Supporting Document - Evaluation Activities for Network Device cPP

Version: 4.0
Date: 25 November 2025

Acknowledgements

This collaborative Protection Profile (cPP) was developed by the Network Device international Technical Community (ND iTC) with representatives from industry, government agencies, Common Criteria Test Laboratories, and members of academia.

Preface

This is a Supporting Document (SD), intended to complement the Common Criteria (CC) Version CC:2022, Revision 1 and the associated Common Evaluation Methodology for Information Technology Security Evaluation (CEM).

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 recognised under the Common Criteria Recognition Arrangement (CCRA). This SD shall be considered a Mandatory Technical Document.

This SD has been developed by the ND iTC, in accordance with [CC4]. The SD is intended to be used to support the evaluations of products against the collaborative Protection Profile (cPP) identified in Section 1.1.

Technical Editor: Network Device International Technical Community (ND iTC)

Revision History:

Version	Date	Description
4.0	25-Nov-2025	Released for use

Version	Date	Description
3.0e	06-Dec-2023	Released for use
3.0	06-Apr-2023	Incorporated comments received. Released for use
2.2	20-Dec-2019	Released for use
2.1	17-Aug-2018	Released for use
2.0	5-May-2017	Released for use
1.1	21-Jul-2016	Updated draft published for public review
1.0	27-Feb-2015	Released for use
0.4	26-Jan-2015	Incorporated comments received from the CCDB review
0.3	17-Oct-2014	Draft version released to accompany CCDB review of Supporting Document.

Version	Date	Description
0.2	13-Oct-2014	Internal draft in response to public review comments, for iTC review
0.1	05-Sep-2014	Draft published 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].

Table of Contents

1. Introduction

1.1. Technology Area and Scope of Supporting Document

1. This Supporting Document (SD) defines the Evaluation Activities (EA) associated with the collaborative Protection Profile for Network Devices [NDcPP].
2. The ND 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 Security Functional Requirements (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 the following cPP:
    a. collaborative Protection Profile for Network Devices [NDcPP]
4. Although Evaluation Activities (EAs) 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 of the product 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 (STs) (especially the TOE Summary Specification (TSS)), Administrator Guidance Documentation (AGD), 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 SFRs and Security Assurance Requirements (SARs). 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 (CB) for the evaluation and documented in the evaluation report.

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'.

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'.

1.3. Application of this Supporting Document

9. This Supporting Document defines three types of Evaluation Activities: TOE Summary Specification, Guidance Documentation, and Tests, and is designed to be used in conjunction with a cPP. 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 the cPP and this 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.

1.4. Terminology

1.4.1. Glossary

13. For definitions of standard CC terminology see [CC1].

Term	Meaning
Administrator	See Security Administrator.
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
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 authorised access to the TOE to perform configuration and management tasks.
Supplementary Information	Information that is not necessarily included in the ST 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 or PP-Module.
Target of Evaluation (TOE)	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]

Term	Meaning
TSF Data	Data for the operation of the TSF upon which the enforcement of the requirements relies.

1.4.2. Acronyms

Acronym	Meaning
AGD	Assurance: Guidance Documents
API	Application Programming Interface
CA	Certificate Authority
CCM	Counter with CBC-MAC
CCRA	Common Criteria Recognition Arrangement
CEM	Common Methodology for Information Technology Security Evaluation
CMAC	Cipher-based Message Authentication Code
CN	Common Name
cPP	collaborative Protection Profile

Acronym	Meaning
CRL	Certificate Revocation List
CTR	Counter (mode)
CVE	Common Vulnerabilities and Exposures (database)
DHE	Ephemeral Diffie-Hellman Key Exchange
DN	Distinguished Name
DNS	Domain Name Service
DRBG	Deterministic Random Bit Generator
EA	Evaluation Activity
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
FFC	Finite Field Cryptography
FQDN	Fully Qualified Domain Name
I&A	Identity and Authentication

Acronym	Meaning
IKE	Internet Key Exchange
iTC	International Technical Community
IV	Initialization Vector
LMS	Leighton-Micali Signature
MAC	Message Authentication Code
MD	Message Digest
ML-DSA	Module-Lattice-Based Digital Signature Algorithm
ML-KEM	Module-Lattice-Based Key Encapsulation Mechanism
MODP	Modular Exponential (Diffie-Hellman group type)
NIST	National Institute of Standards and Technology
OE	Operational Environment
OCSP	Online Certificate Status Protocol
RBG	Random Bit Generator

Acronym	Meaning
SA	Security Association (IPsec)
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SD	Supporting Document
SFR	Security Functional Requirement
SPD	Security Policy Database
SSH	Secure Shell
TLS	Transport Layer Security
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification
XMSS	eXtended Merkle Signature Scheme
XOF	eXtendable Output Function

Acronym	Meaning
XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing

2. Evaluation Activities for SFRs

14. The EAs presented in this section capture the actions the evaluator shall perform 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 TOE Summary Specification (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 Security Administrators as it pertains to SFRs, the evaluator's verdicts will be associated with CEM work units 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. Approval for using tests created by developers is up to the CB. 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, see 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

20. The main reasons for collecting audit information are to detect and identify error conditions, security violations, etc. and to provide sufficient information to the Security Administrator to resolve the issue. The audit information to be collected according to FAU_GEN.1, and the failure conditions identified in tables 2, 11, and 12 need to enable the Security Administrator at least to detect and identify the problem and provide at least basic information to resolve the issue. Also for this level of detail, the other FAU requirements apply, in particular the need for local and remote storage of audit information according to FAU_STG_EXT.1.

21. The level of detail that needs to be provided to the Security Administrator to actually resolve an issue usually depends on the complexity of the underlying use case. It is expected that a product provides additional levels of auditing to support resolution of error conditions, security violations, etc. beyond the level required by FAU_GEN.1, but it should also be clear that a high level of granularity cannot be maintained on most systems by default due to the high number of audit events that would be generated in such a configuration. It is expected that the TOE will be capable of auditing sufficient information to meet the requirements of FAU_GEN.1. If the TOE allows configuration of the level of auditing without taking the TOE out of the evaluated configuration, some of the audit events required by FAU_GEN.1 may only be recorded after corresponding configuration of the audit functionality.

22. The issue described above explicitly refers to the use of X.509v3 certificates. In the case that a certificate-based authentication fails, an error message telling the Security Administrator that 'something is wrong with the certificate' shall not be considered as sufficient information about the 'reason for failure' as a basic information to resolve the issue. The log message will inform the Security Administrator of at least the following:

    - 'Trust issue' with the certificate, e.g., due to failed path validation
    - Use of an 'expired certificate'
    - Absence of the basicConstraints extension
    - The CA flag is not set for a certificate presented as a CA
    - Signature validation failure for any certificate in the certificate path; failure to establish revocation status; revoked certificate

23. As such for audit information related to the use of X.509v3 certificates, that it uniquely identifies the certificate that could not be successfully verified. For example, identification of a certificate could include Key Subject and Key ID (where Key Subject is an identifier contained in the CN or SAN and where Key ID is a certificate's serial number and issuer name) or Subject Key Identifier (SKI) and Authority Key Identifier (AKI). In general, when using open source libraries like OpenSSL, passing on error messages from such libraries to the Security Administrator is regarded as good practice.

2.1.1.1. TSS

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

25. 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, 11, and 12 (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

26. The evaluator shall check the guidance documentation and ensure that it provides an example of each auditable event required by FAU_GEN.1 (i.e., at least one instance of each auditable event, comprising the mandatory, optional and selection-based SFR sections as applicable, shall be provided from the actual audit record).

27. 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

28. 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 the administrative actions listed above. This should include all instances of an event; for instance, if there are several different Identity and Authentication (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.

29. 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.
30. 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 and Guidance Documentation

31. 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

32. This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.
33. 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

34. 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.

