 Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

436 Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

437 Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

438 Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

439 FCS_IPSEC_EXT.1.13

440 For efficiency sake, the testing that is performed may be combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1. The following tests shall be repeated for each peer authentication selected in the FCS_IPSEC_EXT.1.1 selection above:

441 Test 1: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.

442 Test 2 (conditional): The evaluator shall generate a pre-shared key off-TOE and use it, as indicated in the guidance documentation, to establish an IPsec connection with the peer.

443 FCS_IPSEC_EXT.1.14

444 The evaluator shall, if necessary, configure the expected DN according to the guidance documentation. The evaluator shall send a peer certificate signed by a trusted CA with a DN that does not match an expected DN and verify that the TOE denies the connection.

4.1.4 Cryptographic Key Derivation (FCS_KDF_EXT)

4.1.4.1 FCS_KDF_EXT.1/Server Cryptographic Key Derivation (Management Server)

4.1.4.1.1 TSS

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

4.1.4.1.2 Operational Guidance

446 There are no AGD evaluation activities for this SFR.

4.1.4.1.3 KMD

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

4.1.4.1.4 Test

448 There are no test evaluation activities for this SFR.

4.1.5 Cryptographic Construct and Conditioning (FCS_PCC_EXT)

4.1.5.1 FCS_PCC_EXT.1/Server Cryptographic Password Construct and Conditioning (Management Server)

4.1.5.1.1 TSS

449 The evaluator shall ensure the TSS describes the manner in which the TOE enforces the construction of passwords, including the length, and requirements on characters (number and type). The evaluator also verifies that the TSS provides a description of how the password is conditioned and the evaluator ensures it satisfies the requirement.

4.1.5.1.2 Operational Guidance

450 There are no AGD evaluation activities for this SFR.

4.1.5.1.3 KMD

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

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

4.1.5.1.4 Test

453 The evaluator shall perform the following tests:

454 Test 1: Ensure that the TOE supports passwords/passphrases of a minimum length of 64 characters.

455 Test 2: If the TOE supports a password/passphrase length up to a maximum number of characters, n (which would be greater than 64), then ensure that the TOE will not accept more than n characters.

456 Test 3: Ensure that the TOE supports passwords consisting of all characters assigned and supported by the ST author.

4.1.6 SSH Client Protocol (FCS_SSHC_EXT)

4.1.6.1 FCS_SSHC_EXT.1 SSH Client Protocol

4.1.6.1.1 TSS

457 FCS_SSHC_EXT.1.2

458 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHC_EXT.1.5, and ensure that password-based authentication methods are also allowed.

459 FCS_SSHC_EXT.1.3

460 The evaluator shall check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

461 FCS_SSHC_EXT.1.4

462 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component

463 FCS_SSHC_EXT.1.5

464 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

465 FCS_SSHC_EXT.1.6

466 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

467 FCS_SSHC_EXT.1.7

468 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

4.1.6.1.2 Operational Guidance

469 FCS_SSHC_EXT.1.4

470 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

471 FCS_SSHC_EXT.1.5

472 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

473 FCS_SSHC_EXT.1.6

474 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the "none" MAC algorithm is not allowed).

475 FCS_SSHC_EXT.1.7

476 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

4.1.6.1.3 KMD

477 There are no KMD evaluation activities for this SFR.

4.1.6.1.4 Test

478 FCS_SSHC_EXT.1.2

479 Test 1: The evaluator shall, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection to an SSH server. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.

480 Test 2: Using the guidance documentation, the evaluator shall configure the TOE to perform password-based authentication to an SSH server, and demonstrate that a user can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.

481 FCS_SSHC_EXT.1.3

482 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

483 FCS_SSHC_EXT.1.4

484 Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

485 Test 2: The evaluator shall configure an SSH server to only allow the 3descbc encryption algorithm and no other encryption algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

486 FCS_SSHC_EXT.1.5

487 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

488 Test 2: The evaluator shall configure an SSH server to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

489 FCS_SSHC_EXT.1.6

490 Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

491 Test 2: The evaluator shall configure an SSH server to only allow the “none” MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

492 Test 3: The evaluator shall configure an SSH server to only allow the hmacmd5 MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

493 FCS_SSHC_EXT.1.7

494 Test 1: The evaluator shall configure an SSH server to permit all allowed key exchange methods. The evaluator shall attempt to connect from the TOE to the SSH server using each allowed key exchange method, and observe that each attempt succeeds.

