Supporting Document
Mandatory Technical Document

Evaluation Activities for Network Device
cPP

September-2018

Version 2.1

CCDB-2018-<month TBD>-<number TBD>

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 the Network International Technical Community (NDFW-iTC) and is designed to be used to support the evaluations of products against the cPPs identified in section 1.1.

Technical Editor: Network International Technical Community (NDFW-iTC)

Document history:

V2.1, 17 August 2018 (published version)

V2.0, 5 May 2017 (published version)

V1.1, 21 July 2016 (Updated draft published for public review)

V1.0, 27 February 2015 (published version)

V0.4, 26 January 2015 (incorporates changes due to comments received from CCDB review)

V0.3, 17 October 2014 (released version following public review, submitted for CCDB review)

V0.2, 13 October 2014 (internal draft in response to public review comments, for iTC review)

V0.1, 5 September 2014 (Initial release for public review)

General Purpose: See section 1.1.

Field of special use: This Supporting Document applies to the evaluation of TOEs claiming conformance with the collaborative Protection Profile for Network Devices [NDcPP] and collaborative Protection Profile for Stateful Traffic Filter Firewalls [FWcPP].

Acknowledgements:

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

Table of Contents

Table of contents

List of tables

1 Introduction

1.1 Technology Area and Scope of Supporting Document

1 This Supporting Document defines the Evaluation Activities associated with the collaborative Protection Profile for Network Devices [NDcPP].

2 The Network Device technical area has a number of specialised aspects, such as those relating to the secure implementation and use of protocols, and to the particular ways in which remote management facilities need to be assessed across a range of different physical and logical interfaces for different types of infrastructure devices. This degree of specialisation, and the associations between individual SFRs in the cPP, make it important for both efficiency and effectiveness that evaluation activities are given more specific interpretations than those found in the generic CEM activities.

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

a) collaborative Protection Profile for Network Devices [NDcPP]

b) collaborative Protection Profile for Stateful Traffic Filter Firewalls [FWcPP].

4 Although Evaluation Activities are defined mainly for the evaluators to follow, the definitions in this Supporting Document aim to provide a common understanding for developers, evaluators and users as to what aspects of the TOE are tested in an evaluation against the associated cPPs, and to what depth the testing is carried out. This common understanding in turn contributes to the goal of ensuring that evaluations against the cPP achieve comparable, transparent and repeatable results. In general the definition of Evaluation Activities 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 – see section 6).

1.2 Structure of the Document

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

6 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.

7 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.

8 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 Application of this Supporting Document

9 This Supporting Document (SD) defines three types of Evaluation Activities (EAs) – TOE Summary Specification (TSS), Guidance Documentation, and Tests and is designed to be used in conjunction with cPPs. cPPs that rely on this SD will explicitly identify it as a source for their EAs[1]. Each security requirement (SFR or SAR) specified in the cPP could have multiple EAs associated with it. The security requirement naming convention is consistent between cPP and SD ensuring a clear one to one correspondence between security requirements and evaluation activities.

10 The cPP and SD are designed to be used in conjunction with each other, where the cPP lists SFRs and SARs and the SD catalogues EAs associated with each SFR and SAR. Some of the SFRs included in the cPP are optional or selection-based. Therefore an ST claiming conformance to the cPP does not necessarily have to include all possible SFRs defined in the cPP.

11 In an ST conformant to the cPP, several operations need to be performed (mainly selections and assignments). Some EAs define separate actions for different selected or assigned values in SFRs. The evaluator shall neither carry out EAs related to SFRs that are not claimed in the ST nor EAs related to specific selected or assigned values that are not claimed in the ST.

12 EAs do not necessarily have to be executed independently from each other. A description in a guidance documentation or one test case, for example, can cover multiple EAs at a time, no matter whether the EAs are related to the same or different SFRs.

[1] In general a cPP may reference one or more SDs as sources for the Evaluation Activities for different sets of SFRs.

1.4 Terminology

1.4.1 Glossary

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

Term	Meaning
Administrator	**See Security Administrator**
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
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.
Required Supplementary Information	Information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP (see description in Section 6).
Security Administrator	The terms "Administrator", "Security Administrator", and "User" are used interchangeably in this document at present and are used to represent a person that has authorized access to the TOE to perform configuration and management tasks.
Target of Evaluation	A set of software, firmware and/or hardware possibly accompanied by guidance. [CC1]
TOE Security Functionality (TSF)	A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs. [CC1]
TSF Data	Data for the operation of the TSF upon which the enforcement of the requirements relies.
User	See Security Administrator

1.4.2 Acronyms

Acronym	Meaning
cPP	collaborative Protection Profile
CA	Certificate Authority
CN	Certificate Name
CVE	Common Vulnerabilities and Exposures (database)
DN	Domain Name
DNS	Domain Name Service
EA	Evaluation Activity
ECDHE	Elliptic Curve Diffie-Hellman Key Exchange
iTC	International Technical Community
NIST	National Institute of Standards and Technology
SAN	Storage Area Network
SAR	Security Assurance Requirement
SD	Supporting Document
SSL	Secure Sockets Layer
TLS	Transport Layer Security

2 Evaluation Activities for SFRs

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

15 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.

16 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.

17 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.

Additional Note for Distributed TOEs

18 For a distributed TOE, all examination of Operational Guidance information should be extended to include confirmation that it defines sufficient information to configure individual components such that the overall TOE is correctly established.

19 Evaluation activities for SFRs must be carried out for all distributed TOE components that implement the SFR (as defined in the mapping of SFRs to components – cf. section 5.1.2). This applies to optional and selection-based SFRs in section 3 and 4 as well as to the core SFRs in this section.

2.1 Security Audit (FAU)

2.1.1 FAU_GEN.1 Audit data generation

2.1.1.1 TSS

20 For the administrative task of generating/import of, changing, or deleting of cryptographic keys as defined in FAU_GEN.1.1c, the TSS should identify what information is logged to identify the relevant key.

21 For distributed TOEs the evaluator shall examine the TSS to ensure that it describes which of the overall required auditable events defined in FAU_GEN.1.1 are generated and recorded by which TOE components. The evaluator shall ensure that this mapping of audit events to TOE components accounts for, and is consistent with, information provided in Table 1, as well as events in Tables 2, 4, and 5 (where applicable to the overall TOE). This includes that the evaluator shall confirm that all components defined as generating audit information for a particular SFR should also contribute to that SFR as defined in the mapping of SFRs to TOE components, and that the audit records generated by each component cover all the SFRs that it implements.

2.1.1.2 Guidance Documentation

22 The evaluator shall check the guidance documentation and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the cPP is described and that the description of the fields contains the information required in FAU_GEN1.2, and the additional information specified in the table of audit events.

23 The evaluator shall also make a determination of the administrative actions related to TSF data related to configuration changes. The evaluator shall examine the guidance documentation and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the cPP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are related to TSF data related to configuration changes. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

2.1.1.3 Tests

24 The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for

each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

25 For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of auditable events to TOE components in the Security Target. For all events involving more than one TOE component when an audit event is triggered, the evaluator has to check that the event has been audited on both sides (e.g. failure of building up a secure communication channel between the two components). This is not limited to error cases but includes also events about successful actions like successful build up/tear down of a secure communication channel between TOE components.

26 Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

2.1.2 FAU_GEN.2 User identity association

2.1.2.1 TSS & Guidance Documentation

27 The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

2.1.2.2 Tests

28 This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

29 For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated for building up a secure channel between two TOE components. If for some reason (could be e.g. TSS or Guidance Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

2.1.3 FAU_STG_EXT.1 Protected audit event storage

2.1.3.1 TSS

30 The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.

31 The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally; what happens when the local audit data store is full; and how these records are protected against unauthorized access.

32 The evaluator shall examine the TSS to ensure it describes whether the TOE is a standalone TOE that stores audit data locally or a distributed TOE that stores audit data locally on each TOE component or a distributed TOE that contains TOE components that cannot store audit data locally on themselves but need to transfer audit data to other TOE components that can store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs it contains a list of TOE components that store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs that contain components which do not store audit data locally but transmit their generated audit data to other components it contains a mapping between the transmitting and storing TOE components.

33 The evaluator shall examine the TSS to ensure that it details the behaviour of the TOE when the storage space for audit data is full. When the option 'overwrite previous audit record' is selected this description should include an outline of the rule for overwriting audit data. If 'other actions' are chosen such as sending the new audit data to an external IT entity, then the related behaviour of the TOE shall also be detailed in the TSS.

34 The evaluator shall examine the TSS to ensure that it details whether the transmission of audit information to an external IT entity can be done in real-time or periodically. In case the TOE does not perform transmission in real-time the evaluator needs to verify that the TSS provides details about what event stimulates the transmission to be made as well as the possible as well as acceptable frequency for the transfer of audit data.

35 For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how audit data transfer to the external audit server is implemented among the different TOE components (e.g. every TOE components does its own transfer or the data is sent to another TOE component for central transfer of all audit events to the external audit server).

36 For distributed TOEs the evaluator shall examine the TSS to ensure it describes which TOE components are storing audit information locally and which components are buffering audit information and forwarding the information to another TOE component for local storage. For every component the TSS shall describe the behaviour when local storage space or buffer space is exhausted.

2.1.3.2 Guidance Documentation

37 The evaluator shall also examine the guidance documentation to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

38 The evaluator shall also examine the guidance documentation to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and "cleared" periodically by sending the data to the audit server.

39 The evaluator shall also ensure that the guidance documentation describes all possible configuration options for FAU_STG_EXT.1.3 and the resulting behaviour of the TOE for each possible configuration. The description of possible configuration options and resulting behaviour shall correspond to those described in the TSS.

2.1.3.3 Tests

40 Testing of the trusted channel mechanism for audit will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. The evaluator shall perform the following additional tests for this requirement:

a) Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing. The evaluator shall verify that the TOE is capable of transferring audit data to an external audit server automatically without administrator intervention.

b) Test 2: The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.3. Depending on the configuration this means that the evaluator has to check the content of the audit data when the audit data is just filled to the maximum and then verifies that

    1) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded

again after the local storage for audit data is cleared (for the option 'drop new audit data' in FAU_STG_EXT.1.3).

2) The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option 'overwrite previous audit records' in FAU_STG_EXT.1.3)

3) The TOE behaves as specified (for the option 'other action' in FAU_STG_EXT.1.3).

c) Test 3: If the TOE complies with FAU_STG_EXT.2/LocSpace the evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2/LocSpace are correct when performing the tests for FAU_STG_EXT.1.3

d) Test 4: For distributed TOEs, Test 1 defined above should be applicable to all TOE components that forward audit data to an external audit server. For the local storage according to FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3 the Test 2 specified above shall be applied to all TOE components that store audit data locally. For all TOE components that store audit data locally and comply with FAU_STG_EXT.2/LocSpace Test 3 specified above shall be applied. The evaluator shall verify that the transfer of audit data to an external audit server is implemented.

2.2 Cryptographic Support (FCS)

2.2.1 FCS_CKM.1 Cryptographic Key Generation

2.2.1.1 TSS

41 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.

2.2.1.2 Guidance Documentation

42 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 cryptographic protocols defined in the Security Target.

2.2.1.3 Tests

43 Note: 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. Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

44 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*.

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

a) Random Primes:

- Provable primes
- Probable primes

b) 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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

49 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.

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

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

51 and two ways to generate the cryptographic group generator g:

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

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

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

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

54 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.

55 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

56 for each FFC parameter set and key pair.

Diffie-Hellman Group 14

57 Testing for FFC Schemes using Diffie-Hellman group 14 is done as part of testing in CKM.2.1.

2.2.2 FCS_CKM.2 Cryptographic Key Establishment

2.2.2.1 TSS

58 The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme (including whether the TOE acts as a sender, a recipient, or both). If Diffie-Hellman group 14 is selected from FCS_CKM.2.1, the TSS shall describe how the implementation meets RFC 3526 Section 3.

2.2.2.2 Guidance Documentation

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

2.2.2.3 Tests

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

62 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.

63 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.

64 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.

65 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.

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

Validity Test

67 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.

68 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).

69 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.

SP800-56B Key Establishment Schemes

70 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:

a) 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.

71 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:

a) 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.

b) 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.

Diffie-Hellman Group 14

72 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_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses Diffie-Hellman group 14.

2.2.3 FCS_CKM.4 Cryptographic Key Destruction

2.2.3.1 TSS

73 The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for[2]). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

74 The evaluator shall check to ensure the TSS identifies how the TOE destroys keys stored as plaintext in non-volatile memory, and that the description includes identification and description of the interfaces that the TOE uses to destroy keys (e.g., file system APIs, key store APIs).

75 Note that where selections involve '*destruction of reference*' (for volatile memory) or '*invocation of an interface'* (for non-volatile memory) then the relevant interface definition is examined by the evaluator to ensure that the interface supports the selection(s) and description in the TSS. In the case of non-volatile memory the evaluator includes in their examination the relevant interface description for each media type on which plaintext keys are stored. The presence of OS-level and storage device-level swap and cache files is not examined in the current version of the Evaluation Activity.

76 Where the TSS identifies keys that are stored in a non-plaintext form, the evaluator shall check that the TSS identifies the encryption method and the key-encrypting-key used, and that the key-encrypting-key is either itself stored

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

in an encrypted form or that it is destroyed by a method included under FCS_CKM.4.

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

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

2.2.3.2 Guidance Documentation

79 A TOE may be subject to situations that could 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 supporting information used). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

80 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 result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command[3] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

2.2.4 FCS_COP.1/DataEncryption Cryptographic Operation (AES Data Encryption/Decryption)

2.2.4.1 Tests

AES-CBC Known Answer Tests

81 There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

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

82 **KAT-1.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

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

84 **KAT-2.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

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

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

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

AES-GCM Test

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

128 bit and 256 bit keys

a) **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.

a) **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.

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

96 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.

97 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.

98 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.

AES-CTR Known Answer Tests

99 There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, IV, and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator 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.

100 KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all zeros key, and the other five shall be encrypted with a 256-bit all zeros key. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input.