35. 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.

36. The evaluator shall examine the TSS to ensure that it details whether the transmission of audit data to an external IT entity can be done in real-time, periodically, or both. In the case where the TOE is capable of performing transmission periodically, the evaluator shall verify that the TSS provides details about what event stimulates the transmission to be made as well as the possible acceptable frequency for the transfer of audit data.

37. 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 component 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).

38. The evaluator shall examine the TSS to ensure it describes the amount of audit data that can be stored locally and how these records are protected against unauthorised modification or deletion.

39. The evaluator shall examine the TSS to ensure it describes the method implemented for local logging, including format (e.g., buffer, log file, database) and whether the logs are persistent or non-persistent.

40. The evaluator shall examine the TSS to ensure it describes the conditions that must be met for authorised deletion of audit records.

41. The evaluator shall examine the TSS to ensure 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.

42. 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

43. 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.

44. The evaluator shall also examine the guidance documentation to ensure 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.

45. The evaluator shall examine the guidance documentation to ensure it describes any configuration required for protection of the locally stored audit data against unauthorised modification or deletion.

46. If the storage size is configurable, the evaluator shall review the Guidance Documentation to ensure it contains instructions on specifying the required parameters.

47. If more than one selection is made for FAU_STG_EXT.1.5, the evaluator shall review the Guidance Documentation to ensure it contains instructions on specifying which action is performed when the local storage space is full.

2.1.3.3. Tests

48. Testing of secure transmission of the audit data externally (FTP_ITC.1) and, where applicable, intercomponent (FPT_ITT.1 or FTP_ITC.1) shall be performed according to the assurance activities for the particular protocol(s).

49. The evaluator shall perform the following additional test 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: For distributed TOEs, Test 1 defined above shall be applicable to all TOE components that forward audit data to an external audit server.

c. Test 3: The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall then make note of whether the TSS claims persistent or non-persistent logging and perform one of the following actions:

    i. If persistent logging is selected, the evaluator shall perform a power cycle of the TOE and ensure that following power on operations the log events generated are still maintained within the local audit storage.

    ii. If non-persistent logging is selected, the evaluator shall perform a power cycle of the TOE and ensure that following power on operations the log events generated are no longer present within the local audit storage.

d. Test 4: 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.1.5. Depending on the configuration this means that the evaluator shall check the content of the audit data when the audit data is just filled to the maximum and then verifies that:

    i. 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.1.5).

    ii. 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.1.5)

    iii. The TOE behaves as specified (for the option 'other action' in FAU_STG_EXT.1.5).

e. Test 5: For distributed TOEs, for the local storage according to FAU_STG_EXT.1.4, Test 1 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, Test 2

specified above shall be applied. The evaluator shall verify that the transfer of audit data to an external audit server is implemented.

f. Test 6 [Conditional]: In case manual export or ability to view locally is selected in FAU_STG_EXT.1.6, during interruption the evaluator shall perform a TSF-mediated action and verify the event is recorded in the audit trail.

   Note: The intent of the test is to ensure that the local audit TSF (as specified by FAU_STG_EXT.1.3) operates independently from the ability to transmit the generated audit data to an external audit server (as specified in FAU_STG_EXT.1.1). There are no specific requirements on the interruption of the connection between the TOE and the external audit server (as for FTP_ITC.1).

2.2. Cryptographic Support (FCS)

2.2.1. FCS_CKM.1/AKG Cryptographic Key Generation – Asymmetric Key

2.2.1.1. TSS

50. The evaluator shall examine the TSS to verify that it describes how the TOE generates a key based on output from a random bit generator as specified in FCS_RBG.1. The evaluator shall review the TSS to verify that it describes how the functionality described by FCS_RBG.1 is invoked.
51. The evaluator shall examine the TSS to verify that it identifies the usage and key lifecycle for keys generated using each selected algorithm.
52. The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed in accordance with the relevant standards.
53. If the TOE uses the generated key in a key chain or hierarchy then the evaluator shall verify that the TSS describes how the key is used as part of the key chain or hierarchy.

2.2.1.2. Guidance Documentation

54. The evaluator shall verify that the Guidance instructs the administrator how to configure the TOE to generate keys for the selected key generation algorithms for all key types and uses identified in the TSS.

2.2.1.3. Tests

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

RSA Key Generation

56. FIPS PUB 186-5 Key Pair generation specifies five methods for generating the primes p and q. These are:
    a. Random provable primes
    b. Random probable primes
    c. Provable primes with conditions based on auxiliary provable primes
    d. Probable primes with conditions based on auxiliary provable primes
    e. Probable primes with conditions based on auxiliary probable primes

In addition to the key generation method, the input parameters are:

- Modulus [3072, 4096, 6144, 8192]
- Hash algorithm [SHA-384, SHA-512] (methods 1, 3, 4 only)
- Rabin-Miller prime test [2100, 2Security String] (methods 2, 4, 5 only)
- p mod 8 value [0,1,3,5,7]
- q mod 8 value [0,1,3,5,7]
- Private key format [standard, Chinese Remainder Theorem]
- Public exponent [fixed value, random]

The evaluator shall verify the ability of the TSF to correctly produce values for the RSA 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.

Testing for Random Provable Primes and Conditional Methods

57. To test the key generation method for the random provable primes method and for all the primes with conditions methods (methods 1, 3-5), the evaluator must

seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. For each supported combination of the above input parameters, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Testing for Random Probable Primes Method

58. If the TOE generates random probable primes (method 2) then, if possible, the random probable primes method should also be verified against a known good implementation as described above. If verification against a known good implementation is not possible, the evaluator shall have the TSF generate 25 key pairs for each supported key length nlen and verify that all of the following are true.

- n = p*q
- p and q are probably prime according to Miller-Rabin tests with error probability <2(-125)
- 216 < e < 2256 and e is an odd integer
- GCD(p-1,e) = 1
- GCD(q-1,e) = 1
- |p-q| > 2(nlen/2 – 100)
- p ≥ squareroot(2)*(2(nlen/2 -1))
- q ≥ squareroot(2)*(2(nlen/2 -1))
- 2(nlen/2) < d < LCM(p-1,q-1)
- e*d = 1 mod LCM(p-1,q-1)

Elliptic Curve Key Generation

59. To test the TOE's ability to generate asymmetric cryptographic keys using elliptic curves, the evaluator shall perform the ECC Key Generation Test and the ECC Key Validation Test using the following input parameters:
    a. Elliptic curve [P-256, P-384, P-521]
    b. Key pair generation method [extra random bits, rejection sampling]

ECC Key Generation Test

60. For each supported combination of the above input parameters the evaluator shall require the implementation under test to generate 10 private/public key pairs (d, Q). The private key, d, shall be generated using a random bit generator as specified in FCS_RBG.1. The private key, d, is used to compute the public key, Q'. The evaluator shall confirm that 0<d<n (where n is the order of the group), and the computed value Q' is then compared to the generated public/private key pairs' public key, Q, to confirm that Q is equal to Q'.

ECC Key Validation Test

61. For each supported combination of the above parameters the evaluator shall generate 12 private/public key pairs using the key generation function of a known-good implementation. For each set of 12 public keys, the evaluator shall modify four public key values by shifting x or y out of range by adding the order of the field and modify four other public key values by shifting x or y so that they are still in bounds, but not on the curve. The remaining public key values are left unchanged (i.e., correct). To determine correctness, the evaluator shall submit the public keys to the public key validation (PKV) function of the TOE and shall confirm that the results correspond as expected for the modified and unmodified values.