495 FCS_SSHC_EXT.1.8

496 The evaluator shall configure the TOE to create a log entry when a rekey occurs. The evaluator shall connect to the TOE with an SSH client and cause 2^28 packets to be transmitted from the client to the TOE, and subsequently review the audit log to ensure that a rekey occurred.

497 FCS_SSHC_EXT.1.9

498 Test 1: The evaluator shall delete all entries in the TOE’s list of recognized SSH server host keys and, if selected, all entries in the TOE’s list of trusted certification authorities. The evaluator shall initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server’s public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the user to accept or deny the key before continuing the connection.

499 Test 2: The evaluator shall add an entry associating a host name with a public key into the TOE’s local database. The evaluator shall replace, on the corresponding SSH server, the server’s host key with a different host key. The evaluator shall initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords).

4.1.7 SSH Server Protocol (FCS_SSHS_EXT)

4.1.7.1 FCS_SSHS_EXT.1 SSH Server Protocol

4.1.7.1.1 TSS

500 FCS_SSHS_EXT.1.2

501 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHS_EXT.1.5, and ensure that password-based authentication methods are also allowed.

502 FCS_SSHS_EXT.1.3

503 The evaluator shall check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

504 FCS_SSHS_EXT.1.4

505 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

506 FCS_SSHS_EXT.1.5

507 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

508 FCS_SSHS_EXT.1.6

509 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

510 FCS_SSHS_EXT.1.7

511 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

4.1.7.1.2 Operational Guidance

512 FCS_SSHS_EXT.1.4

513 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

514 FCS_SSHS_EXT.1.5

515 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS

(for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

516 FCS_SSHS_EXT.1.6

517 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the “none” MAC algorithm is not allowed).

518 FCS_SSHS_EXT.1.7

519 The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

4.1.7.1.3 KMD

520 There are no KMD evaluation activities for this SFR.

4.1.7.1.4 Test

521 FCS_SSHS_EXT.1.2

522 Test 1: The evaluator shall, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.

523 Test 2: The evaluator shall choose one public key algorithm supported by the TOE. The evaluator shall generate a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

524 Test 3: Using the guidance documentation, the evaluator shall configure the TOE to accept password-based authentication, and demonstrate that a user can be successfully authenticated to the TOE over SSH using a password as an authenticator.

525 Test 4: The evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

526 FCS_SSHS_EXT.1.3

527 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

528 FCS_SSHS_EXT.1.4

529 Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

530 Test 2: The evaluator shall configure an SSH client to only allow the 3descbc encryption algorithm and no other encryption algorithms. The evaluator shall attempt

to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

531 FCS_SSHS_EXT.1.5

532 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate the TOE to an SSH client. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

533 Test 2: The evaluator shall configure an SSH client to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

534 FCS_SSHS_EXT.1.6

535 Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

536 Test 2: The evaluator shall configure an SSH client to only allow the “none” MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

537 Test 3: The evaluator shall configure an SSH client to only allow the hmacmd5 MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

538 FCS_SSHS_EXT.1.7

539 Test 1: The evaluator shall configure an SSH client to only allow the diffiehellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

540 Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

541 FCS_SSHS_EXT.1.8

542 The evaluator shall configure the TOE to create a log entry when a rekey occurs. The evaluator shall connect to the TOE with an SSH client and cause 2^28 packets to be transmitted from the client to the TOE, and subsequently review the audit log to ensure that a rekey occurred.

4.1.8 TLS Client Protocol (FCS_TLSC_EXT)

4.1.8.1 FCS_TLSC_EXT.1 TLS Client Protocol

4.1.8.1.1 TSS

543 FCS_TLSC_EXT.1.1

544 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

545 FCS_TLSC_EXT.1.2

546 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application configured reference identifier, including which types of reference identifiers are supported (e.g. Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE. The TLS shall indicate if TLS_PSK is used with the PSK-identity as a reference identifier.

547 FCS_TLSC_EXT.1.4

548 The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behaviour is performed by default or may be configured.

4.1.8.1.2 Operational Guidance

549 FCS_TLSC_EXT.1.1

550 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

551 FCS_TLSC_EXT.1.2

552 The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

553 FCS_TLSC_EXT.1.4

554 If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

4.1.8.1.3 KMD

555 There are no KMD evaluation activities for this SFR.

4.1.8.1.4 Test

556 Note that if the TSF includes FCS_TLSC_EXT.3, some tests here may not be applicable based on selections made in that SFR. The evaluator shall reference FCS_TLSC_EXT.3 in order to determine the TLS tests that are relevant to the TOE based on the claims made.