101 KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from encryption of an

all zeros plaintext using the given key value and an IV of all zeros. Five of the key values shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.

102 KAT-3 To test the encrypt functionality, the evaluator shall supply the two sets of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second shall have 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit pairs. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros for i in [1, N]. The ciphertext value in each pair shall be the value that results in an all zeros plaintext when decrypted with its corresponding key.

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

AES-CTR Multi-Block Message Test

104 The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

AES-CTR Monte-Carlo Test

105 The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

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

106 The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

107 There is no need to test the decryption engine.

2.2.5 FCS_COP.1/SigGen Cryptographic Operation (Signature Generation and Verification

2.2.5.1 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

110 The evaluator generates or obtains 10 messages for each modulus size/SHA combination supported by the TOE. The TOE generates and returns the corresponding signatures.

111 The evaluator shall verify the correctness of the TOE's signature using a trusted reference implementation of the signature verification algorithm and the associated public keys to verify the signatures.

Signature Verification Test

112 For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (*d*, *e*). Each private key *d* is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, *e*, messages, or signatures are altered so that signature verification

should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key *e* values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

113 The evaluator verifies that the TOE confirms correct signatures on the original messages and detects the errors introduced in the altered messages.

2.2.6 FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)

2.2.6.1 TSS

114 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.

2.2.6.2 Guidance Documentation

115 The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

2.2.6.3 Tests

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

117 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 PP.

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

2.2.7 FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithm)

2.2.7.1 TSS

123 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.

2.2.7.2 Tests

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

resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and message data using a known good implementation.

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

125 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [NDcPP].

2.2.8.1 TSS

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

2.2.8.2 Guidance Documentation

127 The evaluator shall confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

2.2.8.3 Tests

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

129 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).

130 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.

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

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

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

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

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

2.3 Identification and Authentication (FIA)

2.3.1 FIA_AFL.1 Authentication Failure Management

2.3.1.1 TSS

132 The evaluator shall examine the TSS to determine that it contains a description, for each supported method for remote administrative actions, of how successive unsuccessful authentication attempts are detected and tracked. The TSS shall also describe the method by which the remote administrator is prevented from successfully logging on to the TOE, and the actions necessary to restore this ability.

133 The evaluator shall examine the TSS to confirm that the TOE ensures that authentication failures by remote administrators cannot lead to a situation where no administrator access is available, either permanently or temporarily (e.g. by providing local logon which is not subject to blocking).

2.3.1.2 Guidance Documentation

134 The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the remote administrator to once again successfully log on is described for each “action” specified (if that option is chosen). If different actions or mechanisms are implemented depending on the secure protocol employed (e.g., TLS vs. SSH), all must be described.

135 The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

2.3.1.3 Tests

136 The evaluator shall perform the following tests for each method by which remote administrators access the TOE (e.g. any passwords entered as part of establishing the connection protocol or the remote administrator application):

a) Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.

b) Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows.

If the administrator action selection in FIA_AFL.1.2 is included in the ST then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote administrator's access results in successful access (when using valid credentials for that administrator).

If the time period selection in FIA_AFL.1.2 is included in the ST then the evaluator shall wait for just less than the time period configured in Test 1 and show that an authorisation attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorisation attempt using valid credentials results in successful access.

2.3.2 FIA_PMG_EXT.1 Password Management

2.3.2.1 Guidance Documentation

137 The evaluator shall examine the guidance documentation to determine that it:

a) identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and

b) provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

2.3.2.2 Tests

138 The evaluator shall perform the following tests.

a) Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

2.3.3 FIA_UIA_EXT.1 User Identification and Authentication

2.3.3.1 TSS

139 The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a "successful logon".

140 The evaluator shall examine the TSS to determine that it describes which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration.

141 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 and FIA_UAU_EXT.2, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no unauthorized access to any TOE component can occur.

142 For distributed TOEs, the evaluator shall examine the TSS to determine that it describes for each TOE component which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration. For each TOE component that does not support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2 the TSS shall describe any unauthenticated services/services that are supported by the component.

2.3.3.2 Guidance Documentation

143 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 the 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.

2.3.3.3 Tests

144 The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. 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.

b) Test 2: 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.

c) Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.

d) Test 4: For distributed TOEs where not all TOE components support the authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the evaluator shall test that the components authenticate Security Administrators as described in the TSS.

2.3.4 FIA_UAU_EXT.2 Password-based Authentication Mechanism

145 Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

2.3.5 FIA_UAU.7 Protected Authentication Feedback

2.3.5.1 Tests

146 The evaluator shall perform the following test for each method of local login allowed:

a) Test 1: The evaluator shall locally authenticate to the TOE. While making this attempt, the evaluator shall verify that at most obscured feedback is provided while entering the authentication information.

2.4 Security management (FMT)

2.4.1 General requirements for distributed TOEs

2.4.1.1 TSS

147 For distributed TOEs it is required to verify the TSS to ensure that it describes how every function related to security management is realized for every TOE component and shared between different TOE components. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

2.4.1.2 Guidance Documentation

148 For distributed TOEs it is required to verify the Guidance Documentation to describe management of each TOE component. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

2.4.1.3 Tests

149 Tests defined to verify the correct implementation of security management functions shall be performed for every TOE component. For security management functions that are implemented centrally, sampling should be applied when defining the evaluator's tests (ensuring that all components are covered by the sample).

2.4.2 FMT_MOF.1/ManualUpdate

2.4.2.1 TSS

150 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

2.4.2.2 Guidance Documentation

151 The evaluator shall examine the guidance documentation to determine that any necessary steps to perform manual update are described. The guidance documentation shall also provide warnings regarding functions that may cease to operate during the update (if applicable).

152 For distributed TOEs the guidance documentation shall describe all steps how to update all TOE components. This shall contain description of the order in which components need to be updated if the order is relevant to the update process. The guidance documentation shall also provide warnings regarding functions of TOE components and the overall TOE that may cease to operate during the update (if applicable).

2.4.2.3 Tests

153 The evaluator shall try to perform the update using a legitimate update image 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). The attempt to update the TOE shall fail.

154 The evaluator shall try to perform the update with prior authentication as security administrator using a legitimate update image. This attempt should be successful. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

2.4.3 FMT_MTD.1/CoreData Management of TSF Data

2.4.3.1 TSS

155 The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are accessible through an interface prior to administrator log-in are identified. For each of these functions, the evaluator shall also confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

156 If the TOE supports handling of X.509v3 certificates and implements a trust store, the evaluator shall examine the TSS to determine that it contains sufficient information to describe how the ability to manage the TOE's trust store is restricted.

2.4.3.2 Guidance Documentation

157 The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

158 If the TOE supports handling of X.509v3 certificates and provides a trust store, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to configure and maintain the trust store in a secure way. If the TOE supports loading of CA certificates, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to securely load CA certificates into the trust store. The evaluator shall also review the guidance documentation to determine that it explains how to designate a CA certificate a trust anchor.

2.4.4 FMT_SMF.1 Specification of Management Functions

159 The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_SSL_EXT.1, FTA_SSL.3, FTA_TAB.1, FMT_MOF.1/ManualUpdate, FMT_MOF.1/AutoUpdate (if included in the ST), FIA_AFL.1, FIA_X509_EXT.2.2 (if included in the ST), FPT_TUD_EXT.1.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1/Services, and

FMT_MOF.1/Functions (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

2.4.4.1 TSS (containing also requirements on Guidance Documentation and Tests)

160 The evaluator shall examine the TSS, Guidance Documentation and the TOE as observed during all other testing and shall confirm that the management functions specified in FMT_SMF.1 are provided by the TOE. The evaluator shall confirm that the TSS details which security management functions are available through which interface(s) (local administration interface, remote administration interface).

161 For distributed TOEs with the option 'ability to configure the interaction between TOE components' the evaluator shall examine that the ways to configure the interaction between TOE components is detailed in the TSS and Guidance Documentation. The evaluator shall check that the TOE behaviour observed during testing of the configured SFRs is as described in the TSS and Guidance Documentation.

2.4.4.2 Guidance Documentation

162 See section 2.4.4.1.

2.4.4.3 Tests

163 The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

2.4.5 FMT_SMR.2 Restrictions on security roles

2.4.5.1 Guidance Documentation

164 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.

2.4.5.2 Tests

165 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.

2.5 Protection of the TSF (FPT)

2.5.1 FPT_SKP_EXT.1 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys)

2.5.1.1 TSS

166 The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

2.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

2.5.2.1 TSS

167 The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

2.5.3 FPT_TST_EXT.1 TSF testing

2.5.3.1 TSS

168 The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

169 For distributed TOEs the evaluator shall examine the TSS to ensure that it details which TOE component performs which self-tests and when these self-tests are run.

2.5.3.2 Guidance Documentation

170 The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

171 For distributed TOEs the evaluator shall ensure that the guidance documentation describes how to determine from an error message returned which TOE component has failed the self-test.

2.5.3.3 Tests

172 It is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE

b) Verification of the correct operation of the cryptographic functions necessary to fulfil any of the SFRs.

173 Although formal compliance is not mandated, the self-tests performed should aim for a level of confidence comparable to:

a) [FIPS 140-2], chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

b) [FIPS 140-2], chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions. Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

174 The evaluator shall either verify that the self-tests described above are carried out during initial start-up or that the developer has justified any deviation from this.

175 For distributed TOEs the evaluator shall perform testing of self-tests on all TOE components according to the description in the TSS about which self-test are performed by which component.

2.5.4 FPT_TUD_EXT.1 Trusted Update

2.5.4.1 TSS

176 The evaluator shall verify that the TSS describe how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the TSS needs to describe how and when the inactive version becomes active. The evaluator shall verify this description.

177 The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system firmware and software (for simplicity the term 'software' will be used in the following although the requirements apply to firmware and software). The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash

of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

178 If the options ‘support automatic checking for updates’ or ‘support automatic updates’ are chosen from the selection in FPT_TUD_EXT.1.2, the evaluator shall verify that the TSS explains what actions are involved in automatic checking or automatic updating by the TOE, respectively.

179 For distributed TOEs, the evaluator shall examine the TSS to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component. Alternatively, this description can be provided in the guidance documentation. In that case the evaluator should examine the guidance documentation instead.

180 If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or guidance documentation) describes how the certificates are installed/updated/selected, if necessary.

181 If a published hash is used to protect the trusted update mechanism, then the evaluator shall verify that the trusted update mechanism does involve an active authorization step of the Security Administrator, and that download of the published hash value, hash comparison and update is not a fully automated process involving no active authorization by the Security Administrator. In particular, authentication as Security Administration according to FMT_MOF.1/ManualUpdate needs to be part of the update process when using published hashes.

2.5.4.2 Guidance Documentation

182 The evaluator shall verify that the guidance documentation describes how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the guidance documentation needs to describe how to query the loaded but inactive version.

183 The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

184 If a published hash is used to protect the trusted update mechanism, the evaluator shall verify that the guidance documentation describes how the Security Administrator can obtain authentic published hash values for the updates.

185 For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT_TUD_EXT.1, how all TOE components are updated, and the error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. . The guidance documentation only has to describe the procedures relevant for the user; it does not need to give information about the internal communication that takes place when applying updates.

186 If this was information was not provided in the TSS: For distributed TOEs, the evaluator shall examine the Guidance Documentation to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component.

187 If this was information was not provided in the TSS: If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the Guidance Documentation contains a description of how the certificates are contained on the device. The evaluator also ensures that the Guidance Documentation describes how the certificates are installed/updated/selected, if necessary.

2.5.4.3 Tests

188 The evaluator shall perform the following tests:

a) Test 1: The evaluator performs the version verification activity to determine the current version of the product. If a trusted update can be installed on the TOE with a delayed activation, the evaluator shall also query the most recently installed version (for this test the TOE shall be in a state where these two versions match). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

b) Test 2 (if digital signatures are used): The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in

this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

1) A modified version (e.g. using a hex editor) of a legitimately signed update

2) An image that has not been signed

3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)

4) If the TOE allows a delayed activation of updates the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

c) Test 3 (if published hash is verified on the TOE): If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

1) The evaluator obtains or produces an illegitimate update such that the hash of the update does not match the published hash. The evaluator provides the published hash value to the TOE and calculates the hash of the update either on the TOE itself (if that functionality is provided by the TOE), or else outside the TOE. The evaluator confirms that the hash values are different, and attempts to install the update on the TOE, verifying that this fails because of the difference in hash values (and that the failure is logged). Depending on the implementation of the TOE, the TOE might not allow the user to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash

comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

2) The evaluator uses a legitimate update and tries to perform verification of the hash value without storing the published hash value on the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to the TOE, e.g. if the hash value needs to be handed over to the TOE as a parameter in a command line message and the syntax check of the command prevents the execution of the command without providing a hash value. In that case the mechanism that prevents the execution of this check shall be tested accordingly, e.g. that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

3) If the TOE allows delayed activation of updates, the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs. Depending on the point in time when the attempted update is rejected, the most recently installed version might or might not be updated. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

189 If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 3 shall be skipped.

190 The evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all methods supported (manual updates, automatic checking for updates, automatic updates).

191 For distributed TOEs the evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all TOE components.

2.5.5 FPT_STM_EXT.1 Reliable Time Stamps

2.5.5.1 TSS

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

2.5.5.2 Guidance Documentation

193 The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the guidance documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

2.5.5.3 Tests

194 The evaluator shall perform the following tests:

a) Test 1: If the TOE supports direct setting of the time by the Security Administrator then the evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

b) Test 2: If the TOE supports the use of an NTP server; the evaluator shall use the guidance documentation to configure the NTP client on the TOE, and set up a communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

195 If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

2.6 TOE Access (FTA)

2.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

2.6.1.1 Guidance Documentation

196 The evaluator shall confirm that the guidance documentation states whether local administrative session locking or termination is supported and instructions for configuring the inactivity time period.

2.6.1.2 Tests

197 The evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that re-authentication is needed when trying to unlock the session.

2.6.2 FTA_SSL.3 TSF-initiated Termination

2.6.2.1 Guidance Documentation

198 The evaluator shall confirm that the guidance documentation states whether local administrative session locking or termination is supported and instructions for configuring the inactivity time period.