Finite Field Cryptography Key Generation

62. To test the TOE's ability to generate asymmetric cryptographic keys using finite fields, the evaluator shall perform the Safe Primes Generation Test and the Safe Primes Validation Test using the following input parameter:
    a. Fields/Groups [MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]

Safe Primes Generation Test

63. For each supported safe primes group, generate 10 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Safe Primes Validation Test

64. For each supported safe primes group, use a known good implementation to generate 10 key pairs. For each set of 10, the evaluator shall modify three such that they are incorrect. The remaining values are left unmodified (i.e. correct). To determine correctness, the evaluator shall submit the key pairs to the public key validation (PKV) function of the TOE and shall confirm that the results correspond as expected for the modified and unmodified values.

LMS Key Generation

65. To test the TOE's ability to generate asymmetric cryptographic keys using LMS, the evaluator shall perform the LMS Key Generation Test using the following input parameters:
    a. Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]
    b. Winternitz [1, 2, 4, 8]
    c. Tree height [5, 10, 15, 20, 25]

LMS Key Generation Test

66. For each supported combination of the hash algorithm, Winternitz parameter, and tree height, the evaluator shall generate one public key for each of the test cases. The number of test cases depends on the tree height.

Height	Number of test cases
5	5
10	4
15	3
20	2
25	1

The evaluator shall verify the correctness of the TSF's implementation by comparing the public key generated by the TSF with that generated by a known good implementation using the same input parameters.

ML-KEM Key Generation

67. To test the TOE's ability to generate asymmetric cryptographic keys using ML-KEM, the evaluator shall perform the Algorithm Functional Test using the following input parameters:
    - Parameter set [ML-KEM-1024]
    - Random seed d [32 bytes]
    - Random seed z [32 bytes]

Algorithm Functional Test

68. For each supported parameter set the evaluator shall require the implementation under test to generate 25 key pairs using 25 different randomly generated pairs of 32-byte seed values (d, z). To determine correctness, the evaluator shall compare the resulting key pairs (ek, dk) with those generated using a known-good implementation using the same inputs.

ML-DSA Key Generation

69. To test the TOE's ability to generate asymmetric cryptographic keys using ML-DSA, the evaluator shall perform the Algorithm Functional Test using the following input parameters:
    - Parameter set [ML-DSA-87]
    - Random seed [32 bytes]

Algorithm Functional Test

For each supported parameter set the evaluator shall require the implementation under test to generate 25 key pairs using 25 different randomly generated 32-byte seed values. To determine correctness, the evaluator shall compare the resulting key pairs with those generated using a known-good implementation using the same inputs.

XMSS Key Generation

70. To test the TOE's ability to generate asymmetric cryptographic keys using XMSS, the evaluator shall perform the XMSS Key Generation Test using the following input parameters:
    - Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]
    - Tree height [10, 16, 20] (XMSS only)

Height	Number of test cases
10	5
16	4
20	3
40	2
60	1

XMSS Key Generation Test

For each supported combination of hash algorithm and tree height, the evaluator shall generate one public key for each test case. The number of test cases depends on the tree height as specified in the table above.

The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Note: The number of test cases is limited due to the extreme amount of time it can take to generate XMSS trees.

2.2.2. FCS_CKM.6 Timing and Event of Cryptographic Key Destruction

2.2.2.1. TSS

71. The evaluator shall examine 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 shall confirm 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 shall check that this is consistent with the operation of the TOE.

72. 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).

73. 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.

74. 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 in an encrypted form or that it is destroyed by a method included under FCS_CKM.6.

75. 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.

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

2.2.2.2. Guidance Documentation

77. 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.

78. 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 the TSS and Operational Guidance).

2.2.2.3. Tests

79. None

2.2.3. FCS_CKM_EXT.7 Cryptographic Key Agreement

2.2.3.1. TSS

80. The evaluator shall ensure that the TSS documents that the security strength of the material contributed by the TOE is sufficient for the security strength of the key and the agreement method.

81. The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be as shown in the table below. The information provided in this example does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available. The relevant SFR citation may come from a functional package if the TOE boundary includes any functional packages that define uses of key establishment schemes. For example, FCS_TLSS_EXT.1 is referenced below, which would be appropriate if the TOE includes that SFR claim as part of conformance to the Functional Package for TLS.

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	Administration
ECDH	FCS_IPSEC_EXT.1	Authentication Server

2.2.3.2. Guidance Documentation

82. The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key agreement method(s).

2.2.3.3. Tests

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

FFC Diffie-Hellman Key Agreement

84. To test the TOE's implementation of FFC Diffie-Hellman Key Agreement, the evaluator shall perform the Algorithm Functional Test and Validation Test using the following input parameters:
    - Domain Parameter Group [MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]

Algorithm Functional Test

85. For each supported domain parameter group, the evaluator shall generate 10 test cases by generating the initiator and responder secret keys using random data, calculating the responder public key, and creating the shared secret. The resulting shared secrets shall be compared with those generated by a known-good implementation using the same inputs.

Validation Test

86. For each supported combination of the above parameters the evaluator shall generate 15 Diffie Hellman initiator/responder key pairs using the key generation function of a known-good implementation. For each set of key pairs, the evaluator shall modify five initiator private key values. The remaining key values are left unchanged (i.e., correct). To determine correctness, the evaluator shall confirm that the 15 shared secrets correspond as expected for both the modified and unmodified inputs.

Elliptic Curve Diffie-Hellman Key Agreement

87. To test the TOE's implementation of Elliptic Curve Diffie-Hellman Key Agreement, the evaluator shall perform the Algorithm Functional Test and Validation Test using the following input parameters:
    - Elliptic Curve [P-256, P-384, P-521]

Algorithm Functional Test

88. For each supported Elliptic Curve the evaluator shall generate 10 test cases by generating the initiator and responder secret keys using random data, calculating the responder public key, and creating the shared secret. The resulting shared secrets shall be compared with those generated by a known-good implementation using the same inputs.

Validation Test

89. For each supported Elliptic Curve the evaluator shall generate 15 Diffie Hellman initiator/responder key pairs using the key generation function of a known-good implementation. For each set of key pairs, the evaluator shall modify five initiator private key values. The remaining key values are left unchanged (i.e., correct). To determine correctness, the evaluator shall confirm that the 15 shared secrets correspond as expected for the modified and unmodified values.

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

2.2.4.1. TSS

90. The evaluator shall examine the TSS to ensure it identifies the mode or modes in which AES operates.

2.2.4.2. Guidance Documentation

91. There are no additional Guidance evaluation activities for this component.

2.2.4.3. Tests

92. There are no additional Test evaluation activities for this component.

2.2.5. FCS_COP.1/SigGen Cryptographic Operation - Signature Generation

2.2.5.1. TSS

93. The evaluator shall examine the TSS and verify that any hash function is the appropriate security strength for the signing algorithm.

94. The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed and used in accordance with the relevant standards.

95. The evaluator shall examine the TSS to verify that the TOE has appropriate measures in place to ensure that hash-based signature algorithms do not reuse private keys.

96. The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm(s) and key size(s) supported by the TOE for signature generation services.

2.2.5.2. Guidance Documentation

97. There are no additional Guidance evaluation activities for this component.

2.2.5.3. Tests

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

RSA-PKCS Signature Generation

99. To test the TOE's ability to perform RSA Digital Signature Generation using PKCS1-v1,5 signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

    - Modulus size [2048, 3072, 4096, 6144, 8192] bits
    - Hash algorithm [SHA-256, SHA-384, SHA-512]

Generated Data Test

100. For each supported combination of the above parameters, the evaluator shall cause the TOE to generate three test cases using random data. The evaluator shall compare the results against those from a known-good implementation.

RSA-PSS Signature Generation

101. To test the TOE's ability to perform RSA Digital Signature Generation using PSS signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

- Modulus size [2048, 3072, 4096, 6144, 8192] bits
- Hash algorithm [SHA-256, SHA-384, SHA-512]
- Salt length [Fixed based on implementation]
- Mask function [MGF1]

Generated Data Test

102. For each supported combination of the above parameters, the evaluator shall cause the TOE to generate three test cases using random data. The evaluator shall compare the results against those from a known-good implementation.