557 FCS_TLSC_EXT.1.1

558 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is

sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

559 Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field. The test is not applicable if TLS_PSK is selected.

560 Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA256 ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message. The test is not applicable if TLS_PSK is selected.

561 Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS_TLSS_EXT.1.1 or FCS_TLSS_EXT.2.1 can be used as a substitute for this test.

562 Test 5: The evaluator performs the following modifications to the traffic:

563 a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.

564 b) Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

565 c) Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

566 d) Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.

567 e) Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.

568 f) Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

569 FCS_TLSC_EXT.1.2

570 The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection (the tests are not applicable if TLS_PSK is selected):

571 Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.

572 Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

573 Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.

574 Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

575 Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

576 a) The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

577 b) The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

578 Test 6 (conditional): If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

579 Test 7 (conditional): If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

580 FCS_TLSC_EXT.1.3

581 Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates, and show that the certificate is not validated and the trusted channel is not established. The test is not applicable if TLS_PSK is selected.

582 FCS_TLSC_EXT.1.4

583 Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

4.1.8.2 FCS_TLSC_EXT.3 TLS Client Handshake Message Exchange

4.1.8.2.1 TSS

584 The evaluator shall verify that when the reduced handshake is selected with TLS-PSK, the TSS describes the difference against the full handshake and demonstrates the security of communication is not lowered. The evaluator shall verify that the reduced handshake is used within FPT_ITT.1 only.

4.1.8.2.2 Operational Guidance

585 There are no AGD evaluation activities for this SFR.

4.1.8.2.3 KMD

586 There are no KMD evaluation activities for this SFR.

4.1.8.2.4 Test

587 If the reduced TLS handshake message exchange is selected then the following tests in FCS_TLSC_EXT.1 are not applicable:

588 FCS_TLSC_EXT.1.1 Test 5: tests d) and f).

4.1.9 TLS Server Protocol (FCS_TLSS_EXT)

4.1.9.1 FCS_TLSS_EXT.1 TLS Server Protocol

4.1.9.1.1 TSS

589 FCS_TLSS_EXT.1.1

590 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

591 FCS_TLSS_EXT.1.2

592 The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

593 FCS_TLSS_EXT.1.3

594 The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

4.1.9.1.2 Operational Guidance

595 FCS_TLSS_EXT.1.1

596 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

597 FCS_TLSS_EXT.1.2

598 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

599 FCS_TLSS_EXT.1.3

600 The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

4.1.9.1.3 KMD

601 There are no KMD evaluation activities for this SFR.

4.1.9.1.4 Test

602 Note that if the TSF includes FCS_TLSC_EXT.3, some tests here may not be applicable based on selections made in that SFR. The evaluator shall reference FCS_TLSC_EXT.3 in order to determine the TLS tests that are relevant to the TOE based on the claims made.

603 FCS_TLSS_EXT.1.1

604 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

605 Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the

606 TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

607 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA256 ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

608 Test 4: The evaluator shall perform the following modifications to the traffic:

609 a) Modify at a byte in the client's nonce in the Client Hello handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using

mutual authentication) or that the server denies the client's Finished handshake message.

610 b) Modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

611 c) Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

612 d) After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.

613 e) Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

614 FCS_TLSS_EXT.1.2

615 The evaluator shall send a Client Hello requesting a connection with version SSL 1.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 2.0, SSL 3.0, TLS 1.0, and any selected TLS versions.

616 FCS_TLSS_EXT.1.3

617 The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

4.1.9.2 FCS_TLSS_EXT.3 TLS Server Handshake Message Exchange

4.1.9.2.1 TSS

618 The evaluator shall verify that when the reduced handshake is selected with TLS-PSK, the TSS describes the difference against the full handshake and demonstrates that the security of communication is not lowered. The evaluator shall verify that the reduced handshake is used within FPT_ITT.1 only.