2.6.2.2 Tests

199 For each method of remote administration, the evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

2.6.3 FTA_SSL.4 User-initiated Termination

2.6.3.1 Guidance Documentation

200 The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

2.6.3.2 Tests

201 For each method of remote administration, the evaluator shall perform the following tests:

a) Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

b) Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

2.6.4 FTA_TAB.1 Default TOE Access Banners

2.6.4.1 TSS

202 The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access, and might be configured during initial configuration (e.g. via configuration file).

2.6.4.2 Guidance Documentation

203 The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

2.6.4.3 Tests

204 The evaluator shall also perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

2.7 Trusted path/channels (FTP)

2.7.1 FTP_ITC.1 Inter-TSF trusted channel

2.7.1.1 TSS

2.7.1.2

The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

2.7.1.3 Guidance Documentation

205 The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT

entity, and that it contains recovery instructions should a connection be unintentionally broken.

2.7.1.4 Tests

206 The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

207 The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) 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.

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

d) Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the MAC layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

208 Further assurance activities are associated with the specific protocols.

209 For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

210 The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public- facing document or report.

2.7.2 FTP_TRP.1/Admin Trusted Path

2.7.2.1 TSS

211 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.

2.7.2.2 Guidance Documentation

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

2.7.2.3 Tests

213 The evaluator shall perform the following tests:

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

b) Test 2: The evaluator shall ensure, for each communication channel, the channel data is not sent in plaintext.

214 Further assurance activities are associated with the specific protocols.

215 For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

3 Evaluation Activities for Optional Requirements

3.1 Security Audit (FAU)

3.1.1 FAU_STG.1 Protected audit trail storage

3.1.1.1 TSS

216 The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally and how these records are protected against unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.

217 For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how local storage is implemented among the different TOE components (e.g. every TOE component does its own local storage or the data is sent to another TOE component for central local storage of all audit events).

3.1.1.2 Guidance Documentation

218 The evaluator shall examine the guidance documentation to determine that it describes any configuration required for protection of the locally stored audit data against unauthorized modification or deletion.

3.1.1.3 Tests

219 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall access the audit trail without authentication as Security Administrator (either by authentication as a non-administrative user, if supported, or without authentication at all) and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to access the audit trail can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

b) Test 2: The evaluator shall access the audit trail as an authorized administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.

220 For distributed TOEs the evaluator shall perform test 1 and test 2 for each component that is defined by the TSS to be covered by this SFR.

3.1.2 FAU_STG_EXT.2/LocSpace Counting lost audit data

221 This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

3.1.2.1 TSS

222 The evaluator shall examine the TSS to ensure that it details the possible options the TOE supports for information about the number of audit records that have been dropped, overwritten, etc. if the local storage for audit data is full.

223 For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies. Since this SFR is optional, it might only apply to some TOE components but not all. This might lead to the situation where all TOE components store their audit information themselves but FAU_STG_EXT.2/LocSpace is supported only by one of the components.

3.1.2.2 Guidance Documentation

224 The evaluator shall also ensure that the guidance documentation describes all possible configuration options and the meaning of the result returned by the TOE for each possible configuration. The description of possible configuration options and explanation of the result shall correspond to those described in the TSS.

225 The evaluator shall verify that the guidance documentation contains a warning for the administrator about the loss of audit data when clearing the local storage for audit records.

3.1.2.3 Tests

226 The evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2/LocSpace are correct when performing the tests for FAU_STG_EXT.1.3.

227 For distributed TOEs the evaluator shall verify the correct implementation of counting of lost audit data for all TOE components that are supporting this feature according to the description in the TSS.

3.1.3 FAU_STG.3/LocSpace Action in case of possible audit data loss

228 This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

3.1.3.1 TSS

229 The evaluator shall examine the TSS to ensure that it details how the user is warned before the local storage for audit data is full.

230 For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how each TOE component realises this SFR. Since this SFR is optional, it might only apply to some TOE components but not all. This might lead to the situation where all TOE components store their audit information themselves but FAU_STG.3/LocSpace is supported only by one of the components. In particular, the evaluator has to verify, that the TSS describes for every component supporting this functionality, whether the warning is generated by the component itself or through another component and name the corresponding component in the latter case. The evaluator has to verify that the TSS makes clear any situations in which audit records might be 'invisibly lost'.

3.1.3.2 Guidance Documentation

231 The evaluator shall also ensure that the guidance documentation describes how the user is warned before the local storage for audit data is full and how this warning is displayed or stored (since there is no guarantee that an administrator session is running at the time the warning is issued, it is probably stored in the log files). The description in the guidance documentation shall correspond to the description in the TSS.

3.1.3.3 Tests

232 The evaluator shall verify that a warning is issued by the TOE before the local storage space for audit data is full.

233 For distributed TOEs the evaluator shall verify the correct implementation of display warning for local storage space for all TOE components that are supporting this feature according to the description in the TSS. The evaluator shall verify that each component that supports this feature according to the description in the TSS is capable of generating a warning itself or through another component.

3.2 Identification and Authentication (FIA)

3.2.1 FIA_X509_EXT.1/ITT X.509 Certificate Validation

3.2.1.1 TSS

234 The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). If selected, the TSS shall describe how certificate revocation checking is performed. It is not sufficient to verify the status of a X.509 certificate only when it's loaded onto the device.

3.2.1.2 Tests

235 The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device. The evaluator shall perform the following tests for FIA_X509_EXT.1.1/ITT. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols.:

a) Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOEs trust store, or by setting up the trust store in a way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store).

   Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE's trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

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

c) 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. No testing is required if no revocation method is selected. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore the revoked certificate(s) used for testing shall not be a trust anchor.

d) 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.

e) 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.)

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

g) 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.)

236 The evaluator shall perform the following tests for FIA_X509_EXT.1.2/ITT. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/ITT. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

237 The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

238 For each of the following tests the evaluator shall create a chain of at least two certificates: a self-signed root CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

a) Test 1: The evaluator shall ensure that one CA in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

b) Test 2: The evaluator shall ensure that at least one of the CA certificates in the chain has a basicConstraints extension in which the CA flag is set to FALSE. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of

the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate with the CA flag set to FALSE to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

3.3 Protection of the TSF (FPT)

3.3.1 FPT_ITT.1 Basic internal TSF data transfer protection

239 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.

3.3.1.1 TSS

240 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 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.

3.3.1.2 Guidance Documentation

241 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.

3.3.1.3 Tests

242 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall ensure that communications using each 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.

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

c) Test 3: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route between distributed components.

The evaluator shall ensure that, for each different pair of non-equivalent component types, the connection is physically interrupted for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration that is shorter than the

application layer timeout but is of sufficient length to interrupt the MAC layer.

The evaluator shall ensure that when physical connectivity is restored, either communications are appropriately protected, or the secure channel is terminated and the registration process (as described in the FTP_TRP.1/Join) re-initiated, with the TOE generating adequate warnings to alert the administrator.

In the case that the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the components.

The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

243 Further assurance activities are associated with the specific protocols.

3.4 Trusted path/channels (FTP)

3.4.1 FTP_TRP.1/Join Trusted Path

3.4.1.1 TSS

244 The evaluator shall examine the TSS to determine that the methods of joining components to the TOE are identified, along with how those communications are protected, including identification of whether the environment is required to provide confidentiality of the communications or whether the registration data exchanged does not require confidentiality. If the TSS asserts that registration data does not require confidentiality protection then the evaluator shall examine the justification provided to confirm that.

245 The evaluator shall also check that all protocols listed in the TSS in support of this process are included in the SFRs in the ST, and that if the ST uses FTP_TRP.1/Join for the registration channel then this channel cannot be reused as the normal inter-component communication channel (the latter channel must meet FTP_ITC.1 or FPT_ITT.1).

246 The evaluator shall examine the TSS to confirm that sufficient information is provided to determine the TOE actions in the case that the initial component joining attempt fails

3.4.1.2 Guidance Documentation

247 The evaluator shall examine the guidance documentation to confirm that it contains instructions for establishing and using the enablement and registration channel. The evaluator shall confirm that the guidance documentation makes clear which component initiates the communication. The evaluator shall confirm that the guidance documentation contains recovery instructions should a connection be unintentionally broken during the registration process.

248 In the case of a distributed TOE that relies on the operational environment to provide security for some aspects of the registration channel security then there are particular requirements on the Preparative Procedures as listed below. (Reliance on the operational environment in this way is indicated in an ST by a reference to operational guidance in the assignment in FTP_TRP.1.3/Join.) In this case the evaluator shall examine the Preparative Procedures to confirm that they:

a) clearly state the strength of the authentication and encryption provided by the registration channel itself and the specific requirements on the environment used for joining components to the TOE (e.g. where the environment is relied upon to prevent interception of sensitive messages, IP spoofing attempts, man-in-the-middle attacks, or race conditions)

b) identify what confidential values are transmitted over the enablement channel (e.g. any keys, their lengths, and their purposes), use of any non-confidential keys (e.g. where a developer uses the same key for more than one device or across all devices of a type or family), and use of any unauthenticated identification data (e.g. IP addresses, self-signed certificates)

c) highlight any situation in which a secret value/key may be transmitted over a channel that uses a key of lower comparable strength than the transmitted value/key. Comparable strength is defined as the amount of work required to compromise the algorithm or key and is typically expressed as 'bits' of security. The ST author and evaluator should consult NIST 800-57 Table 2 for further guidance on comparable algorithm strength.

3.4.1.3 Tests

249 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall ensure that the communications path for joining components to the TSF is tested for each distinct (non-equivalent) component type[4], setting up the connections as described in the guidance documentation and ensuring that communication is successful. In particular the evaluator shall confirm that requirements on environment protection for the registration process are consistent with observations made on the test configuration (for example, a requirement to isolate the components from the Internet during registration might be inconsistent with the need for a component to contact a license server). If no requirements on the registration environment are identified as necessary to protect confidentiality, then the evaluator shall confirm that the key used for registration can be

[4] The intention here is to cover all different software sections involved. For example, a single software image may be installed on different TOE components, but with different sections of the image executed according to the hardware platform or communications stack. In such as case tests should be carried out for each different software section.

configured (following the instructions in the guidance documentation) to be at least the same length as the key used for the internal TSF channel that is being enabled. The evaluator shall confirm that the key used for the channel is unique to the pair of components (this is done by identifying the relevant key during the registration test: it is not necessary to examine the key value).

b) Test 2: The evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be enabled by an administrator for all the TOE components identified in the guidance documentation as capable of initiation.

c) Test 3: The evaluator shall ensure that if the guidance documentation states that the channel data is encrypted then the data observed on the channel is not plaintext.

250 Further assurance activities are associated with the specific protocols.

3.5 Communication (FCO)

3.5.1 FCO_CPC_EXT.1 Component Registration Channel Definition

251 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. In carrying out these activities the evaluator shall determine answers to the following questions based on a combination of documentation analysis and testing (possibly also using input from carrying out the Evaluation Activities for the relevant registration channel, such as FTP_TRP.1/Join), and shall report the answers.

a) What stops[5] a component from successfully communicating with TOE components (in a way that enables it to participate as part of the TOE) before it has properly authenticated and joined the TOE?

b) What is the enablement step? (Describe what interface it uses, with a reference to the relevant section and step in the operational guidance).

[5] The intent of the phrasing "what stops…" as opposed to "what secures…" is for the evaluator to pursue the answer to its lowest level of dependency, i.e. a level at which the security can clearly be seen to depend on things that are under appropriate control. For example, a channel may be protected by a public key that is provided to the relying party in a self-signed certificate. This enables cryptographic mechanisms to be applied to provide authentication (and therefore invites an answer that "the check on the public key certificate secures…"), but does not ultimately stop an attacker from apparently authenticating because the attacker can produce their own self-signed certificate. The question "what stops an unauthorised component from successfully communicating…" focuses attention on what an attacker needs to do, and therefore pushes the answer down to the level of whether a self-signed certificate could be produced by an attacker. Similarly a well-known key, or a key that is common to a type of device rather than an individual device, may be used in a confidentiality mechanism but does not provide confidentiality because an attacker can find the well-known key or obtain his own instance of a device containing the non-unique key.

1) What stops anybody other than a Security Administrator from carrying out this step?

2) How does the Security Administrator know that they are enabling the intended component to join? (Identification of the joiner might be part of the enablement action itself or might be part of secure channel establishment, but it must prevent unintended joining of components)

c) What stops a component successfully joining if the Administrator has not carried out the enablement step; or, equivalently, how does the TOE ensure that an action by an authentic Administrator is required before a component can successfully join?

d) What stops a component from carrying out the registration process over a different, insecure channel?

e) If the FTP_TRP.1/Join channel type is selected in FCO_CPC_EXT.1.2 then how do the registration process and its secure channel ensure that the data is protected from disclosure and provides detection of modification?

f) Where the registration channel does not rely on protection of the registration environment, does the registration channel provide a sufficient level of protection (especially with regard to confidentiality) for the data that passes over it?

g) Where the registration channel is subsequently used for normal internal communication between TOE components (i.e. after the joiner has completed registration), do any of the authentication or encryption features of the registration channel result in use of a channel that has weaker protection than the normal FPT_ITT.1 requirements for such a channel?

h) What is the disablement step? (Describe what interface it uses, with a reference to the relevant section and step in the operational guidance).

i) What stops a component successfully communicating with other TOE components if the Administrator has carried out the disablement step?

3.5.1.1 TSS

252 (Note: paragraph 251 lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

253 The evaluator shall examine the TSS to confirm that it:

a) Describes the method by which a Security Administrator enables and disables communications between pairs of TOE components.

b) Describes the relevant details according to the type of channel in the main selection made in FCO_CPC_EXT.1.2:

- First type: the TSS identifies the relevant SFR iteration that specifies the channel used
- Second type: the TSS (with support from the operational guidance if selected in FTP_TRP.1.3/Join) describes details of the channel and the mechanisms that it uses (and describes how the process ensures that the key is unique to the pair of components) – see also the Evaluation Activities for FTP_TRP.1/Join.