ECDSA Signature Generation

103. To test the TOE's ability to perform ECDSA Digital Signature Generation using extra random bits or rejection sampling for secret number generation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- Elliptic Curve [P-256, P-384, P-521]
- Hash algorithm [SHA-256, SHA-384, SHA-512]

104. To test the TOE's ability to perform ECDSA Digital Signature Generation using deterministic secret number generation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- Elliptic Curve [P-256, P-384, P-521]
- Hash algorithm [SHA-256, SHA-384, SHA-512]

Algorithm Functional Test

105. For each supported combination of the above parameters, the evaluator shall cause the TOE to generate 10 test cases using random data. The evaluator shall compare the results against those from a known-good implementation.

LMS Signature Generation

106. To test the TOE's ability to generate cryptographic digital signature using LMS, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]
- Winternitz [1, 2, 4, 8]
- Tree height [5, 10, 15, 20, 25]

Algorithm Functional Test

107. For each supported combination of the above parameters, the evaluator shall generate 10 signatures. The evaluator shall verify the correctness of the implementation by comparing values generated by the TOE with those generated by a known good implementation using the same input parameters.

ML-DSA Signature Generation

108. To test the TOE's ability to generate digital signatures using ML-DSA, the evaluator shall perform the Algorithm Functional Test using the following input parameters:
    - Parameter set [ML-DSA-87]
    - Seed [32 random bytes] (for non-deterministic signature testing), or
    - Seed [32 zero bytes] (for deterministic signature testing)
    - Message to sign [8-65535] bytes
    - Mu value (if generated externally)
    - Previously generated private key (sk)
    - Context (for external interface testing)

Algorithm Functional Test

109. For each combination of supported parameter set and capabilities, the evaluator shall require the implementation under test to generate 15 signatures pairs using 15 different randomly generated 32-byte seed values. To determine correctness, the evaluator shall compare the resulting key pairs with those generated using a known-good implementation using the same inputs.

Known Answer Test for Rejection Cases

110. For each supported parameter set, the evaluator shall cause the TOE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in the fourth row for each corresponding test case below.

The test values are defined as follows: * Seed is the seed to generate the key pair (pk, sk) * Hash of keys is computed by SHA-256(pk||sk) * Message is the message to be signed * Hash of sig is computed by SHA-256(sig)

ML-DSA-87 Test Cases for Rejection Cases

Test case 87-RC-01
Seed:
E4F5AFCF697E0EC3C1BDEB66FAA903221E803902F9C3F716E1056A63D77DC250
Hash of Keys:
61618E8DDA6998072C8EB36974E03880D741CAF0BD523356DFC161E7C9E63934
Message:
F4F1C05004D5B946F69EAFE104C4020519086ADDB9582A20FDE887D13DFC36B1
Hash of sig:
B584E38FA442FC3C81A147D4BDBF058D73C822CAF5CA4C06B0110867F60A8001
Test case 87-RC-02
Seed:
8B828D871254D6C57384A8E7025AA3F7160CAD1D2C754499DF3844426062C3DD
Hash of Keys:
BB64481317D6C0DBAD20C0C7EF11078AD54E5D574F4A07652115A95F77C655FA
Message:
0F9409C5A4930C25B83FC5B77FDB5BB49C75372DE724D9C1A77DB700CF0CF154
Hash of sig:
F86B49BE9DEB2B209BDEB4E922E5939E92D38E562C44BB09AFBD67323C345192
Test case 87-RC-03
Seed:
E693D282CACB8CE65FD4D108DA7A373F097F0AA9713550BE242AAD5BD3E2E452
Hash of Keys:
B0BEAF56713A69BD4AB2CBEE006FA5001E7B41F3AE541E05F088933AA0CC78DF

Message:
24DABB9D57ADEBD560ED65D9451C5106D437061708F849BA53F3543CDF9AA
AE0
Hash of sig:
DBF65CEFF9F96A74AAF6F3AB27B043231BE6AA04FBA2EEC987A24A00BDD6A0
8E
Test case 87-RC-04
Seed:
4002163EB8EED01A8E0919BA8C07D291341EDCAE25B02B9779A2CFFE50561AF
0
Hash of Keys:
FED1BE685C20ECB322FC40D41DEE7E0E98D0409FBF989CAE71B8AD2D58AD64
5E
Message:
EE316BB5EBED53325B4A55571C60657B53E353B51B831F4A0BBB28107EBA4BA
8
Hash of sig:
3BE9B5545FDCED92547B3409C83B3312CCB5792A8EC3A4DA63BA692C79BEF1
7C
Test case 87-RC-05
Seed:
9C7AD524F65854C27E565BCEDF8E86D650F13A40D0448F9AE10C05F10F77712
0
Hash of Keys:
0EA872CA5A4BEA94F4E8EF7ED31800727899A51059FDEE111E5CB15F0233B534
Message:
CE09831294AA96CAF684B9E667947B021C57B24C138EC7D4DA270694C82F2E0
8
Hash of sig:
3B9526CEE6587F2418BFE603ADB0F7DF0D69EBA31C9F9F005C60C993945EBD3
3
Test case 87-RC-06
Seed:
2EB7676D4A28700DA7772A7A035EB495CAA6F842352A74824EF5FD891BC38B
2A
Hash of Keys:
D5B73703A1DDC5BCB0D14AE39B193A25D6ADA6535827973181ADB0BE70435
A5B

Message:
C2B3A0AC483A5517682285C205974B2A506946448A8F7D3E1934C155EFDFE92
2
Hash of sig:
375D598704B722C8A1FEF1626FD7738A532C06329AA4217357460E3B729660F
8
Test case 87-RC-07
Seed:
E4E80CCE8B26DF1B02B99949851EE2F907FE4F0CC34790352C76D5D91634D07
3
Hash of Keys:
84B7E61684A12698400B09EA332EA3C4FBCFA47FE37FD6AE725CBC5FA8A99D3
F
Message:
89E6AB43C9CB1CC59C3986D53217A558357E62102A26F666F2B64CD1DBB7A5
36
Hash of sig:
7C4AABD163CAEF8F6EBFDA3E3EEBC0A9604675B0E991ABAFD284F1AE8BA07B
2A
Test case 87-RC-08
Seed:
5787262B803499223D4E5A8C1EE572E89F7A69B359B3F8505355B0BDEAB95E5
C
Hash of Keys:
85AE1DE605A7B479C02730BF4B7DD6D0FD8FFE5C980893CA6DAD00BD8BD1C
E68
Message:
D3230C4E061964BBFB17702432D5D36FC1EB3D1068F8CCAA84044776E3B5CC
55
Hash of sig:
D3ABE460EE2DD9595F413CFE2780A319E4E4DFD6592995298A7AB0B82A5E281
5
Test case 87-RC-09
Seed:
CE099B99330537DD153052243FC32ACAD509A126AB982410258858567D410D
79
Hash of Keys:
E04A9F15EDF8F078EB336CE624249EF2A8EDF2CDBF6A8276E9F5E92ED9B0BAE8
Message:
0035931762665F561A1B22176567E3B10FDE2441521F77030733A8E39312EEEE

Hash of sig:
3EEF413CB5EB179896ECA172D0DBFB9B251545DC561D61580BD5BBC8B6D734E1
Test case 87-RC-10
Seed:
FC8F2929878CBD81E1CCC23913F290380120C043A4A8A251AEEBF09705B8E590
Hash of Keys:
7E2ECCA86F532E8E8092FEBB6E0007F92E7909AD2BCBE2E02AB375DAC9969E5E
Message:
D3C28875D2671C0EF23BFDC8869E8ECF8868D3F0561C3134D254F7479D0CE0E5
Hash of sig:
EB69A908EDCC04320A0B61AD57E21B044465F2037698636B64229CF2DB259789

Known Answer Test for Large Number of Rejection Cases (Total Rejection Count)

111. For each supported parameter set, the evaluator shall cause the TOE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in the fourth row of the corresponding test case below.