4.1.9.2.2 Operational Guidance

619 There are no AGD evaluation activities for this SFR.

4.1.9.2.3 KMD

620 There are no KMD evaluation activities for this SFR.

4.1.9.2.4 Test

621 If the reduced TLS handshake message exchange is selected then the following tests are not applicable:

622 FCS_TLSC_EXT.1.1 Test 4: tests b), d), and e);

623 FCS_TLSS_EXT.1.3

4.1.10 Validation of Cryptographic Elements (FCS_VAL_EXT)

4.1.10.1 FCS_VAL_EXT.2 User Validation

4.1.10.1.1 TSS

624 The evaluator shall examine the TSS to determine which component of the Operational Environment is used to assert the User's identity. The evaluator shall examine the TSS to determine how the TOE responds to an assertion by the Operational Environment. The evaluator shall examine the TSS to verify that it describes how validation is performed. The evaluator shall verify the TSS ensures that the validation process does not expose any material that might compromise key material.

4.1.10.1.2 Operational Guidance

625 The evaluator shall examine the operational guidance to ensure it describes how to configure the TOE and Operating Environment to enable the OE to provide User identity assertions to the TOE.

626 (conditional) If the number of User authentication attempts is configurable in the TOE, the examiner shall examine the operational guidance to ensure it describes how to configure the TOE.

4.1.10.1.3 KMD

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

4.1.10.1.4 Test

628 The evaluator shall perform the following tests:

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

630 Test 2: For each validated authorization factor, ensure that when the user provides an incorrect authorization factor, the TOE prevents the BEV from being forwarded outside the TOE (e.g., to the EE).

4.2 Identification and Authentication (FIA)

4.2.1 Challenge/Response Recovery Credential (FIA_CHR_EXT)

4.2.1.1 FIA_CHR_EXT.1 Challenge/Response Recovery Credential

4.2.1.1.1 TSS

631 The evaluator shall examine the TSS to determine that the methods requesting a Recovery credential are specified. The TSS shall also describe the methods used to verify user or both user and device requesting the Recovery credential. The evaluator shall also verify that the TSS contains the estimation of the strength of the ephemeral response and that it has at least as many potential values as a corresponding password or PIN.

4.2.1.1.2 Operational Guidance

632 The evaluator shall confirm that the guidance documentation contains instructions for enforcing verification of the user or both user and device for which the Recovery is requested. The guidance shall also describe configuring of the limit for consecutive failed validation attempts if this value is configurable.

4.2.1.1.3 KMD

633 There are no KMD evaluation activities for this SFR.

4.2.1.1.4 Test

634 The evaluator shall ensure that a response is only generated if the user or both the user and device for which recovery is requested are verified as specified in TSS. The evaluator shall also ensure that the response is applicable only on behalf of the requesting user and on the device where the challenge was generated with the constraints specified for consecutive failed authentication attempts.

635 The term “managed” below is used to refer a user or device which is registered on the server, i.e. their identity can be successfully verified by either administrator or TSF. The “unmanaged” presumes that the user/device cannot be successfully verified.

636 The evaluator shall perform the following tests:

637 Test 1: The evaluator shall configure the Challenge/Response recovery to validate the user and device. The evaluator shall then issue a challenge on behalf of a managed user for a managed device and ensure that TSF successfully generates the response.

638 Test 2: The evaluator shall configure the Challenge/Response recovery to validate the user. The evaluator shall then issue a challenge on behalf of managed User A and attempt to use it as an unmanaged User B to obtain a response. This should fail.

639 Test 3: The evaluator shall configure the Challenge/Response recovery to validate the device. The evaluator shall then issue a challenge on behalf of managed User A for an unmanaged device or a managed device that User A has no access to and then attempt to use it to obtain a response. This should fail.

640 Test 4: The evaluator shall issue a challenge on behalf of a managed user from a specific device and ensure that the response received successfully will log the user in on that device.

641 Test 5: The evaluator shall attempt to reuse the response of User A with User B on the same system and it should fail.

642 Test 6: The evaluator shall issue a challenge on behalf of a managed user from a specific device, and then attempt to reuse the response on a different device. This should fail.

643 Test 7: The evaluator shall issue a challenge on behalf of a managed user from a managed system, reboot the system [system terminates the session] and enter the response. This should fail.

644 Test 8: The evaluator shall issue a challenge on behalf of a managed user from a managed system and attempt to enter an incorrect response on the system the number of times described in the Guidance Documents. The observed behavior shall conform to the assignments/selections in FIA_CHR_EXT.1.5 and FIA_CHR_EXT.1.6.