254 The evaluator shall confirm that if any aspects of the registration channel are identified as not meeting FTP_ITC.1 or FPT_ITT.1, then the ST has also selected the FTP_TRP.1/Join option in the main selection in FCO_CPC_EXT.1.2.

3.5.1.2 Guidance Documentation

255 (Note: paragraph 251 lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

256 The evaluator shall examine the guidance documentation to confirm that it contains instructions for enabling and disabling communications with any individual component of a distributed TOE. The evaluator shall confirm that the method of disabling is such that all other components can be prevented from communicating with the component that is being removed from the TOE (preventing the remaining components from either attempting to initiate communications to the disabled component, or from responding to communications from the disabled component).

257 The evaluator shall examine the guidance documentation to confirm that it includes recovery instructions should a connection be unintentionally broken during the registration process.

258 If the TOE uses a registration channel for registering components to the TOE (i.e. where the ST author uses the FTP_ITC.1/FPT_ITT.1 or FTP_TRP.1/Join channel types in the main selection for FCO_CPC_EXT.1.2) then the evaluator shall examine the Preparative Procedures to confirm that they:

a) describe the security characteristics of the registration channel (e.g. the protocol, keys and authentication data on which it is based) and shall highlight any aspects which do not meet the requirements for a steady-state inter-component channel (as in FTP_ITC.1 or FPT_ITT.1)

b) identify any dependencies between the configuration of the registration channel and the security of the subsequent inter-component communications (e.g. where AES-256 inter-component communications depend on transmitting 256 bit keys between

components and therefore rely on the registration channel being configured to use an equivalent key length)

c) identify any aspects of the channel can be modified by the operational environment in order to improve the channel security, and shall describe how this modification can be achieved (e.g. generating a new key pair, or replacing a default public key certificate).

259 As background for the examination of the registration channel description, it is noted that the requirements above are intended to ensure that administrators can make an accurate judgement of any risks that arise from the default registration process. Examples would be the use of self-signed certificates (i.e. certificates that are not chained to an external or local Certification Authority), manufacturer-issued certificates (where control over aspects such as revocation, or which devices are issued with recognised certificates, is outside the control of the operational environment), use of generic/non-unique keys (e.g. where the same key is present on more than one instance of a device), or well-known keys (i.e. where the confidentiality of the keys is not intended to be strongly protected – note that this need not mean there is a positive action or intention to publicise the keys).

260 In the case of a distributed TOE for which the ST author uses the FTP_TRP.1/Join channel type in the main selection for FCO_CPC_EXT.1.2 and the TOE relies on the operational environment to provide security for some aspects of the registration channel security then there are additional requirements on the Preparative Procedures as described in section 3.4.1.2.

3.5.1.3 Tests

261 (Note: paragraph 251 lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

262 The evaluator shall carry out the following tests:

a) Test 1.1: the evaluator shall confirm that an IT entity that is not currently a member of the distributed TOE cannot communicate with any component of the TOE until the non-member entity is enabled by a Security Administrator for each of the non-equivalent TOE components[6] that it is required to communicate with (non-equivalent TOE components are as defined in the minimum configuration for the distributed TOE)

[6] An 'equivalent TOE component' is a type of distributed TOE component that exhibits the same security characteristics, behaviour and role in the TSF as some other TOE component. In principle a distributed TOE could operate with only one instance of each equivalent TOE component, although the minimum configuration of the distributed TOE may include more than one instance (see discussion of the minimum configuration of a distributed TOE, in section B.4). In practice a deployment of the TOE may include more than one instance of some equivalent TOE components for practical reasons, such as performance or the need to have separate instances for separate subnets or VLANs.

b) Test 1.2: the evaluator shall confirm that after enablement, an IT entity can communicate only with the components that it has been enabled for. This includes testing that the enabled communication is successful for the enabled component pair, and that communication remains unsuccessful with any other component for which communication has not been explicitly enabled

Some TOEs may set up the registration channel before the enablement step is carried out, but in such a case the channel must not allow communications until after the enablement step has been completed.

263 The evaluator shall repeat Tests 1.1 and 1.2 for each different type of enablement process that can be used in the TOE.

c) Test 2: The evaluator shall separately disable each TOE component in turn and ensure that the other TOE components cannot then communicate with the disabled component, whether by attempting to initiate communications with the disabled component or by responding to communication attempts from the disabled component.

d) Test 3: The evaluator shall carry out the following tests according to those that apply to the values of the main (outer) selection made in the ST for FCO_CPC_EXT.1.2.

1) If the ST uses the first type of communication channel in the selection in FCO_CPC_EXT.1.2 then the evaluator tests the channel via the Evaluation Activities for FTP_ITC.1 or FPT_ITT.1 according to the second selection – the evaluator shall ensure that the test coverage for these SFRs includes their use in the registration process.

2) If the ST uses the second type of communication channel in the selection in FCO_CPC_EXT.1.2 then the evaluator tests the channel via the Evaluation Activities for FTP_TRP.1/Join.

3) If the ST uses the 'no channel' selection then no test is required.

e) Test 4: The evaluator shall perform one of the following tests, according to the TOE characteristics identified in its TSS and operational guidance:

1) If the registration channel is not subsequently used for inter-component communication, and in all cases where the second selection in FCO_CPC_EXT.1.2 is made (i.e. using FTP_TRP.1/Join) then the evaluator shall confirm that the registration channel can no longer be used after the registration process has completed, by attempting to use the channel to communicate with each of the endpoints after registration has completed

2) If the registration channel is subsequently used for inter-component communication then the evaluator shall confirm

that any aspects identified in the operational guidance as necessary to meet the requirements for a steady-state inter-component channel (as in FTP_ITC.1 or FPT_ITT.1) can indeed be carried out (e.g. there might be a requirement to replace the default key pair and/or public key certificate).

f) Test 5: For each aspect of the security of the registration channel that operational guidance states can be modified by the operational environment in order to improve the channel security (cf. AGD_PRE.1 refinement item 2 in (cf. the requirements on Preparative Procedures in 3.5.1.2), the evaluator shall confirm, by following the procedure described in the operational guidance, that this modification can be successfully carried out.

4 Evaluation Activities for Selection-Based Requirements

4.1 Security Audit (FAU)

4.1.1 FAU_GEN_EXT.1 Security Audit Data Generation for Distributed TOE Components

264 For distributed TOEs, the requirements on TSS, Guidance Documentation and Tests regarding FAU_GEN_EXT.1 are already covered by the corresponding requirements for FAU_GEN.1.

4.1.2 FAU_STG_EXT.3 Protected Local audit event storage for distributed TOEs & FAU_STG_EXT.4 Protected Remote audit event storage for Distributed TOEs

4.1.2.1 TSS

265 The evaluator examines the TSS to confirm that it describes which TOE components store their security audit events locally and which send their security audit events to other TOE components for local storage. For the latter, the target TOE component(s) which store security audit events for other TOE components shall be identified. For every sending TOE component the corresponding receiving TOE component(s) need to be identified. For every transfer of audit information between TOE components it shall be described how the data is secured during transfer according to FTP_ITC.1 or FPT_ITT.1.

266 For each TOE component which does not store audit events locally by itself, the evaluator confirms that the TSS describes how the audit information is buffered before sending to another TOE component for local storage.

4.1.2.2 Guidance Documentation

267 The evaluator shall examine the guidance documentation to ensure that it describes how the link between different TOE components is established if audit data is exchanged between TOE components for local storage. The guidance documentation shall describe all possible configuration options for local storage of audit data and provide all instructions how to perform the related configuration of the TOE components.

268 The evaluator shall also ensure that the guidance documentation describes for every TOE component which does not store audit information locally how audit information is buffered before transmission to other TOE components.

4.1.2.3 Tests

269 For at least one of each type of distributed TOE components (sensors, central nodes, etc.), the following tests shall be performed using distributed TOEs.

270 Test 1: For each type of TOE component, the evaluator shall perform a representative subset of auditable actions and ensure that these actions cause the generation of appropriately formed audit records. Generation of such records can be observed directly on the distributed TOE component (if there is appropriate interface), or indirectly after transmission to a central location.

271 Test 2: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to the external audit server (as specified in FTP_ITC.1), the evaluator shall configure a trusted channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

272 Test 3: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to another TOE component (as specified in FTP_ITT.1 or FTP_ITC.1, respectively), the evaluator shall configure a secure channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

273 While performing these tests, the evaluator shall verify that the TOE behaviour observed during testing is consistent with the descriptions provided in the TSS and the Guidance Documentation. Depending on the TOE configuration, there might be a large number of different possible configurations. In such cases, it is acceptable to perform subset testing, accompanied by an equivalency argument describing the evaluator's sampling methodology.

4.2 Cryptographic Support (FCS)

4.2.1 FCS_DTLSC_EXT.1 Extended: DTLS Client Protocol

4.2.1.1 TSS

FCS_DTLSC_EXT.1.1

274 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.

FCS_DTLSC_EXT.1.2

275 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.

276 Note that where a DTLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process. The TSS should describe the discovery process and highlight how the reference identifier is supplied to the "joining" component.

FCS_DTLSC_EXT.1.4

277 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.

4.2.1.2 Guidance Documentation

FCS_DTLSC_EXT.1.1

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

FCS_DTLSC_EXT.1.2

279 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 DTLS.

FCS_DTLSC_EXT.1.4

280 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.

4.2.1.3 Tests

281 For clarification: For DTLS communication packets might be received in a different order than sent due to the use of the UDP protocol. All tests requiring a specific order of test steps ("before", "after") are therefore referring to the sequence numbering of DTLS packets.

FCS_DTLSC_EXT.1.1

282 Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established

as part of the establishment of a higher-level application protocol, e.g., as part of a syslog 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).

283 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.

284 Test 3: The evaluator shall send a server certificate in the DTLS connection that does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

285 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.

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

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

b) If using DHE or ECDH, 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.

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.

d) If using DHE or ECDH, 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. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS then this test shall be omitted.

e) Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and

ACK message. This is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.

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

FCS_DTLSC_EXT.1.2

287 Note that where a DTLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process. The TSS should describe the discovery process and highlight how the reference identifier is supplied to the "joining" component.

288 The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a DTLS connection:

a) Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

b) 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.

c) Test 3: [conditional]: If the TOE does not mandate the presence of the SAN extension, 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. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

d) 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.

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

1) 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.

2) 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.com) and verify that the connection fails. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

f) 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.

g) 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.

FCS_DTLSC_EXT.1.3

289 The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

290 Test 1: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

291 Test 2: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

292 Test 3[conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator

shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

FCS_DTLSC_EXT.1.4

293 Test 1: If using ECDHE ciphers, the evaluator shall configure the server to perform an ECDHE key exchange in the DTLS 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.

4.2.2 FCS_DTLSC_EXT.2 Extended: DTLS Client Protocols with authentication

4.2.2.1 TSS

FCS_DTLSC_EXT.2.1

294 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.

FCS_DTLSC_EXT.2.2

295 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.

FCS_DTLSC_EXT.2.4

296 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.

FCS_DTLSC_EXT.2.5

297 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for DTLS mutual authentication.

FCS_DTLSC_EXT.2.6

298 The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS Server fails the MAC integrity check.

FCS_DTLSC_EXT.2.7

299 The evaluator shall verify that TSS describes how replay is detected and silently discarded for DTLS records that have previously been received and too old to fit in the sliding window.

4.2.2.2 Guidance Documentation

FCS_DTLSC_EXT.2.1

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

FCS_DTLSC_EXT.2.2

301 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 DTLS.

FCS_DTLSC_EXT.2.4

302 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.

FCS_DTLSC_EXT.2.5

303 The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for DTLS mutual authentication.

4.2.2.3 Tests

304 For clarification: For DTLS communication packets might be received in a different order than sent due to the use of the UDP protocol. All tests requiring a specific order of test steps ("before", "after") are therefore referring to the sequence numbering of DTLS packets.

FCS_DTLSC_EXT.2.1

305 Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level application protocol, e.g., as part of a syslog 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).

306 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.

307 Test 3: The evaluator shall send a server certificate in the DTLS connection that does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

308 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.

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

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

b) If using DHE or ECDHE, 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.

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.

d) If using DHE or ECDHE, 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. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS then this test shall be omitted.

e) Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and ACK message. This is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.

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

FCS_DTLSC_EXT.2.2

310 Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process. The TSS should

describe the discovery process and highlight how the reference identifier is supplied to the "joining" component.The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a DTLS connection:

a) Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

   Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

b) 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.

c) Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, 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. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

d) 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.

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

   1) 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.

   2) 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.com) and verify that the connection fails. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does

not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

f) 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.

g) 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.

FCS_DTLSC_EXT.2.3

311 The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

312 Test 1: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

313 Test 2: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

314 Test 3[conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

FCS_DTLSC_EXT.2.4

315 Test 1: If using DHE or ECDH, the evaluator shall configure the server to perform an ECDHE key exchange in the DTLS 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.

FCS_DTLSC_EXT.2.5

316 The purpose of these tests is to confirm that the TOE appropriately handles connection to peer servers that support and do not support mutual authentication.

317 Test 1: The evaluator shall establish a connection to a peer server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a DTLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.

318 Test 2: The evaluator shall establish a connection to a peer server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a DTLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) messages.

FCS_DTLSC_EXT.2.6

319 Test 1: The evaluator shall establish a DTLS connection. The evaluator will then modify at least one byte in a record message, and verify that the Client discards the record or terminates the DTLS session.

FCS_DTLSC_EXT.2.7

320 Test 1: The evaluator shall set up a DTLS connection with a DTLS Server. The evaluator shall then capture traffic sent from the DTLS Server to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the DTLS Server. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

4.2.3 FCS_DTLSS_EXT.1 Extended: DTLS Server Protocol

4.2.3.1 TSS

FCS_DTLSS_EXT.1.1

321 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.

FCS_DTLSS_EXT.1.3

322 The evaluator shall verify that the TSS describes how the DTLS Client IP address is validated prior to issuing a ServerHello message.