ML-DSA-87 Test Cases for Total Rejection Count

Test case 87-LN-01
Seed:
98B6298051D92BF37293C93C97370747BF527B87B71F6C4264182F45155ADE4C
Hash of Keys:
04A135B5C9B7020332C7B16E7108E8FF7FC1EAE1C23C5FA0B5D5CED0FEEE7424
Message:
D7B0341269259083ABF3C8DC47559A19D57669B4486E0224F376DC43E577A3D8
Hash of sig:
58D72D76EC0FB65BFB9893C4479366B79DD7B8B7577E4291D13514FCC76C26DD
Test case 87-LN-02

Seed:
DFB5BDD90F58571DCA962426C623F13D046BBE814D183886AC90D143EAD725A7
Hash of Keys:
2B6AB8CFCCCC41F759CAF01932E9413F5DC6D949BC827F739866929683FB155E
Message:
21005DB2B583CC826A9684BFFD0EE00AB97E0479FE4A1D266699337540145778
Hash of sig:
C93EA34E00FFFFC3ECEA072D5FB038A83B5539CAF7B831AEDCFA785E50B3CA5E
Test case 87-LN-03
Seed:
5AD414E0DD0EF2FE685F342871875FDF06F503717A86C3B3466565ADD2096417
Hash of Keys:
BD9C2D52F3FC78DB17E682DA2E78947ECFC0898333838D60C892700B2B0DDA9F
Message:
29139C279816B25F2D6BB52C8247D163544F7BA332C3CF63359B9E23FBC56515
Hash of sig:
DB4BE2DE19FB40437BDB7E9B6578D665DB05B4E88C16907DF4546EBA9BE03AEA
Test case 87-LN-04
Seed:
484DD2F406A4D15F49A91AD5FC3BDC1D0FF253622EB68F83D6E1C870D0E89E29
Hash of Keys:
A719DC9A77C91C46295555C2353BA0CBEA513DA9A92A5C34D2E949EFF46A12D8
Message:
6AD6E959F0EA60126364FB7C95FA71133F246A9265A11B4965EE78AB0CB5AF0E
Hash of sig:
5050D7A665074EC63D9F3966C1F01A1BFB18F9E83AE0B09F838BC1E2342ED6F4
Test case 87-LN-05

Seed:
B25C1816F82D59940D5CB829BAC364AAD013C4C16415CE1CF6DCC2F15199B391
Hash of Keys:
ADBB2CD43F222640BD9FF4E61C80E63853E8DC1F759C581B7447C9C166EAA38E
Message:
824E47322895BFFE37B6B4AFC41CF6115C07EEC0C24EB81076C87A1B01AE8617
Hash of sig:
667ADA46073BC69D64DC47BB9A76DD0D78302E7415D87D5E816B05FB95F9E84D
Test case 87-LN-06
Seed:
B2CE72B3560AF07E06465881F56ADA00262BA708D87B73F39E04E310F3B8A3E9
Hash of Keys:
FD9C4AC53AE803242A62DF933B8E8BAD6CE5207AC4A73683B6D9383B5E70B17A
Message:
A1501CC84C917E0D2D7C27C2AC382220BD8FFFE807DB38E37A9E429EC2781911
Hash of sig:
779553B195E11558EE59EF3942F5F6B446A2144600D1F4F50B300C6C56504760
Test case 87-LN-07
Seed:
AB01D0E591B7DDCD3C03395AED808FA2763C0A486D44119D621BE0FD0B022B25
Hash of Keys:
93B6ADE34F78A4ADB36B2F6D2C51DB793E659E1243E80488AE1C03B65125D6D7
Message:
8DE8122D89D15FE84A4C34F6B59B2C4B11F33B6A053154D199B634F557FDF5F6
Hash of sig:
0483045999A79B583F403DB96A736F0F0B24E2DFBC4E5CFA9B50E3D910786F07
Test case 87-LN-08
Seed:
15D60D3693762F82C9AC1DCB0576936651AC81D863842EDB91109C8EE83AE705

Hash of Keys:
2DF544E2E939AA717741C2437288FAEB308DEB8FF37A2652FAE34BAE8B84D779
Message:
F05946A6113905C34163AEF2246FD69016CE24A7BA40F8E7E42EDAC2D0A44605
Hash of sig:
F8383917AF79C8E540D2356AB05F08B465BF32DFEC444B787CE31BF48CC6C3DD
Test case 87-LN-09
Seed:
21212285BED53B3411705DAF5F3BDDB6F0618EB571B36EE11A74053407A269F5
Hash of Keys:
737061155A9A03F11F9FEBBB940BED4DD54542C4A6212F89A5EB4EC2BE542782
Message:
FFE38246BF3DEFD9CAD15CC17CEA511C067D582E04227B479E32F9197CF91482
Hash of sig:
C4C12C58032052FB2D21F0C6A7388A63154FB85B74287D2859DE6C1C6F7F277B
Test case 87-LN-10
Seed:
A2744470587C71BA43EC26DC390CE3531978F315993C653E5D3EFD2849D5D9F1
Hash of Keys:
B1BF37BFFB11531B6ADD697870D7DB2E2462D0A97A63F09C1D0038457C6D795A
Message:
9831A830231A160B9847203341A5F30BF3E87A2A482AEEA6886315C92B5C4E4C
Hash of sig:
46C669D2FEB643A38E54FF87B790CC33F44043A1B6B31DB9474D301328CA2A7F

XMSS Signature Generation

112. To test the TOE's ability to generate digital signatures using XMSS, the evaluator shall perform the XMSS Key Generation Test using the following input parameters:

- Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]
- Tree height [10, 16, 20]

XMSS Key Generation Test

113. For each supported combination of the above parameters, the evaluator shall generate 10 signatures. The evaluator shall verify the correctness of the implementation by comparing values generated by the TOE with those generated by a known-good implementation using the same input parameters.

2.2.6. FCS_COP.1/SigVer Cryptographic Operation - Signature Verification

2.2.6.1. TSS

114. The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed and used in accordance with the relevant standards.

2.2.6.2. Guidance Documentation

115. There are no additional Guidance evaluation activities for this component.

2.2.6.3. Tests

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

RSA-PKCS Signature Verification

117. To test the TOE's ability to perform RSA Digital Signature Verification using PKCS1-v1,5 signature type, the evaluator shall perform the Generated Data Test using the following input parameters:
    - Modulus size [2048, 3072, 4096, 6144, 8192] bits
    - Hash algorithm [SHA-256, SHA-384, SHA-512]

Generated Data Test

118. For each supported combination of the above parameters, the evaluator shall cause the TOE to generate six test cases using a random message and its signature such that the test cases are modified as follows:
    - One test case is left unmodified
    - For one test case the Message is modified
    - For one test case the Signature is modified
    - For one test case the exponent (e) is modified
    - For one test case the IR is moved
    - For one test case the Trailer is moved
119. The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

RSA-PSS Signature Verification

120. To test the TOE's ability to perform RSA Digital Signature Verification using PSS signature type, the evaluator shall perform the Generated Data Test using the following input parameters:
    - Modulus size [2048, 3072, 4096, 6144, 8192] bits
    - Hash algorithm [SHA-256, SHA-384, SHA-512]
    - Salt length [Fixed based on implementation]
    - Mask function [MGF1]