4.2.2 PIN Recovery Credential (FIA_PIN_EXT)

4.2.2.1 FIA_PIN_EXT.1 PIN Recovery Credential

4.2.2.1.1 TSS

645 The evaluator shall examine the TSS to determine that the methods using a PIN Recovery Credential are specified.

4.2.2.1.2 Operational Guidance

646 The evaluator shall confirm that the guidance documentation contains instructions for establishing Recovery PIN credential for each hard drive or set of drives on the Management Server.

4.2.2.1.3 KMD

647 There are no KMD evaluation activities for this SFR.

4.2.2.1.4 Test

648 Test 1: The evaluator shall populate the Management Server with a Recovery PIN and then try to retrieve it. The test shall pass. The evaluation shall remove the Recovery PIN and try to retrieve it. This test shall fail.

649 Test 2: The evaluator shall use the retrieved Recovery PIN to authenticate to the system where the drive or set of the drives protected by the Recovery PIN is installed. This test shall pass. The evaluator shall then try to use the Recovery PIN on a device with different drive(s). This test shall fail.

650 Test 3: After successful authentication with the Recovery PIN the evaluator shall restart the system and try to authenticate again with the same PIN. This should fail.

4.2.3 Support for Recovery Credentials (FIA_REC_EXT)

4.2.3.1 FIA_REC_EXT.1 Support for Recovery Credentials

4.2.3.1.1 TSS

651 The evaluator shall examine the TSS to determine that types of supported recovery credential are specified.

4.2.3.1.2 Operational Guidance

652 The evaluator shall confirm that the guidance documentation contains instructions for turning off the ability of the server to return a recovery credential is specified.

4.2.3.1.3 KMD

653 There are no KMD evaluation activities for this SFR.

4.2.3.1.4 Test

654 The evaluator shall disable the ability of a server to return a recovery credential. The evaluator should then attempt to obtain the recovery credential and this should fail.

5 Evaluation Activities for SARs

655 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.

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

5.1 ASE: Security Target Evaluation

657 An evaluation activity is defined here for evaluation of Exact Conformance claims against a cPP in a Security Target. Other aspects of ASE remain as defined in [CEM, 10].

5.1.1 Conformance Claims (ASE_CCL.1)

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

Table 1: ASE_CCL.1 Exact Conformance Actions

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

5.2 Development (ADV)

5.2.1 Basic Functional Specification (ADV_FSP.1)

659 The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces,

network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Sections 2 through 4, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

660 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.

661 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.

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

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

CEM ADV_FSP.1 Work Units	***Evaluation Activities***
ADV_FSP.1-1 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each SFR-supporting and SFR-enforcing TSFI.	Evaluation Activity: The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.
ADV_FSP.1-2 The evaluator ***shall examine*** the functional specification to determine that the method of use for each SFR-supporting and SFR-enforcing TSFI is given.	Evaluation Activity: The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.
ADV_FSP.1-3 The evaluator ***shall examine*** the presentation of the TSFI to determine that it identifies all parameters associated with each SFR-enforcing and SFR supporting TSFI.	Evaluation Activity: *The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.*
ADV_FSP.1-4 The evaluator ***shall examine*** the rationale provided by the developer for the implicit categorisation of interfaces as SFR-non-interfering to determine that it is accurate.	Paragraph 561 from the CEM: "In the case where the developer has provided adequate documentation to perform the analysis called for by the rest of the work units for this component without explicitly identifying SFR-enforcing and SFR-

	supporting interfaces, this work unit should be considered satisfied." Since the rest of the ADV_FSP.1 work units will have been satisfied upon completion of the EAs, it follows that this work unit is satisfied as well.
ADV_FSP.1-5 The evaluator ***shall check*** that the tracing links the SFRs to the corresponding TSFIs.	Evaluation Activity: *The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.*
ADV_FSP.1-6 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the SFRs.	EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3 and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e., at the TSFI) are covered. Therefore, the intent of this work unit is covered.
ADV_FSP.1-7 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the SFRs.	EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3 and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e., at the TSFI) are addressed, and that the description of the interfaces is accurate with respect to the specification captured in the SFRs. Therefore, the intent of this work unit is covered.

5.2.1.1 Evaluation Activity

663 *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.*

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

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

5.2.1.2 Evaluation Activity

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

667 The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Sections 2 through 4, including the EAs associated with testing of the interfaces.

668 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.

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

5.3 Guidance Documents (AGD)

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

5.3.1 Operational User Guidance (AGD_OPE.1)

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

5.3.1.1 Evaluation Activity

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

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

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

675 The contents of the operational guidance will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in Sections 2 through 4.