FCS_DTLSS_EXT.1.4

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

FCS_DTLSS_EXT.1.5

324 The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS Client fails the MAC integrity check.

FCS_DTLSS_EXT.1.6

325 The evaluator shall verify that TSS describes how replay is detected and silently discarded for DTLS records that have previously been received and too old to fit in the sliding window.

4.2.3.2 Guidance Documentation

FCS_DTLSS_EXT.1.1

326 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that DTLS 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).

FCS_DTLSS_EXT.1.4

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

4.2.3.3 Tests

328 For clarification: For DTLS communication packets might be received in a different order than sent due to the use of the UDP protocol. All tests requiring a specific order of test steps ("before", "after") are therefore referring to the sequence numbering of DTLS packets.

FCS_DTLSS_EXT.1.1

329 Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level application protocol, e.g., as part of a syslog 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).

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

331 Test 3: The evaluator shall use a client to send a key exchange message in the DTLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA 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.

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

a) withdrawn

b) withdrawn

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.

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.

e) (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)

The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

FCS_DTLSS_EXT.1.3

333 Modify at least one byte in the cookie from the Server's HelloVerifyRequest message, and verify that the Server rejects the Client's handshake message.

FCS_DTLSS_EXT.1.4

334 If using ECDHE ciphers, 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.

335 The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_DTLSS_EXT.1.4. For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.

336 Note that this testing can be accomplished in conjunction with the other testing activities.

FCS_DTLSS_EXT.1.5

337 The evaluator shall establish a connection using a client. The evaluator will then modify at least one byte in a record message, and verify that the Server discards the record or terminates the DTLS session.

FCS_DTLSS_EXT.1.6

338 The evaluator shall set up a DTLS connection. The evaluator shall then capture traffic sent from the DTLS Client to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the DTLS Client. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

4.2.4 FCS_DTLSS_EXT.2 Extended: DTLS Server Protocol with mutual authentication

4.2.4.1 TSS

FCS_DTLSS_EXT.2.1

339 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.

FCS_DTLSS_EXT.2.3

340 The evaluator shall verify that the TSS describes how the DTLS Client IP address is validated prior to issuing a ServerHello message.

FCS_DTLSS_EXT.2.4

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

FCS_DTLSS_EXT.2.5

342 The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS Client fails the MAC integrity check.

FCS_DTLSS_EXT.2.6

343 The evaluator shall verify that TSS describes how replay is detected and silently discarded for DTLS records that have previously been received and too old to fit in the sliding window.

FCS_DTLSS_EXT.2.7 and FCS_DTLSS_EXT.2.8

344 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for DTLS mutual authentication.

FCS_DTLSS_EXT.2.9

345 The evaluator shall verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

4.2.4.2 Guidance Documentation

FCS_DTLSS_EXT.2.1

346 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that DTLS 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).

FCS_DTLSS_EXT.2.4

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

FCS_DTLSS_EXT.2.7 and FCS_DTLSS_EXT.2.8

348 The evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for DTLS mutual authentication.

FCS_DTLSS_EXT.2.9

349 If the DN is not compared automatically to the Domain Name or IP address, username, or email address, then the evaluator shall ensure that the AGD guidance includes configuration of the expected DN or the directory server for the connection.

4.2.4.3 Tests

350 For clarification: For DTLS communication packets might be received in a different order than sent due to the use of the UDP protocol. All tests requiring

a specific order of test steps ("before", "after") are therefore referring to the sequence numbering of DTLS packets.

FCS_DTLSS_EXT.2.1

351 Test 1: The evaluator shall establish a DTLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level application protocol, e.g., as part of a syslog 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).

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

353 Test 3: The evaluator shall use a client to send a key exchange message in the DTLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA 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.

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

a) withdrawn

b) withdrawn

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.

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.

e) (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)

The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted

application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

FCS_DTLSS_EXT.2.3

355 Modify at least one byte in the cookie from the Server's HelloVerifyRequest message, and verify that the Server rejects the Client's handshake message.

FCS_DTLSS_EXT.2.4

356 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.

357 The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_DTLSS_EXT.2.4. For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.

358 Note that this testing can be accomplished in conjunction with the other testing activities

FCS_DTLSS_EXT.2.5

359 The evaluator shall establish a connection using a client. The evaluator will then modify at least one byte in a record message, and verify that the Server discards the record or terminates the DTLS session.

FCS_DTLSS_EXT.2.6

360 The evaluator shall set up a DTLS connection. The evaluator shall then capture traffic sent from the DTLS Client to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the DTLS Client. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

FCS_DTLSS_EXT.2.7 and FCS_DTLSS_EXT.2.8

361 Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

362 Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

363 Test 3: The evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied.

364 Test 4: The aim of this test is to check the response of the server when it receives a client identity certificate that is signed by an impostor CA (either Root CA or intermediate CA). To carry out this test the evaluator shall configure the client to send a client identity certificate with an issuer field that identifies a CA recognised by the TOE as a trusted CA, but where the key used for the signature on the client certificate does not in fact correspond to the CA certificate trusted by the TOE (meaning that the client certificate is invalid because its certification path does not in fact terminate in the claimed CA certificate). The evaluator shall verify that the attempted connection is denied.

365 Test 5: The evaluator shall perform the following modifications to the traffic:

a) Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.

b) Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

366 The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

367 Test 6: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

368 Test 7: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

369 Test 8[conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

FCS_DTLSS_EXT.2.9

370 The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

4.2.5 FCS_HTTPS_EXT.1 HTTPS Protocol

4.2.5.1 TSS

371 The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

4.2.5.2 Tests

372 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall attempt to establish each trusted path or channel that utilizes HTTPS, observe the traffic with a packet analyser, verify that the connection succeeds, and verify that the traffic is identified as TLS or HTTPS.

373 Other tests are performed in conjunction with the TLS evaluation activities.

374 If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

4.2.6 FCS_IPSEC_EXT.1 IPsec Protocol

4.2.6.1 TSS

FCS_IPSEC_EXT.1.1

375 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.

376 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.

FCS_IPSEC_EXT.1.3

377 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).

FCS_IPSEC_EXT.1.4

378 The evaluator shall examine the TSS to verify that the selected algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1/KeyedHash Cryptographic Operations (for keyed-hash message authentication).

FCS_IPSEC_EXT.1.5

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

380 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.

FCS_IPSEC_EXT.1.6

381 The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms chosen in the selection of the requirement are included in the TSS discussion.

FCS_IPSEC_EXT.1.7

382 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.

FCS_IPSEC_EXT.1.8

383 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.

FCS_IPSEC_EXT.1.9

384 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.

FCS_IPSEC_EXT.1.10

385 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.

386 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.

FCS_IPSEC_EXT.1.11

387 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.

FCS_IPSEC_EXT.1.12

388 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.

FCS_IPSEC_EXT.1.13

389 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/SigGen Cryptographic Operations (for cryptographic signature).

390 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.

FCS_IPSEC_EXT.1.14

391 The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN). If the TOE simultaneously supports the same identifier type in the CN and SAN, the TSS shall describe how the TOE prioritizes the comparisons (e.g. the result of comparison if CN matches but SAN does not). If the location (e.g. CN or SAN) of non-DN identifier types must explicitly be configured as part of the reference identifier, the TSS shall state this. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate, including what field(s) are compared and which fields take precedence in the comparison.

4.2.6.2 Guidance Documentation

FCS_IPSEC_EXT.1.1

392 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.

FCS_IPSEC_EXT.1.3

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

FCS_IPSEC_EXT.1.4

394 The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

FCS_IPSEC_EXT.1.5

395 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 how to configure the TOE to perform NAT traversal (if selected).

396 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.

FCS_IPSEC_EXT.1.6

397 The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement.

FCS_IPSEC_EXT.1.7

398 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. 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.

FCS_IPSEC_EXT.1.8

399 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. 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.

FCS_IPSEC_EXT.1.11

400 The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement.

FCS_IPSEC_EXT.1.13

401 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.

402 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.

403 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".

FCS_IPSEC_EXT.1.14

404 The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not, and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE does not guarantee unique identifiers, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

FCS_IPSEC_EXT.1.1

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

a) 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 behaviour: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

b) 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.

FCS_IPSEC_EXT.1.2

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

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

408 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.

FCS_IPSEC_EXT.1.3

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

a) Test 1: 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.

b) Test 2: If transport mode is selected, 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.

FCS_IPSEC_EXT.1.4

410 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.

FCS_IPSEC_EXT.1.5

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

a) Test 1: If IKEv1 is selected, 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.

b) Test 2: If NAT traversal is selected within the IKEv2 selection, 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.

FCS_IPSEC_EXT.1.6

412 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.

FCS_IPSEC_EXT.1.7

413 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."

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

a) Test 1: If 'number of bytes' is selected as the SA lifetime measure, 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.

b) Test 2: If 'length of time' is selected as the SA lifetime measure, 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 a new Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

FCS_IPSEC_EXT.1.8

415 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."

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

a) Test 1: If 'number of bytes' is selected as the SA lifetime measure, 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.

b) Test 2: If 'length of time' is selected as the SA lifetime measure, 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 a new Phase 2 SA is negotiated when or before 8 hours has lapsed. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

FCS_IPSEC_EXT.1.10

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

a) Test 1: 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.

b) Test 2: 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.

FCS_IPSEC_EXT.1.11

418 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.

FCS_IPSEC_EXT.1.12

419 The evaluator simply follows the guidance to configure the TOE to perform the following tests.

a) 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.

b) 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.

c) 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.

d) 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.

FCS_IPSEC_EXT.1.13

420 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 method selected in FCS_IPSEC_EXT.1.13:

a) Test 1: If pre-shared keys are selected, 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.

FCS_IPSEC_EXT.1.14

421 In the context of the tests below, a valid certificate is a certificate that passes FIA_X509_EXT.1 validation checks but does not necessarily contain an authorized subject.

422 The evaluator shall perform the following tests:

- Test 1: (conditional) For each CN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes CN checking over SAN (through explicit configuration of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the SAN so it contains an incorrect identifier of the correct type (e.g. the reference identifier on the TOE is example.com, the CN=example.com, and the SAN:FQDN=otherdomain.com) and verify that IKE authentication succeeds.
- Test 2: (conditional) For each SAN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes SAN checking over CN (through explicit specification of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the CN so it contains an incorrect identifier formatted to be the same type (e.g. the reference identifier on the TOE is DNS-ID; identify certificate has an identifier in SAN with correct DNS-ID, CN with incorrect DNS-ID (and not a different type of identifier)) and verify that IKE authentication succeeds.
- Test 3: (conditional) For each CN/identifier type combination selected, the evaluator shall:

b) Create a valid certificate with the CN so it contains the valid identifier followed by '\0'. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the evaluator shall configure the SAN so it matches the reference identifier.

c) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the CN without the '\0' and verify that IKE authentication fails.

- Test 4: (conditional) For each SAN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with an incorrect identifier in the SAN. The evaluator shall configure a string representation of the correct identifier in the DN. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the addition/modification shall be to any non-CN field of the DN. Otherwise, the addition/modification shall be to the CN.

b) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the correct identifier (expected in the SAN) and verify that IKE authentication fails.

- Test 5: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds.
- Test 6: (conditional) If the TOE supports DN identifier types, to demonstrate a bit-wise comparison of the DN, the evaluator shall create the following valid certificates and verify that the IKE authentication fails when each certificate is presented to the TOE:

a) Duplicate the CN field, so the otherwise authorized DN contains two identical CNs.

b) Append '\0' to a non-CN field of an otherwise authorized DN.

4.2.7 FCS_NTP_EXT.1 NTP Protocol

4.2.7.1 TSS

FCS_NTP_EXT.1.1

423 The evaluator shall examine the TSS to ensure identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained.

The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp.

4.2.7.2 Guidance Documentation

FCS_NTP_EXT.1.1

424 The evaluator shall examine the guidance documentation to ensure it provides the administrator instructions as how to configure the version of NTP supported, how to configure multiple NTP servers for the TOE's time source and how to configure the TOE to use the method(s) that are selected in the ST.

FCS_NTP_EXT.1.2

425 For each of the secondary selections made in the ST, the evaluator shall examine the guidance document to ensure it instructs the administrator how to configure the TOE to use the algorithms that support the authenticity of the timestamp and/or how to configure the TOE to use the protocols that ensure the integrity of the timestamp.

Assurance Activity Note:

Each primary selection in the SFR contains selections that specify a cryptographic algorithm or cryptographic protocol. For each of these secondary selections made in the ST, the evaluator shall examine the guidance documentation to ensure that the documentation instructs the administrator how to configure the TOE to use the chosen option(s).

FCS_NTP_EXT.1.3

426 The evaluator shall examine the guidance documentation to ensure it provides the administrator instructions as how to configure the TOE to not accept broadcast and multicast NTP packets that would result in the timestamp being updated.

4.2.7.3 Tests

FCS_NTP_EXT.1.1

427 The version of NTP selected in element 1.1 and specified in the ST shall be verified by observing establishment of a connection to an external NTP server known to be using the specified version(s) of NTP. This may be combined with tests of other aspects of FCS_NTP_EXT.1 as described below.

FCS_NTP_EXT.1.2

428 The cryptographic algorithms selected in element 1.2 and specified in the ST will have been specified in an FCS_COP SFR and tested in the accompanying Evaluation Activity for that SFR. Likewise, the cryptographic protocol selected in in element 1.2 and specified in the ST will have been specified in an FCS SFR and tested in the accompanying Evaluation Activity for that SFR.

[Conditional] If the message digest algorithm is claimed in element 1.2, the evaluator will change the message digest algorithm used by the NTP server in such a way that new value does not match the configuration on the TOE and confirms that the TOE does not synchronize to this time source.

The evaluator shall use a packet sniffer to capture the network traffic between the TOE and the NTP server. The evaluator uses the captured network traffic, to verify the NTP version, to observe time change of the TOE and uses the TOE's audit log to determine that the TOE accepted the NTP server's timestamp update.

The captured traffic is also used to verify that the appropriate message digest algorithm was used to authenticate the time source and/or the appropriate protocol was used to ensure integrity of the timestamp that was transmitted in the NTP packets.