Generated Data Test

121. For each supported combination of the above parameters, the evaluator shall cause the TOE to generate six test cases using random data such that the test cases are modified as follows:
    - One test case is left unmodified
    - For one test case the Message is modified
    - For one test case the Signature is modified
    - For one test case the exponent (e) is modified
    - For one test case the IR is moved
    - For one test case the Trailer is moved

122. The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

ECDSA Signature Verification

123. To test the TOE's ability to perform ECDSA Digital Signature Verification, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- Elliptic Curve [P-256, P-384, P-521]
- Hash algorithm [SHA-256, SHA-384, SHA-512]

Algorithm Functional Test

124. For each supported combination of the above parameters, the evaluator shall cause the TOE to generate test cases consisting of messages and signatures such that the 21 test cases are modified as follows:

- Three test cases are left unmodified
- For three test cases the Message is modified
- For three test cases the key is modified
- For three test cases the r value is modified
- For three test cases the s value is modified
- For three test cases the value r is zeroed
- For three test cases the value s is zeroed

125. The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

LMS Signature Verification

126. To test the TOE's ability to verify cryptographic digital signature using LMS, the evaluator shall perform the Algorithm Functional Test using the following input parameters

- Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]
- Winternitz[1, 2, 4, 8]
- Tree height [5, 10, 15, 20, 25]

Algorithm Functional Test

127. For each supported combination of the above parameters, the evaluator shall generate 4 test cases consisting of signed messages and keys, such that

- One test case is unmodified (i.e. correct)
- For one test case modify the message, i.e. the message is different
- For one test case modify the signature, i.e. signature is different
- For one test case modify the signature header so that it is a valid header for a different LMS parameter set.

128. The TOE must correctly verify the unmodified test case and fail to verify the modified test cases.

XMSS Signature Verification

129. To test the TOE's ability to verify digital signatures using XMSS or XMSS MT, the evaluator shall perform the XMSS digital signature verification test using the following input parameters:

- Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]
- Tree height [10, 16, 20]

To test the TOE's ability to perform RSA Digital Signature Verification using PSS signature typ

130. For each supported combination of the above parameters, the evaluator shall generate four test cases consisting of signed messages and keys, such that

- One test case is unmodified (i.e. correct)
- For one test case modify the message, i.e. the message is different
- For one test case modify the signature, i.e. signature is different
- For one test case modify the signature header so that it is a valid header for a different XMSS parameter set

131. The evaluator shall verify the correctness of the implementation by verifying that the TOE correctly verifies the unmodified test case and fails to verify the modified test cases

ML-DSA Signature Verification

132. To test the TOE's ability to validate digital signatures using ML-DSA, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- Parameter set [ML-DSA-87]
- Previously generated signed Message [8-65535] bytes
- Mu value (if generated externally)
- Context (for external interface testing)
- Previously generated Public key (pk)
- Previously generated Signature

Algorithm Functional Test

133. For each combination of supported parameter set and capabilities, the evaluator shall require the implementation under test to validate 15 signatures. Each group of 15 test cases is modified as follows:
    - Three test cases are left unmodified
    - For three test cases the Signed message is modified
    - For three test cases the component of the signature that commits the signer to the message is modified
    - For three test cases the component of the signature that allows the verifier to construct the vector z is modified
    - For three test cases the component of the signature that allows the verifier to construct the hint array is modified
134. The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

2.2.7. FCS_COP.1/Hash Cryptographic Operation - Hashing

2.2.7.1. TSS

135. 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.

2.2.7.2. Guidance Documentation

136. The evaluator shall check the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

2.2.7.3. Tests

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

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

SHA-256, SHA-384, SHA-512

139. To test the TOE's ability to generate hash digests using SHA2 the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Large Data Test for each claimed SHA2 algorithm.

Algorithm Functional Test

140. The evaluator shall generate a number of test cases equal to the block size of the hash (512 for SHA2-256; 1024 for the other SHA2 algorithms).

141. Each test case is to consist of random data of a random length between 0 and 65536 bits, or the largest size supported.

142. Each test case is to consist of random data of a random length between 0 and 65536 bits, or the largest size supported.

Monte Carlo Test

143. Monte Carlo tests begin with a single seed and run 100 iterations of the chained computation.

144. There are two versions of the Monte Carlo test for SHA-1 and SHA-2. Either one is acceptable. For the Standard Monte Carlo test the message hashed is always three times the length of the initial seed.

```
For j = 0 to 99
        A = B = C = SEED
        For i = 0 to 999
                MSG = A || B || C
                MD = SHA(MSG)
                A = B
                B = C
```

```
            C = MD
        Output MD
        SEED = MD
```

145. For the alternate version of the Monte Carlo Test, the hashed message is always the same length as the seed.

```
INITIAL_SEED_LENGTH = LEN(SEED)
For j = 0 to 99
        A = B = C = SEED
        For i = 0 to 999
                MSG = A || B || C
                if LEN(MSG) >= INITIAL_SEED_LENGTH:
                        MSG = leftmost INITIAL_SEED_LENGTH bits of MSG
                else:
                        MSG = MSG || INITIAL_SEED_LENGTH - LEN(MSG) 0
bits
                MD = SHA(MSG)
                A = B
                B = C
                C = MD
        Output MD
        SEED = MD
```

146. The evaluator shall compare the output against results generated by a known-good implementation with the same input

Large Data Test

147. The implementation must be tested against one test case each on large data messages of 1GB, 2GB, 4GB, and 8GB of data as supported. The data need not be random. It may, for example, consist of a repeated pattern of 64 bits.

148. The evaluator shall compare the output against results generated by a known-good implementation with the same input.

SHA3-384, SHA3-512

149. To test the TOE's ability to generate hash digests using SHA3 the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Large Data Tests for each claimed SHA3 algorithm.

Algorithm Functional Test

150. Generate a test case consisting of random data for every message length from 0 bits (or the smallest supported message size) to rate bits, where rate equals

- 832 for SHA3-384 and
- 576 for SHA3-512.

151. Additionally, generate tests cases of random data for messages of every multiple of (rate+1) bits starting at length rate, and continuing until 65535 is exceeded.

152. The evaluator shall compare the output against results generated by a known-good implementation with the same input.

Monte Carlo Test

153. Monte Carlo tests begin with a single seed and run 100 iterations of the chained computation.

154. For this Monte Carlo Test, the hashed message is always the same length as the seed.

```
MD[0] = SEED
INITIAL_SEED_LENGTH = LEN(SEED)
For 100 iterations
        For i = 1 to 1000
                MSG = MD[i-1];
                if LEN(MSG) >= INITIAL_SEED_LENGTH:
                        MSG = leftmost INITIAL_SEED_LENGTH bits of MSG
                else:
                        MSG = MSG || INITIAL_SEED_LENGTH - LEN(MSG) 0
bits
                MD[i] = SHA3(MSG)
        MD[0] = MD[1000]
        Output MD[0]
```

155. The evaluator shall compare the output against results generated by a known-good implementation with the same input.

Large Data Test

156. The implementation must be tested against one test case each on large data messages of 1GB, 2GB, 4GB, and 8GB of data as supported. The data need not be random. It may, for example, consist of a repeated pattern of 64 bits.

157. The evaluator shall compare the output against results generated by a known-good implementation with the same input.

2.2.8. FCS_COP.1/KeyedHash Cryptographic Operation - Keyed Hash

2.2.8.1. TSS

158. The evaluator shall examine the TSS to ensure that the size of the key is sufficient for the desired security strength of the output.

2.2.8.2. Guidance Documentation

159. There are no additional Guidance evaluation activities for this component.

2.2.8.3. Tests

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

HMAC

161. To test the TOE's ability to generate keyed hashes using HMAC the evaluator shall perform the Algorithm Functional Test for each combination of claimed HMAC algorithm the following parameters:

- Hash function [SHA-256, SHA-384, SHA-512]
- Key length [8-65536] bits by 8s
- MAC length [32-[digest size of hash function (256, 384, 512)]] bits

Algorithm Functional Test

162. For each supported Hash function the evaluator shall generate 150 test cases using random input messages of 128 bits, random supported key lengths, random keys, and random supported MAC lengths such that across the 150 test cases:

- The key length includes the minimum, the maximum, a key length equal to the block size, and key lengths that are both larger and smaller than the block size.