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

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

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

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

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

5.3.2 Preparative Procedures (AGD_PRE.1)

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

5.3.2.1 Evaluation Activity

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

679 The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in Sections 2 through 4.

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

681 The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in Sections 2 through 4.

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

683 Preparative procedures must include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target). The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE itself).

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

685 The preparative procedures must include

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

5.4 Life-cycle Support (ALC)

5.4.1 Labelling of the TOE (ALC_CMC.1)

686 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)

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

5.5 Tests (ATE)

5.5.1 Independent Testing – Conformance (ATE_IND.1)

688 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.

689 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 Sections 2 through 4.

5.6 Vulnerability Assessment (AVA)

5.6.1 Vulnerability Survey (AVA_VAN.1)

690 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.

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

Activity. If clarification has been provided, a reference to this clarification is provided in the table.

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

CEM AVA_VAN.1 Work Units	***Evaluation Activities***
AVA_VAN.1-1 The evaluator ***shall examine*** the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.	The evaluator shall perform the CEM activity as specified. *If the iTC specifies any tools to be used in performing this analysis in section A.3.4, the following text is also included in this cell: "The calibration of test resources specified in paragraph 1418 of the CEM applies to the tools listed in Appendix A, Section A.1.4."*
AVA_VAN.1-2 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state	The evaluator shall perform the CEM activity as specified.
AVA_VAN.1-3 The evaluator ***shall examine*** sources of information publicly available to identify potential vulnerabilities in the TOE.	Replace CEM work unit with activities outlined in Appendix A, Section A.1.
AVA_VAN.1-4 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.	Replace the CEM work unit with the analysis activities on the list of potential vulnerabilities in Appendix A, section A.1, and documentation as specified in Appendix A, Section A.3.
AVA_VAN.1-5 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.	Replace the CEM work unit with the activities specified in Appendix A, section A.2.
AVA_VAN.1-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include: a) identification of the potential vulnerability the TOE is being tested for; b) instructions to connect and setup all required test equipment as required to conduct the penetration test; c) instructions to establish all penetration test prerequisite initial conditions;	The CEM work unit is captured in Appendix A, Section A.3; there are no substantive differences.

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

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.	

6 Required Supplementary Information

692 This Supporting Document refers in various places to the possibility that 'supplementary information' may need to be supplied as part of the deliverables for an evaluation. This term is intended to describe information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP.

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

7 References

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

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

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

[FDE-AA] collaborative Protection Profile for Full Disk Encryption – Authorization Acquisition, Version 2.0, 04 January 2018

[FDE-EE] collaborative Protection Profile for Full Disk Encryption – Encryption Engine, Version 2.0, 04 January 2018

[FDE-EM] collaborative Protection Profile Module for Full Drive Encryption – Enterprise Management, Version 2.0, 04 January 2018

Appendixes

A. Vulnerability Analysis

A.1 Sources of Vulnerability Information

694 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.

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

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

1. A list of flaw hypotheses applicable to the technology described by the cPP derived from public sources as documented in Section A.1.1—this fixed set has been agreed to by the iTC. Additionally, this will be supplemented with entries for a set of public sources (as indicated below) that are directly applicable to the TOE or its identified components (as defined by the process in Section A.1.1 below); this is to ensure that the evaluators include in their assessment applicable entries that have been discovered since the cPP was published;
2. A list of flaw hypotheses contained in this document that are derived from lessons learned specific to that technology and other iTC input (that might be derived from other open sources and vulnerability databases, for example) as documented in Section A.1.2;
3. A list of flaw hypotheses derived from information available to the evaluators; this includes the baseline evidence provided by the vendor described in this Supporting Document (documentation associated with EAs, documentation described in Section 5.6.1), as well as other information (public and/or based on evaluator experience) as documented in Section A.1.3; and
4. A list of flaw hypotheses that are generated through the use of iTC-defined tools (e.g., nmap, protocol testers) and their application is specified in section A.1.4.