FCS_NTP_EXT.1.3

429 The evaluator shall configure NTP server(s) to support periodic time updates to broadcast and multicast addresses. The evaluator shall confirm the TOE is configured to not accept broadcast and multicast NTP packets that would

result in the timestamp being updated. The evaluator shall check that the time stamp is not updated after receipt of the broadcast and multicast packets.

4.2.8 FCS_SSHC_EXT.1 SSH Client

4.2.8.1 TSS

FCS_SSHC_EXT.1.2

430 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication and that this list conforms to FCS_SSHC_EXT.1.5. and ensure that if password-based authentication methods have been selected in the ST then these are also described.

FCS_SSHC_EXT.1.3

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

FCS_SSHC_EXT.1.4

432 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.

FCS_SSHC_EXT.1.5

433 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.

FCS_SSHC_EXT.1.6

434 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.

FCS_SSHC_EXT.1.7

435 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.

FCS_SSHC_EXT.1.8

436 The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

b) Rekeying is performed upon reaching the threshold that is hit first.

4.2.8.2 Guidance Documentation

FCS_SSHC_EXT.1.4

437 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).

FCS_SSHC_EXT.1.5

438 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).

FCS_SSHC_EXT.1.6

439 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).

FCS_SSHC_EXT.1.7

440 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.

FCS_SSHC_EXT.1.8

441 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, then the evaluator shall check that the guidance documentation describes how to configure those thresholds. Either the allowed values are specified in the guidance documentation and must not exceed the limits specified in the SFR (one hour of session time, one gigabyte of transmitted traffic) or the TOE must not accept values beyond the limits specified in the SFR. The evaluator shall check that the guidance documentation describes that the TOE reacts to the first threshold reached.

4.2.8.3 Tests

FCS_SSHC_EXT.1.2

442 Test 1: If password-based authentication methods have been selected in the ST then 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.

Note: Public key authentication is tested as part of testing for FCS_SSHC_EXT.1.5

FCS_SSHC_EXT.1.3

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

FCS_SSHC_EXT.1.4

444 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a SSH connection with a remote server (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

FCS_SSHC_EXT.1.5

445 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.

446 Test 2: The evaluator shall configure an SSH server to only allow a public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

FCS_SSHC_EXT.1.6

447 Test 1: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall establish an SSH connection using each of the algorithms, except "implicit", 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.

448 Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test.

449 Test 2: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall configure an SSH server to only allow a MAC algorithm that is not included in the ST selection. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

450 Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test.

FCS_SSHC_EXT.1.7

451 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.

FCS_SSHC_EXT.1.8

452 The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

453 For testing of the time-based threshold the evaluator shall use the TOE to connect to an SSH server and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

454 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one hour of session time but the value used for testing shall not exceed one hour. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH server the TOE is connected to.

455 For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH server, and shall transmit data from and to the TOE within the active SSH session until the threshold for transmitted traffic is reached. The transmitted traffic is the total traffic comprising incoming and outgoing traffic.

456 The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

457 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one gigabyte of transferred traffic but the value used for testing shall not exceed one gigabyte. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH server the TOE is connected to.

458 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1/Functions).

459 In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

a) An argument is present in the TSS section describing this hardware-based limitation and

b) All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

FCS_SSHC_EXT.1.9

460 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.

461 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. If 'password-based' is selected for the TOE in FCS_SSHC_EXT.1.2, 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). If 'password-based' is not selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using public key-based authentication, and shall ensure that the TOE rejects the connection.

4.2.9 FCS_SSHS_EXT.1 SSH Server

4.2.9.1 TSS

FCS_SSHS_EXT.1.2

462 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication and that this list conforms to FCS_SSHS_EXT.1.5. and ensure that if password-based authentication methods have been selected in the ST then these are also described.

FCS_SSHS_EXT.1.3

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

FCS_SSHS_EXT.1.4

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 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.

FCS_SSHS_EXT.1.5

465 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.

FCS_SSHS_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.

FCS_SSHS_EXT.1.7

467 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.

FCS_SSHS_EXT.1.8

468 The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

b) Rekeying is performed upon reaching the threshold that is hit first.

4.2.9.2 Guidance Documentation

FCS_SSHS_EXT.1.4

469 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).

FCS_SSHS_EXT.1.5

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).

FCS_SSHS_EXT.1.6

471 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).

FCS_SSHS_EXT.1.7

472 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.

FCS_SSHS_EXT.1.8

473 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, then the evaluator shall check that the guidance documentation describes how to configure those thresholds. Either the allowed values are specified in the guidance documentation and must not exceed the limits specified in the SFR (one hour of session time, one gigabyte of transmitted traffic) or the TOE must not accept values beyond the limits specified in the SFR. The evaluator shall check that the guidance documentation describes that the TOE reacts to the first threshold reached.

4.2.9.3 Tests

FCS_SSHS_EXT.1.2

474 Test 1: If password-based authentication methods have been selected in the ST then using the guidance documentation, the evaluator shall configure the TOE to accept password-based authentication, and demonstrate that user authentication succeeds when the correct password is provided by the user.

475 Test 2: If password-based authentication methods have been selected in the ST then the evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

Note: Public key authentication is tested as part of testing for FCS_SSHS_EXT.1.5.

FCS_SSHS_EXT.1.3

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

FCS_SSHS_EXT.1.4

477 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a SSH connection from a remote client (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify

that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

FCS_SSHS_EXT.1.5

478 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.

479 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.

480 Test 3: The evaluator shall configure an SSH client to only allow a public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

FCS_SSHS_EXT.1.6

481 Test 1: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall establish an SSH connection using each of the algorithms, except "implicit", 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.

482 Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test.

483 Test 2: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall configure an SSH client to only allow a MAC algorithm that is not included in the ST selection. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

484 Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*-gcm@openssh.com encryption algorithm is negotiated while performing this test.

FCS_SSHS_EXT.1.7

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

486 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.

FCS_SSHS_EXT.1.8

487 The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

488 For testing of the time-based threshold the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

489 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one hour of session time but the value used for testing shall not exceed one hour. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

490 For testing of the traffic-based threshold the evaluator shall use an SSH client to connect to the TOE, and shall transmit data from and to the TOE within the active SSH session until the threshold for transmitted traffic is reached. The transmitted traffic is the total traffic comprising incoming and outgoing traffic.

491 The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

492 Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one gigabyte of transferred traffic but the value used for testing shall not exceed one gigabyte. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

493 If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1/Functions).

494 In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

c) An argument is present in the TSS section describing this hardware-based limitation and

d) All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet

Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

495

4.2.10 FCS_TLSC_EXT.1 Extended: TLS Client Protocol

4.2.10.1 TSS

FCS_TLSC_EXT.1.1

496 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.

FCS_TLSC_EXT.1.2

497 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. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies if certificate pinning is supported or used by the TOE and how it is implemented.

498 Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process. The TSS should describe the discovery process and highlight how the reference identifier is supplied to the "joining" component.

FCS_TLSC_EXT.1.4

499 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.

4.2.10.2 Guidance Documentation

FCS_TLSC_EXT.1.1

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

FCS_TLSC_EXT.1.2

501 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.

FCS_TLSC_EXT.1.4

502 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.

4.2.10.3 Tests

FCS_TLSC_EXT.1.1

503 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 to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

504 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.

505 Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

506 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.

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

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

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.

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.

d) If using DHE or ECDH, 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. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

e) Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and ACK message. This is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.

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

FCS_TLSC_EXT.1.2

508 Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a "Gatekeeper" discovery process.

509 The evaluator shall configure the reference identifier per the AGD guidance and perform the following tests during a TLS connection:

a) Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

b) 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.

c) Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, 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. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

d) 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.

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

1) 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.

2) 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. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

f) 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.

g) 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.

FCS_TLSC_EXT.1.3

510 The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

511 Test 1: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

512 Test 2: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the

expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

513 Test 3[conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

FCS_TLSC_EXT.1.4

514 Test 1: If using ECDHE ciphers, 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.

4.2.11 FCS_TLSC_EXT.2 Extended: TLS Client Protocol with authentication

4.2.11.1 TSS

FCS_TLSC_EXT.2.1

515 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.

FCS_TLSC_EXT.2.2

516 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. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies if certificate pinning is supported or used by the TOE and how it is implemented.

FCS_TLSC_EXT.2.4

517 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.

FCS_TLSC_EXT.2.5

518 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

4.2.11.2 Guidance Documentation

FCS_TLSC_EXT.2.1

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

FCS_TLSC_EXT.2.2

520 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.

FCS_TLSC_EXT.2.4

521 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.

FCS_TLSC_EXT.2.5

522 If the TSS indicates that mutual authentication using X.509v3 certificates is used, the evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for TLS mutual authentication.

4.2.11.3 Tests

FCS_TLSC_EXT.2.1

523 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 to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

524 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.

525 Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

526 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.

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

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

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.

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.

d) If using DHE or ECDH, 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. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS then this test shall be omitted.

e) Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and ACK message. This is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.

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

FCS_TLSC_EXT.2.2

528 The evaluator shall configure the reference identifier per the AGD guidance and perform the following tests during a TLS connection:

a) Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

b) 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.

c) Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, 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. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

d) 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.

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

1) 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.

2) 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. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

f) 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.

g) 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.

FCS_TLSC_EXT.2.3

529 The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

530 Test 1: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

531 Test 2: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

532 Test 3[conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

FCS_TLSC_EXT.2.4

533 Test 1: If using DHE or ECDH, 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.

FCS_TLSC_EXT.2.5

534 The purpose of these tests is to confirm that the TOE appropriately handles connection to peer servers that support and do not support mutual authentication.

535 Test 1: The evaluator shall establish a connection to a peer server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.

536 Test 2: The evaluator shall establish a connection to a peer server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of

a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) messages.

4.2.12 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

4.2.12.1 TSS

FCS_TLSS_EXT.1.1

537 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.

FCS_TLSS_EXT.1.2

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

FCS_TLSS_EXT.1.3

539 The evaluator shall verify that the TSS describes the key agreement parameters of the server Key Exchange message.

4.2.12.2 Guidance Documentation

FCS_TLSS_EXT.1.1

540 The evaluator shall 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).

FCS_TLSS_EXT.1.2

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

FCS_TLSS_EXT.1.3

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

4.2.12.3 Tests

FCS_TLSS_EXT.1.1

543 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 to discern the ciphersuite being used

(for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

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

545 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the key exchange message.

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

a) withdrawn

b) withdrawn

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.

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.

e) (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)

The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished

message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

FCS_TLSS_EXT.1.2

547 The evaluator shall send a Client Hello requesting a connection for all mandatory and selected protocol versions in the SFR (e.g. by enumeration of protocol versions in a test client) and verify that the server denies the connection for each attempt.

FCS_TLSS_EXT.1.3

548 If using ECDHE ciphers, the evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve. Using a packet analyser, 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.

549 The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_TLSS_EXT.1.3. For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.

550 Note that this testing can be accomplished in conjunction with other testing activities

4.2.13 FCS_TLSS_EXT.2 Extended: TLS Server Protocol with mutual authentication

4.2.13.1 TSS

FCS_TLSS_EXT.2.1

551 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.

FCS_TLSS_EXT.2.2

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

FCS_TLSS_EXT.2.3

553 The evaluator shall verify that the TSS describes the key agreement parameters of the server Key Exchange message.

FCS_TLSS_EXT.2.4 and FCS_TLSS_EXT.2.5

554 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

FCS_TLSS_EXT.2.6

555 The evaluator shall verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

4.2.13.2 Guidance Documentation

FCS_TLSS_EXT.2.1

556 The evaluator shall 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).

FCS_TLSS_EXT.2.2

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

FCS_TLSS_EXT.2.3

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

FCS_TLSS_EXT.2.4 and FCS_TLSS_EXT.2.5

559 If the TSS indicates that mutual authentication using X.509v3 certificates is used, the evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for TLS mutual authentication.

FCS_TLSS_EXT.2.6

560 If the DN is not compared automatically to the Domain Name or IP address, username, or email address, then the evaluator shall ensure that the AGD guidance includes configuration of the expected DN or the directory server for the connection.

4.2.13.3 Tests

FCS_TLSS_EXT.2.1

561 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 to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

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

563 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the Key Exchange message.

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

a) withdrawn

b) withdrawn

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.

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.

e) (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)

The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

FCS_TLSS_EXT.2.2

565 The evaluator shall send a Client Hello requesting a connection for all mandatory and selected protocol versions in the SFR (e.g. by enumeration of protocol versions in a test client) and verify that the server denies the connection for each attempt.

FCS_TLSS_EXT.2.3

566 The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve. Using a packet analyser, 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.

567 The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_TLSS_EXT.2.3. For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.

568 Note that this testing can be accomplished in conjunction with the other testing activities

FCS_TLSS_EXT.2.4 and FCS_TLSS_EXT.2.5

569 Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

570 Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

571 Test 3 [conditional]: If the TOE supports sending a non-empty Certificate Authorities list in its Certificate Request message, the evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied. If the TOE doesn't support sending a non-empty Certificate Authorities list in its Certificate Request message, this test shall be omitted.

572 Test 4: The aim of this test is to check the response of the server when it receives a client identity certificate that is signed by an impostor CA (either Root CA or intermediate CA). To carry out this test the evaluator shall configure the client to send a client identity certificate with an issuer field that identifies a CA recognised by the TOE as a trusted CA, but where the key used for the signature on the client certificate does not in fact correspond to the CA certificate trusted by the TOE (meaning that the client certificate is invalid because its certification path does not in fact terminate in the claimed CA certificate). The evaluator shall verify that the attempted connection is denied.

573 Test 5: The evaluator shall perform the following modifications to the traffic:

a) Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.

b) Configure the server to require mutual authentication and then modify a byte in the signature block of the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

574 The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

575 Test 6: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

576 Test 7: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in

the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

577 Test 8[conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

FCS_TLSS_EXT.2.6

578 The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

4.3 Identification and Authentication (FIA)

4.3.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

4.3.1.1 TSS

579 The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

580 The TSS shall describe when revocation checking is performed and on what certificates. If the revocation checking during authentication is handled differently depending on whether a full certificate chain or only a leaf certificate is being presented, any differences must be summarized in the TSS section and explained in the Guidance.