- The MAC size includes the minimum, the maximum, and two other random values.

163. The evaluator shall compare the output against results generated by a known-good implementation with the same input.

2.2.9. FCS_RBG.1 Random Bit Generation (RBG)

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

2.2.9.1. TSS

FCS_RBG.1.1

165. The evaluator shall examine the TSS to determine it identifies the DRBGs used by the TOE.

FCS_RBG.1.2

166. There are no additional TSS evaluation activities for this element.

FCS_RBG.1.3

167. The evaluator shall verify that the TSS identifies how the DRBG state is updated, and the situations under which this may occur.

2.2.9.2. Guidance Documentation

FCS_RBG.1.1

168. If the DRBG functionality is configurable, the evaluator shall verify that the operational guidance includes instructions on how to configure this behaviour.

FCS_RBG.1.2

169. There are no additional Guidance evaluation activities for this element.

FCS_RBG.1.3

170. If the ST claims that the DRBG state can be updated on demand, the evaluator shall verify that the operational guidance has instructions for how to perform this operation.

2.2.9.3. Tests

FCS_RBG.1.1

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

172. 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 shall verify 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).

173. 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 shall verify 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.

174. 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.

FCS_RBG.1.2

175. There are no test activities for this element.

FCS_RBG.1.3

176. There are no test activities for this element.

2.3. Identification and Authentication (FIA)

2.3.1. FIA_UIA_EXT.1 User Identification and Authentication

2.3.1.1. TSS

177. The evaluator shall examine the TSS to determine that it describes the logon process for remote authentication mechanism (e.g., SSH public key, Web GUI password, etc.) and optional local authentication mechanisms supported by the TOE. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a "successful logon".

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

179. For distributed TOEs, the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no unauthorised access to any TOE component can occur.

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

2.3.1.2. Guidance Documentation

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

2.3.1.3. Tests

182. 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 all combinations of supported credentials and login methods, 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, the evaluator shall test that the components authenticate Security Administrators as described in the TSS.

2.4. Security management (FMT)

2.4.1. General Requirements for Distributed TOEs

For distributed TOEs, the evaluation activities defined in this chapter shall be performed.

2.4.1.1. TSS

183. For distributed TOEs, the evaluator shall verify that the TSS 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

184. For distributed TOEs, the evaluator shall verify that the Guidance Documentation describes 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

185. 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 Management of security functions behaviour

2.4.2.1. TSS

186. For distributed TOEs, see Section 2.4.1.1. There are no specific requirements for non-distributed TOEs.

2.4.2.2. Guidance Documentation

187. 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).

188. For distributed TOEs, the guidance documentation shall describe all steps for how to update all TOE components. This shall contain a 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

189. The evaluator shall perform the following tests:

a. Test 1: 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.

b. Test 2: 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 is covered by Test 1 for FPT_TUD_EXT.1.

2.4.3. FMT_MTD.1/CoreData Management of TSF Data

2.4.3.1. TSS

190. For each administrative function identified in the guidance documentation that is accessible through an interface prior to administrator log-in, the evaluator shall confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

191. 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

192. 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.

193. 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.

2.4.3.3. Tests

194. No separate testing for FMT_MTD.1/CoreData is required unless one of the management functions has not already been exercised under any other SFR.

2.4.4. FMT_SMF.1 Specification of Management Functions

195. 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, FPT_TUD_EXT.1.2 and FPT_TUD_EXT.2.2 (if included in the ST and if an administrator-configurable action is included as per Application Note 65 of FPT_TUD_EXT.2.4 in the cPP), 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. If the TOE claims conformance to the Functional Package for X.509, any management functions defined there are relevant to this SFR as well. Compliance to these requirements satisfies compliance with FMT_SMF.1.

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

196. The evaluator shall examine the TSS and Guidance Documentation and confirm that each management function specified in FMT_SMF.1 is adequately described. The evaluator shall confirm that the TSS details which security management functions are available through the local and/or remote administration interfaces.

197. [Conditional] The evaluator shall examine the TSS and Guidance Documentation to verify they both describe the local administrative interface. The evaluator shall ensure the Guidance Documentation includes appropriate warnings for the administrator to ensure the interface is local.

198. 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.

199. (If 'configure local audit' is selected) The evaluator shall examine the TSS and Guidance Documentation to ensure that a description of the logging implementation is described in enough detail to determine how log files are maintained on the TOE.

2.4.4.2. Guidance Documentation

200. See Section 2.4.4.1.

2.4.4.3. Tests

201. The evaluator shall test 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. TSS

202. The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE (e.g., if local administrators and remote administrators have different privileges, or if several types of administrators with different privileges are supported by the TOE).

2.4.5.2. Guidance Documentation

203. 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.

2.4.5.3. Tests

204. 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.

205. For example, if the following are possible:

- direct connection if the TOE can be administered through a local hardware interface
- SSH if the TSF shall be validated against the Functional Package for Secure Shell referenced in Section 2.2 of the cPP
- TLS/HTTPS if the TSF has TLS implemented as defined in the Functional Package for TLS

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 Symmetric Keys)

2.5.1.1. TSS

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

2.5.1.2. Guidance Documentation

207. None

2.5.1.3. Tests

208. None

2.5.2. FPT_STM_EXT.1 Reliable Time Stamps

2.5.2.1. TSS

209. 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.

210. If 'obtain time from the underlying virtualization system' is selected, the evaluator shall examine the TSS to ensure that it identifies the Virtualization System (VS) interface the TOE uses to obtain time. If there is a delay between updates to the time on the VS and updating the time on the TOE, the TSS shall identify the maximum possible delay.

2.5.2.2. Guidance Documentation

211. The evaluator shall examine 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.

212. If the TOE supports obtaining time from the underlying VS, then the evaluator shall verify that the Guidance Documentation specifies any configuration steps necessary. If no configuration is necessary, then no statement is necessary in the Guidance Documentation. If there is a delay between updates to the time on the VS and updating the time on the TOE, then the evaluator shall ensure the Guidance Documentation informs the administrator of the maximum possible delay.

2.5.2.3. Tests

213. 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 shall use the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

b. Test 2: If the TOE supports the use of an NTP server, then 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 shall 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, then the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

c. Test 3 [conditional]: If the TOE obtains time from the underlying VS, then the evaluator shall record the time on the TOE, modify the time on the underlying VS, and verify the modified time is reflected by the TOE. If there is a delay between setting the time on the VS and when the time is reflected on the TOE, then the evaluator shall ensure this delay is consistent with the TSS and Guidance.

214. 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.

2.5.3. FPT_TST_EXT.1 TSF Testing

2.5.3.1. TSS

215. The evaluator shall examine the TSS to ensure that it details each of the self-tests that are identified by the SFR; 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. If more than one failure response is listed in FPT_TST_EXT.1.2, then the evaluator shall examine the TSS to ensure it clarifies which response is associated with which type of failure.

216. 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. The evaluator shall also examine the TSS to ensure it describes how the TOE reacts if one or more TOE components fail self-testing (e.g., halting and displaying an error message; failover behaviour).

2.5.3.2. Guidance Documentation

217. 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.

218. 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

219. 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.

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

    a. [FIPS 140-2], Section 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], Section 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.

c. [FIPS 140-3] ([ISO/IEC 19790:2015]), Section 7.10.2.2, 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.

d. [ISO/IEC 19790:2025], Section 7.10.3.2, Pre-operational software/firmware integrity test. Verification using an approved integrity technique. Note that the testing is not restricted to the cryptographic functions of the TOE.