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

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

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

b. Search the National Vulnerability Database: https://nvd.nist.gov/
c. Search US-CERT http://www.kb.cert.org/vuls/html/search

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

- General (for all)
  - Product name
  - underlying components (e.g., OS, software libraries (crypto libraries), chipsets, databases, hypervisors, cloud environments)
  - drive encryption, disk encryption, data at rest (DAR)
  - key destruction/sanitization
- AA:
  - Underlying components (e.g., smart card libraries)
  - Opal management software, SED management software
  - Password caching
- EE:
  - Underlying components (e.g., chipsets, firmware)
- For SEDs (for EE):
  - Self Encrypting Drive (SED), OPAL
- For Software FDE (AA or EE):
  - Key caching
- For Enterprise Management Server:
  - Enterprise Management
  - Key Recovery
  - Remote Management
  - Remote Recovery
  - Encryption Management
  - SQL Server
  - Active Directory
  - Data at Rest (DAR) Management

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

700 The evaluator will also consider the requirements that are chosen and the appropriate guidance that is tied to each requirement. For example, with FCS_AFA_EXT.1, if the Smartcard selection is chosen, then the evaluator will use the appropriate search terms for smart cards.

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

702 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

703 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:

704 *General:*

705 AA:

- In order to validate the AA is properly encrypting keying material (e.g., BEV, KEK, authorization submasks) in the readable part of the disk (e.g., shadow MBR), the evaluator should examine the disk using a tool to view the drive (e.g. WinHex) to look for material that exposes a key value.

- When an authentication or recovery credential is changed, it is critical that the AA does not leave old keys/key chains/key material around. This process should also be monitored using a tool to view the drive.

706 AA (for ISV's)

- It is possible that preboot authentication appears to function normally and it's possible that the SED could forget to lock the global range, which results in the preboot being locked, but the rest of the drive is unencrypted. This could be tested using a tool (e.g. WinHex) by writing a known pattern, locking the drive and looking for the pattern.

707 EE:

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

708 EM:

- Leaving key material around – it's possible that a user can be removed/deactivated in the Enterprise Management Server, but the server might not clean up all key material associated with that user, thus allowing a user to access their encrypted data. The evaluator should verify this by first creating a user and configuring the product to allow them to access it. Have the user perform a key recovery and then remove or deactivate that user and verify they cannot access the data.

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

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

710 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 warranted.

A.1.4 Type 4 Hypotheses—Tool-Generated

711 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 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

712 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.

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

activity/cPP and cannot provide evidence otherwise that the flaw is disproved, the evaluator prepares an appropriate set of materials as follows:

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

714 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).

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

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

717 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.

718 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).

719 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.

720 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.

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

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

A.3 Reporting

723 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.

724 The public-facing report contains:

725 * 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;

726 * 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.

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

728 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:

  b) its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);

  c) the SFR(s) not met;

  d) a description;

  e) whether it is exploitable in its operational environment or not (i.e. exploitable or residual).

  f) 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);

  g) 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

  h) in the case that actual testing was performed in the investigation (either as part of flaw hypothesis generation using tools specified by the iTC in Section A.1.4, or in proving/disproving a particular flaw) the steps followed in setting up the TOE (and any required test equipment);

executing the test; post-test procedures; and the actual results (to a level of detail that allow repetition of the test, including the following:

- identification of the potential vulnerability the TOE is being tested for;
- instructions to connect and setup all required test equipment as required to conduct the penetration test;
- instructions to establish all penetration test prerequisite initial conditions;
- instructions to stimulate the TSF;
- instructions for observing the behaviour of the TSF;
- descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;
- instructions to conclude the test and establish the necessary post-test state for the TOE. (cf. AVA_VAN.1-6, AVA_VAN.1-8).

B. FDE Equivalency Considerations

729 Introduction

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

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

- **Variations in models:** Separate TOE models/variations may include differences that could necessitate separate testing across each model. If there are no variations in any of the categories listed below, the models may be considered equivalent.

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

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

733 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.

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

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

736 Evaluator guidance for determining equivalence

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

Factor	Same/Not Same	Evaluator Guidance
Platform/Hardware Dependencies	Independent	If there are no identified platform/hardware dependencies, the evaluator shall consider testing on multiple hardware platforms to be equivalent.

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

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

739 When performing the equivalency analysis, the evaluator should consider each factor independently. Each analysis of an individual factor will result in one of two outcomes,

- For the particular factor, all variations of the TOE on all supported platforms are equivalent. In this case, testing may be performed on a single model in a single test environment and cover all supported models and environments.

- For the particular factor, a subset of the TOE has been identified to require separate testing to ensure that it operates identically to all other equivalent TOE. The analysis would identify the specific combinations of models/testing environments that needed to be tested.

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

741 Test presentation/Truth in advertising

742 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.