581 It is expected that revocation checking is performed when a certificate is used in an authentication step. It is expected that revocation checking is performed on both leaf and intermediate CA certificates when a leaf certificate is presented to the TOE as part of the certificate chain during authentication. Revocation checking of any CA certificate designated a trust anchor is not

required. It is not sufficient to perform a revocation check of a CA certificate only when it is loaded onto the device.

4.3.1.2 Tests

582 The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

a) Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOEs trust store, or by setting up the trust store in a way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store).

   Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE's trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

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

c) 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 peer certificate and revocation of the peer 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. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore the revoked certificate(s) used for testing shall not be a trust anchor.

d) 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.

e) 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.)

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

g) 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.)

583 The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

584 The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

585 For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

a) Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

b) Test 2: The evaluator shall ensure that at least one of the CA certificates in the chain has a basicConstraints extension in which the CA flag is set to FALSE. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate with the CA flag set to FALSE to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

586 The evaluator shall repeat these tests for each distinct use of certificates. Thus, for example, use of certificates for TLS connection is distinct from use of certificates for trusted updates so both of these uses would be tested. But there is no need to repeat the tests for each separate TLS channel in FTP_ITC.1 and FTP_TRP.1/Admin (unless the channels use separate implementations of TLS).

4.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication

4.3.2.1 TSS

587 The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

588 The evaluator shall examine the TSS to confirm that it describes the behaviour 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. 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.

4.3.2.2 Tests

589 The evaluator shall perform the following test for each trusted channel:

590 The evaluator shall demonstrate that using a valid certificate that 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 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.

4.3.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests

4.3.3.1 TSS

591 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.

4.3.3.2 Guidance Documentation

592 The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request. 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 Certification Request.

4.3.3.3 Tests

593 The evaluator shall perform the following tests:

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

b) Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request 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 response message, and demonstrate that the function succeeds.

4.4 Protection of the TSF (FPT)

4.4.1 FPT_TST_EXT.2 Self tests based on certificates

4.4.1.1 Tests

594 The evaluator shall verify that the self test mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.

595 The evaluator shall use an invalid certificate and perform the self test. This attempt should fail. The evaluator shall use a certificate that does not have the Code Signing purpose and verify that the self test fails. The evaluator shall repeat the test using a valid certificate and a certificate that contains the Code Signing purpose and verify that the self test succeeds. Testing for this element is performed in conjunction with the assurance activities for FPT_TST_EXT.1.

596 It is not necessary to verify the revocation status of X.509 certificates during power-up.

4.4.2 FPT_TUD_EXT.2 Trusted Update based on certificates

4.4.2.1 TSS

597 The evaluator shall verify that the TSS describes how the TOE reacts if X.509 certificates are used for trusted updates and the administrator attempts to perform the trusted update using an expired certificate.

598 The TSS shall describe the point at which revocation checking is performed. It is expected that revocation checking is performed when a certificate is used when performing trusted updates. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device.

4.4.2.2 Guidance Documentation

599 The evaluator shall verify that the guidance documentation describes how the TOE reacts if X.509 certificates are used for trusted updates and the administrator attempts to perform the trusted update using an expired certificate. The description shall correspond to the description in the TSS.

4.4.2.3 Tests

600 The evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.

601 The evaluator shall digitally sign the update with an invalid certificate and verify that update installation fails. The evaluator shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. The evaluator shall repeat the test using a valid certificate and a certificate that contains the Code Signing purpose and verify that the application installation succeeds. The evaluator shall use a previously valid but expired certificate and verifies that the TOE reacts as described in the TSS and the guidance documentation. Testing for this element is performed in conjunction with the assurance activities for FPT_TUD_EXT.1.

602 The evaluator shall demonstrate that checking the validity of a certificate is performed at the time a certificate is used when performing trusted updates. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device.

4.5 Security management (FMT)

4.5.1 FMT_MOF.1/AutoUpdate

4.5.1.1 TSS

603 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

4.5.1.2 Tests

604 The evaluator shall try to enable and disable automatic checking for updates or automatic updates (whichever is supported by the TOE) without prior authentication as security administrator (by authenticating as a user with no administrator privileges or without user authentication). The attempt to enable/disable automatic checking for updates should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to enable/disable automatic checking for updates can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

605 The evaluator shall try to enable and disable automatic checking for updates or automatic updates (whichever is supported by the TOE) with prior authentication as security administrator. The attempt to enable/disable automatic checking for updates should be successful.

4.5.2 FMT_MOF.1/Functions Management of security functions behaviour

4.5.2.1 TSS

606 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

4.5.2.2 Tests

607 Test 1 (if ‘transmission of audit data to external IT entity’ is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to modify the security related parameters can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

608 Test 2 (if ‘transmission of audit data to external IT entity’ is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as security administrator. The effects of the modifications should be confirmed.

609 The evaluator does not have to test all possible values of the security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity but at least one allowed value per parameter.

610 Test 1 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The term ‘handling of audit data’ refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

611 Test 2 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data with prior authentication as security administrator. The effects of the modifications should be confirmed. The term ‘handling of audit data’ refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

612 The evaluator does not necessarily have to test all possible values of the security related parameters for configuration of the handling of audit data but at least one allowed value per parameter.

613 Test 1 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). This attempt should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

614 Test 2 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full with prior authentication as security administrator. This attempt should be successful. The effect of the change shall be verified.

615 The evaluator does not necessarily have to test all possible values for the behaviour when Local Audit Storage Space is full but at least one change between allowed values for the behaviour.

616 Test 3 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions without administrator authentication shall fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

617 Test 4 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection with prior authentication as security administrator. This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions with administrator authentication shall be successful.

4.5.3 FMT_MOF.1/Services

4.5.3.1 TSS

618 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

4.5.3.2 Tests

619 The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) without prior authentication as security administrator (either by authenticating as a user with no administrator privileges, if possible, or without prior authentication at all). The attempt to enable/disable this service/these services should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to enable/disable this service/these services can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

620 The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) with prior authentication as security administrator. The attempt to enable/disable this service/these services should be successful.

4.5.4 FMT_MTD.1/CryptoKeys Management of TSF Data

4.5.4.1 TSS

621 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

4.5.4.2 Tests

622 The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

623 The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

5 Evaluation Activities for SARs

624 The sections below specify EAs for the Security Assurance Requirements (SARs) included in the related cPPs (see section 1.1 above). 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.

625 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

5.1.1 General ASE

626 When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2 (Evaluation Activities for SFRs).

5.1.2 TOE summary specification (ASE_TSS.1) for Distributed TOEs

627 For distributed TOEs only the SFRs classified as ‘all’ have to be fulfilled by all TOE parts. The SFRs classified as ‘One’ or ‘Feature Dependent’ only have to be fulfilled by either one or some TOE parts, respectively. To make sure that the distributed TOE as a whole fulfills all the SFRs the following actions for ASE_TSS.1 have to be performed as part of ASE_TSS.1.1E.

ASE_TSS.1 element	Evaluator Action
ASE_TSS.1.1C	The evaluator shall examine the TSS to determine that it is clear which TOE components contribute to each SFR or how the components combine to meet each SFR. The evaluator shall verify the sufficiency to fulfil the related SFRs. This includes checking that the TOE as a whole fully covers all SFRs and that all functionality that is required to be audited is in fact audited regardless of the component that carries it out.

628 Note that additional Evaluation Activities for the TSS in the case of a distributed TOE are defined in section B.4.1.1.

5.2 ADV: Development

5.2.1 Basic Functional Specification (ADV_FSP.1)

629 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 Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

630 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.

631 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.

632 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.

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.	5.2.1.1 Evaluation Activity: *The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.*
ADV_FSP.1-2 The evaluator ***shall examine*** the functional specification to determine that the	5.2.1.1 Evaluation Activity: *The evaluator shall examine the interface documentation*

method of use for each SFR-supporting and SFR-enforcing TSFI is given.	*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.	5.2.1.2 Evaluation Activity: *The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.*
ADV_FSP.1-4 The evaluator ***shall examine*** the rationale provided by the developer for the implicit categorisation of interfaces as SFR-non-interfering to determine that it is accurate.	Paragraph 561 from the CEM: "In the case where the developer has provided adequate documentation to perform the analysis called for by the rest of the work units for this component without explicitly identifying SFR-enforcing and SFR-supporting interfaces, this work unit should be considered satisfied." Since the rest of the ADV_FSP.1 work units will have been satisfied upon completion of the EAs, it follows that this work unit is satisfied as well.
ADV_FSP.1-5 The evaluator ***shall check*** that the tracing links the SFRs to the corresponding TSFIs.	5.2.1.3 Evaluation Activity: *The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.*
ADV_FSP.1-6 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the SFRs.	EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3 and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e. at the TSFI) are covered. Therefore, the intent of this work unit is covered.
ADV_FSP.1-7 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the SFRs.	EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3 and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e. at the TSFI) are addressed, and that the description of the interfaces is accurate with respect to the specification captured in the SFRs. Therefore, the intent of this work unit is covered.

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

5.2.1.1 Evaluation Activity:

633 *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.*

634 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.

635 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

636 *The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.*

5.2.1.3 Evaluation Activity

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

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

639 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.

640 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 AGD: Guidance Documents

641 It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and

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.

642 Note that additional Evaluation Activities for the guidance documentation in the case of a distributed TOE are defined in section B.4.1.1.

5.3.1 Operational User Guidance (AGD_OPE.1)

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

644 In addition, the evaluator performs the EAs specified below.

5.3.1.1 Evaluation Activity:

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

5.3.1.2 Evaluation Activity

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

5.3.1.3 Evaluation Activity

647 *The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.*

5.3.1.4 Evaluation Activity

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

5.3.1.5 Evaluation Activity

649 In addition the evaluator shall ensure that the following requirements are also met.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of

other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes instructions that describe at least one method of validating the hash/digital signature.

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

5.3.2 Preparative Procedures (AGD_PRE.1)

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

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

652 In addition, the evaluator performs the EAs specified below.

5.3.2.1 Evaluation Activity:

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

654 The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

5.3.2.2 Evaluation Activity

655 *The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as*

claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

5.3.2.3 Evaluation Activity

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

5.3.2.4 Evaluation Activity

657 *The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.*

5.3.2.5 Evaluation Activity

658 In addition the evaluator shall ensure that the following requirements are also met.

659 The preparative procedures must

a) include instructions to provide a protected administrative capability; and

b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

5.4 ALC: Life-cycle Support

5.4.1 Labelling of the TOE (ALC_CMC.1)

660 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)

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

5.5 ATE: Tests

5.5.1 Independent Testing – Conformance (ATE_IND.1)

662 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.

663 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, 3 and 4.

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

665 Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section B.4.3.1.

5.6 AVA: Vulnerability Assessment

5.6.1 Vulnerability Survey (AVA_VAN.1)

666 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.

667 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.

668

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. The calibration of test resources specified in paragraph 1418 of the CEM applies to the tools listed in 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 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 Section A.1, and documentation as specified in 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 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; 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.	The CEM work unit is captured in Section A.3; there are no substantive differences.
AVA_VAN.1-7 The evaluator ***shall conduct*** penetration testing.	The evaluator shall perform the CEM activity as specified. See Section A.2, paragraph 700 for guidance related to attack potential for confirmed flaws.
AVA_VAN.1-8 The evaluator ***shall record*** the actual results of the penetration tests.	The evaluator shall perform the CEM activity as specified.

AVA_VAN.1-9 The evaluator ***shall report*** in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.	Replace the CEM work unit with the reporting called for in Section A.3.
AVA_VAN.1-10 The evaluator ***shall examine*** the results of all penetration testing to determine that the TOE, in its operational environment, is resistant to an attacker possessing a Basic attack potential.	This work unit is not applicable for Type 1 and Type 2 flaws (as defined in Section A.1), as inclusion in this Supporting Document by the iTC makes any confirmed vulnerabilities stemming from these flaws subject to an attacker possessing a Basic attack potential. This work unit is replaced for Type 3 and Type 4 flaws by the activities defined in Section A.2, paragraph 700.
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 TOE, level of opportunity and the equipment required to perform the identified vulnerabilities, and the corresponding values using the tables 3 and 4 of Annex B.4.	Replace the CEM work unit with the reporting called for in Section A.3.

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

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

5.6.1.1 Evaluation Activity (Documentation):

670 In addition to the activities specified by the CEM in accordance with Table 2, the evaluator shall perform the following activities.

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

672 The developer shall provide documentation identifying the list of software and hardware components[7] that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

673 If the TOE is a distributed TOE then the developer shall provide:

a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]

b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.4.1.2 and 3.5.1.2.

5.6.1.2 Evaluation Activity:

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

[7] In this sub-section the term "components" refers to parts that make up the TOE. It is therefore distinguished from the term "distributed TOE components", which refers to the parts of a TOE that are present in one physical part of a distributed TOE. Each distributed TOE component will therefore generally include a number of the hardware and software components that are referred to in this sub-section: for example, each distributed TOE component will generally include hardware components such as processors and software components such as an operating system and libraries.

6 Required Supplementary Information

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

676 The cPPs associated with this SD require an entropy analysis as described in [NDcPP, Appendix D].

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,
Evaluation Methodology,
CCMB-2017-04-004, Version 3.1, Revision 5, April 2017

[FIPS 140-2] FIPS PUB 140-2, Security Requirements for cryptographic modules, May 25 2001 with change notices (12-03-2002)

[FIPS 186-4] FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013

[FWcPP] collaborative Protection Profile for Stateful Traffic Filter Firewalls,
Version 2.0, May 2017

[NDcPP] collaborative Protection Profile for Network Devices, Version 2.1, 24 September 2018

[NIST SP800-56A] NIST Special Publication SP800-56B Revision 3: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, Apr 2018

[NIST SP800-56B] NIST Special Publication SP800-56B Revision 1: Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography, March 2014

References

[NIST SP800-90A] NIST Special Publication SP800-90A Deterministic Random Bit Generator Validation System (DRBGVS), October 29 2015

[SHAVS] The Secure Hash Algorithm Validation System (SHAVS), Updated: May 21, 2014

Appendices

A. Vulnerability Analysis

A.1 Sources of vulnerability information

677 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 (in this case, a network device) derived from public sources as documented in Section A.1.1 – this fixed set has been agreed 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 listed 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 developer described in this Supporting Document (documentation associated with EAs, documentation described in Section 5.6.1.2, documentation described in Section 6), 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 TC-defined tools (e.g., nmap, fuzz testers) and their application as specified in section A.1.4.

A.1.1 Type 1 Hypotheses – Public-Vulnerability-Based

678 The list of public sources of vulnerability information selected by the iTC is given in Section A.4.

679 The evaluators shall perform a search on the sources listed in Section A.4 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.

680 The search criteria to be used when searching the sources published after the publication date of the cPP shall include:

- The terms “router” and “switch” (or similar generic term describing the device type of the TOE)
- The following protocols: TCP
- Any protocols not listed above supported (through an SFR) by the TOE (these will include at least one of the remote management protocols (IPsec, TLS, SSH))
- The TOE name (including appropriate model information as appropriate)

681 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

682 Section A.5 contains the list of flaw hypothesis generated by the iTC for this technology that must be considered by the evaluation team as flaw hypotheses in performing the vulnerability assessment.

683 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

684 Type 3 flaws are formulated by the evaluator based on information presented by the product (through on-line help, product documentation and user guides, etc.) and product behaviour during the (functional) testing activities. The evaluator is also free to formulate flaws that are based on material that is not part of the baseline evidence (e.g., information gleaned from an Internet mailing list, or reading interface documentation on interfaces not included in the set provided by the developer), although such activities have the potential to vary significantly based upon the product and evaluation facility performing the analysis.

685 If the evaluators discover a Type 3 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.1.4 Type 4 Hypotheses – Tool-Generated

686 The evaluator shall perform the following activities to generate type 4 flaw hypotheses:

- Fuzz testing
    - Examine effects of sending:

- mutated packets carrying each ‘Type’ and ‘Code’ value that is undefined in the relevant RFC for each of ICMPv4 (RFC 792) and ICMPv6 (RFC 4443)
- mutated packets carrying each ‘Transport Layer Protocol’ value that is undefined in the respective RFC for IPv4 (RFC 791) IPv6 (RFC 2460) should also be covered if it is supported and claimed by the TOE.

Since none of these packets will belong to an allowed session, the packets should not be processed by the TOE, and the TOE should not be adversely affected by this traffic. Any results that are unexpected (e.g., core dumps) are candidates for a flaw hypothesis.

- Mutation fuzz testing of the remaining fields in the required protocol headers. This testing requires sending mutations of well-formed packets that have both carefully chosen and random values inserted into each header field in turn (i.e. testing is to include both carefully chosen and random insertion test cases). The original well-formed packets would be accepted as part of a normal existing communication stream and may still be accepted as valid packets when subject to the carefully chosen mutations (the individual packet alone would be valid although its contents may not be valid in the context of preceding and/or following packets), but will often not be valid packets when random values are inserted into fields. The carefully chosen values should include semantically significant values that can be determined from the type of the data that the field represents, such as values indicating positive and negative integers, boundary conditions, invalid binary combinations (e.g. for flag sets with dependencies between bits), and missing start or end values. Randomly chosen values may not result in well-formed packets, but are included nonetheless to see whether they can lead to the device entering an insecure state. Any results that are unexpected (e.g., core dumps) are candidates for a flaw hypothesis.

687 The iTC has not identified a specific tool to be used in accomplishing the above flaw hypothesis generation activity, so any tool used by the evaluation team is acceptable. The evaluation team shall record in the test report the name, version, parameters, and results of all test tools used for this this activity.

688 If the evaluators discover a 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.2 Process for Evaluator Vulnerability Analysis

689 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.

690 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 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 all of 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:

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

691 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).

692 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.

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

694 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.

695 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).

696 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.

697 Disagreements between evaluator and developer 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.

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

699 As indicated in Section A.3, 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.

700 For flaws of Types 3 and 4, each CB will be responsible for determining what constitutes Basic Attack Potential for the purposes of determining whether a flaw is exploitable in the TOE's environment. The determination criteria shall be documented in the CB-internal report as specified in Section A.3. As this is a per-CB activity, no public claims are made with respect to the resistance of a particular TOE against flaws of Types 3 and 4; rather, the claim is that the activities outlined in this appendix were carried out, and the evaluation team and CB agreed that any residual vulnerabilities are not exploitable by an attacker with Basic Attack Potential.

A.3 Reporting

701 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.

702 The public-facing report contains:

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

703 A statement that the evaluation team developed Types 3 and 4 flaw hypotheses in accordance with Sections A.1.3, A.1.4, and A.2, and that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential as defined by the CB in accordance with the guidance in the CEM. It should be noted that this is just a statement about the "fact of" Types 3 and 4 flaw hypotheses being developed, and that no specifics about the number of

flaws, the flaws themselves, or the analysis pertaining to those flaws will be included in the public-facing report.

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

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

- a list of all of the flaw hypotheses generated (cf. AVA_VAN.1-4);
- the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results (cf. AVA_VAN.1-9);
- all documentation used to generate the flaw hypotheses (in identifying the documentation used in coming up with the flaw hypotheses, the evaluation team must characterize the documentation so that a reader can determine whether it is strictly required by this Supporting Document, and the nature of the documentation (design information, developer engineering notebooks, etc.));
- the evaluator shall report all exploitable vulnerabilities and residual vulnerabilities, detailing for each:
    - its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);
    - the SFR(s) not met;
    - a description;
    - whether it is exploitable in its operational environment or not (i.e. exploitable or residual).
    - the amount of time, level of expertise, level of knowledge of the TOE, level of opportunity and the equipment required to perform the identified vulnerabilities (cf. AVA_VAN.1-11);
- how each flaw hypothesis was resolved (this includes whether the original flaw hypothesis was confirmed or disproved, and any analysis relating to whether a residual vulnerability is exploitable by an attacker with Basic Attack Potential) (cf. AVA_VAN1-10); and
- in the case that actual testing was performed in the investigation (either as part of flaw hypothesis generation using tools specified by the iTC in Section A.1.4, or in proving/disproving a particular flaw) the steps followed in setting up the TOE (and any required test equipment); executing the test; post-test procedures; and the actual results (to a level of detail that allow repetition of the test, including the following:
    - identification of the potential vulnerability the TOE is being tested for;

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

A.4 Public Vulnerability Sources

706 The following sources of public vulnerabilities are sources for the iTC to consider in both formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform key-word searches during the evaluation of a specific TOE.

a) NIST National Vulnerabilities Database (can be used to access CVE and US-CERT databases identified below):
https://web.nvd.nist.gov/view/vuln/search

b) Common Vulnerabilities and Exposures:
http://cve.mitre.org/cve/
https://www.cvedetails.com/vulnerability-search.php

c) US-CERT:
http://www.kb.cert.org/vuls/html/search

d) Exploit / Vulnerability Search Engine:
www.exploitsearch.net

e) SecurITeam Exploit Search:
www.securiteam.com

f) Tenable Network Security
http://nessus.org/plugins/index.php?view=search

g) Tipping Point Zero Day Initiative
http://www.zerodayinitiative.com/advisories

h) Offensive Security Exploit Database:
https://www.exploit-db.com/

i) Rapid7 Vulnerability Database:
https://www.rapid7.com/db/vulnerabilities

A.5 Additional Flaw Hypotheses

707 No entries are currently defined for this list.

B. Network Device Equivalency Considerations

B.1 Introduction

708 This appendix provides a foundation for evaluators to determine whether a developer's request for equivalency of products for different models wishing to claim conformance to the Network Device collaborative Protection Profiles is allowed.

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

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

709 Determination of equivalency between models can result in several different testing outcomes.

710 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.

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

712 If a developer disagrees with the evaluator's assessment of equivalency, the Certification Body arbitrates between the two parties as to whether equivalency exists.

B.2 Evaluator guidance for determining equivalence

B.2.1 Strategy

713 When performing the equivalency analysis, the evaluator should consider each factor independently. A factor may be any number of things at various levels

of abstraction, ranging from the processor a device uses, to the underlying operating system and hardware platform a software application relies upon. Examples may be the various chip sets employed by the product, the type of network interface (different device drivers), storage media (solid state drive, spinning disk, EEPROM). It is important to consider how the difference in these factors may influence the TOE's ability to enforce the SFRs. Each analysis of an individual factor will result in one of two outcomes,

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

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

B.2.2 Guidance for Network Devices

715 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.

Factor	Same/Not Same	Evaluator guidance
Platform/Hardware Dependencies	Independent	If there are no identified platform/hardware dependencies, the evaluator shall consider testing on multiple hardware platforms to be equivalent.
	Dependencies	If there are specified differences between platforms/hardware, the evaluator must identify if the differences affect the cPP-specified security functionality or if they apply to non-cPP-specified functionality. If functionality specified in the cPP is dependent upon platform/hardware provided services, the product must be tested on each of the different platforms 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-cPP-specified functionality, the variations may still be considered equivalent. For each

Factor	Same/Not Same	Evaluator guidance
		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.
Differences in Libraries Used to Provide TOE Functionality	Same	If there are no differences between the libraries used in various TOE models, the model variations shall be considered equivalent.
	Different	If the separate libraries are used between model variations, a determination of whether the functionality provided by the library affects cPP-specified functionality must be made. If cPP-specified functionality is affected, the models are not considered equivalent and must be tested separately. The evaluator has the option of only retesting the functionality that was affected by the differences in the included libraries. If the different libraries only affect non-PP specified functionality, the models may still be considered equivalent. For each different library, the evaluator must provide an explanation of why the different libraries do or do not affect cPP specified functionality.
TOE Management Interface Differences	Consistent	If there are no differences in the management interfaces between various TOE models, the models variations shall be considered equivalent.
	Differences	If the product provides separate interfaces based on the model variation, a determination must be made of whether cPP-specified functionality can be configured by the different interfaces. If the interface differences affect cPP-specified

Factor	Same/Not Same	Evaluator guidance
		functionality, the variations 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.

Table 3: Evaluation Equivalency Analysis

B.3 Test presentation/Truth in advertising

716 In addition to determining what to test, the evaluation results and resulting Certification Report, must identify the actual module and 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.

B.4 Evaluating additional components for a distributed TOE

717 In the case of a distributed TOE the Security Target will identify an evaluated configuration that consists of a number of separate components chosen by the

ST author, which collectively satisfy the requirements of the cPP. This evaluated configuration need not be the minimum set of components that could *possibly* meet the cPP (e.g. if the TOE is intended for large enterprise deployments then the evaluated configuration might include some redundancy in components in order to support expected connectivity and loads), but because this is the main configuration referred to in the ST and the evaluation, it is treated in this section as the minimum configuration of interest and is referred to here as the 'minimum configuration' as well as the 'evaluated configuration'.

718 In addition to the minimum configuration above, the ST may also identify (at the author's discretion, and subject to verification as described in this section) which TOE components can have instances added to an operational configuration without affecting the validity of the CC certification. The ST description may include constraints on how such components are added, including required and/or prohibited configurations of the components.

719 Extra instances of a TOE component must have the same hardware and software as the original component included in the evaluated configuration.

720 It is noted that undesirable configurations may be possible in the operational deployment of a TOE – such as allowing a TOE component to be managed from separate and potentially conflicting administration domains. However, the definition of 'undesirable' and of the risks involved in such cases will be specific to each operational environment and is therefore not treated as part of the evaluation. Correct and appropriate configuration of this sort remains a matter for expert network planning and design in the operational environment.

B.4.1 Evaluator Actions for Assessing the ST

B.4.1.1 TSS

721 The evaluator shall examine the TSS to identify any extra instances of TOE components allowed in the ST and shall examine the description of how the additional components maintain the SFRs to confirm that it is consistent with the role that the component plays in the evaluated configuration. For example: the secure channels used by the extra component for intra-TOE communications (FPT_ITT) and external communications (FTP_ITC) must be consistent, the audit information generated by the extra component must be maintained, and the management of the extra component must be consistent with that used for the original instance of the component in the minimum configuration.

B.4.2 Evaluator Actions for Assessing the Guidance Documentation

B.4.2.1 Guidance Documentation

722 The evaluator shall examine the description of the extra instances of TOE components in the guidance documentation to confirm that they are consistent with those identified as allowed in the ST. This includes confirmation that the result of applying the guidance documentation to configure the extra

component will leave the TOE in a state such that the claims for SFR support in each component are as described in the ST and therefore that all SFRs continue to be met when the extra components are present.

723 The evaluator shall examine the secure communications described for the extra components to confirm that they are the same as described for the components in the minimum configuration (additional connections between allowed extra components and the components in the minimum configuration are allowed of course).

B.4.3 Evaluator Actions for Testing the TOE

B.4.3.1 Tests

724 The evaluator tests the TOE in the minimum configuration as defined in the ST (and the guidance documentation).

725 If the description of the use of extra components in the ST and guidance documentation identifies any difference in the SFRs allocated to a component, or the scope of the SFRs involved (e.g. if different selections apply to different instances of the component) then the evaluator tests these additional SFR cases that were not included in the minimum configuration.

726 In addition the evaluator tests the following aspects for each extra component that is identified as allowed in the distributed TOE:

- Communications: the evaluator follows the guidance documentation to confirm, by testing, that any additional connections introduced with the extra component and not present in the minimum configuration are consistent with the requirements stated in the ST (e.g. with regard to protocols and ciphersuites used). An example of such an additional connection would be if a single instance of the component is present in the minimum configuration and adding a duplicate component then introduces an extra communication between the two instances. Another example might be if the use of the additional components necessitated the use of a connection to an external authentication server instead of using locally stored credentials.
- Audit: the evaluator confirms that the audit records from different instances of a component can be distinguished so that it is clear which instance generated the record.
- Management: if the extra component manages other components in the distributed TOE then the evaluator shall follow the guidance documentation to confirm that management via the extra component uses the same roles and role holders for administrators as for the component in the minimum configuration.