221. The evaluator shall verify that the self-tests described above are carried out according to the SFR and in agreement with the descriptions in the TSS.

222. 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-tests are performed by which component.

2.5.4. FPT_TUD_EXT.1 Trusted Update

2.5.4.1. TSS

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

224. 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 the method used to authenticate the update, including either digital-signature verification or X.509 certificate-based verification when selected in FPT_TUD_EXT.1. The evaluator shall verify that the TSS describes how candidate updates are obtained, the processing performed to authenticate the update, and the actions taken for both successful and unsuccessful authentication.

225. If the options ‘support automatic checking for updates’ or ‘support automatic updates’ are chosen from the selections 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.

226. 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 is performed for each TOE component.

2.5.4.2. Guidance Documentation

227. 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 shall describe how to query the loaded but inactive version.

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

229. 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 Security Administrator; it does not need to give information about the internal communication (amongst TOE components) that takes place when applying updates.

230. 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 is performed for each TOE component.

231. 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 shall also ensure that the Guidance Documentation describes how the certificates are installed/updated/selected, if necessary.

2.5.4.3. Tests

232. The evaluator shall perform the following tests:

a. Test 1: The evaluator shall perform 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 will obtain a legitimate update using procedures described in the guidance documentation and verify that it has been 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 shall verify 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 shall perform 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 the TOE itself verifies a digital signature in order to authorise the installation of an image to update the TOE, the following test shall be performed (otherwise the test shall be omitted). The evaluator shall first confirm that no updates are pending and then perform 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 shall obtain or produce illegitimate updates as defined below and attempt to install them on the TOE. The evaluator shall verify that the TOE rejects all of the illegitimate updates. The evaluator shall perform this test using all of the following forms of illegitimate updates:

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

    ii. An image that has not been signed

    iii. An otherwise valid image with a properly formed signature that was signed by an unknown key. The purpose of this test is to verify the TOE would only accept images signed by an explicitly trusted key (or a key associated with a trusted certificate).

    iv. 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

the current version and most recently installed version reflect the same version information as prior to the update attempt.

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

234. For distributed TOEs, the evaluator shall perform Test 1 and Test 2 for all TOE components.

2.6. TOE Access (FTA)

2.6.1. FTA_SSL.3 TSF-Initiated Termination

2.6.1.1. TSS

235. The evaluator shall examine the TSS to determine that it details the administrative remote session termination and the related inactivity time period.

2.6.1.2. Guidance Documentation

236. The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

2.6.1.3. Tests

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

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

2.6.2. FTA_SSL.4 User-Initiated Termination

2.6.2.1. TSS

238. The evaluator shall examine the TSS to determine that it details how the remote administrative session (and if applicable the local administrative session) are terminated.

2.6.2.2. Guidance Documentation

239. The evaluator shall confirm that the guidance documentation states how to terminate a remote interactive session (and if applicable the local administrative session).

2.6.2.3. Tests

240. The evaluator shall perform the following tests:

a. Test 1 [conditional]: If the TOE supports local administration, the evaluator shall initiate an interactive local session with the TOE. The evaluator shall then follow the guidance documentation to exit or log off the session and observe that the session has been terminated.

b. Test 2: For each method of remote administration, the evaluator shall initiate an interactive remote session with the TOE. The evaluator shall then follow the guidance documentation to exit or log off the session and observe that the session has been terminated.

2.6.3. FTA_TAB.1 Default TOE Access Banners

2.6.3.1. TSS

241. The evaluator shall check the TSS to ensure that it details each administrative method of access (local and/or 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 may be different for different administrative methods of access and may be configured during initial configuration (e.g., via a configuration file).

2.6.3.2. Guidance Documentation

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

2.6.3.3. Tests

243. The evaluator shall also perform the following test:

a. Test 1: The evaluator shall follow the guidance documentation to configure a notice and consent warning message. The evaluator shall

then, for each method of human interactive 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

244. The evaluator shall examine the TSS to determine that, for all communications with authorised 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 SFRs listed in the ST.

2.7.1.2. Guidance Documentation

245. The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorised IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

2.7.1.3. Tests

246. The developer 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.

247. The evaluator shall perform the following tests:

    a. Test 1: The evaluator shall ensure that communications using each protocol with each authorised IT entity is tested during the course of the evaluation, setting up each connection 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 the communication channel can in fact be initiated from the TOE.

c. Test 3: The evaluator shall verify that, for each communication channel with an authorised 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 using a secure communication mechanism with a distinct IT entity, interrupt the connection of that IT entity for: 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 network link layer.

   The evaluator shall ensure that, when the 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 TOE external interruption (such as a cable being physically removed or a virtual connection being disabled), another network device shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall be external to the TOE (i.e., by manipulating the test environment and not by TOE configuration change).

248. Further assurance activities are associated with the specific protocols.

249. 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.

250. The developer 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

251. The evaluator shall examine the TSS to determine that the remote TOE administration methods 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

252. 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

253. 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 are tested during the course of the evaluation, setting up the connections as described in the guidance documentation and verifying that communication is successful.

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

254. Further assurance activities are associated with the specific protocols.

255. 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.2 Protected audit data storage

3.1.1.1. TSS

256. The evaluator shall examine the TSS to ensure it describes the amount of audit data stored locally and how it is protected against unauthorised modification or deletion. The evaluator shall verify that the TSS describes the conditions that must be met for authorised deletion of audit records.

257. 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 either provides 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

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

3.1.1.3. Tests

259. The evaluator shall perform the following tests:

a. Test 1: The evaluator shall attempt to access the audit trail without authentication as a 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.

   In the case that no other users than the Security Administrator can be defined, without user authentication the user may not be able to get to the point where the attempt to access the audit trail can be executed. In this 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 authenticated Security Administrator and attempt to delete the audit records (if supported by the TOE, and to the extent described in the TSS). The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorised for deletion are deleted.

260. 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 Counting Lost Audit Data

261. This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.4 and FAU_STG_EXT.1.5.

3.1.2.1. TSS

262. 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. when the local storage for audit data is full.

263. 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 and not all. This might lead to the situation where all TOE components store their audit information themselves but FAU_STG_EXT.2 is supported only by one of the components.

3.1.2.2. Guidance Documentation

264. 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.

265. 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

266. The evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2 are correct when performing the tests for FAU_STG_EXT.1.5.

267. 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_EXT.3 Action in Case of Possible Audit Data Loss

268. This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.4 and FAU_STG_EXT.1.5.

3.1.3.1. TSS

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

270. 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 implements this SFR. Since this SFR is optional, it might only apply to some TOE components and not all. This might lead to the situation where all TOE components store their audit information themselves but FAU_STG_EXT.3 is supported only by one of the components. In particular, the evaluator shall 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 shall verify that the TSS makes clear any situations in which audit records might be 'invisibly lost'.

3.1.3.2. Guidance Documentation

271. The evaluator shall also verify that the guidance documentation describes how the Security Administrator 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 very likely that it will be stored in the log files). The description in the guidance documentation shall correspond to the description in the TSS.

3.1.3.3. Tests

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

273. 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.

3.1.4. FCS_CKM.2 Cryptographic Key Distribution

3.1.4.1. TSS

274. The evaluator shall ensure that the TSS documents that the security strength supported by the selected key distribution methods is sufficient for the security strength of the keys distributed through those methods.

275. It is not necessary to identify the services that use each key distribution method here. That information should be documented in the requirements for the individual services and protocols that invoke key distribution.

3.1.4.2. Guidance Documentation

276. The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key distribution methods.

3.1.4.3. Tests

277. Specific testing for this component is covered by testing for the claimed components in FCS_COP.1/KeyEncap, FCS_COP.1/KeyWrap, and the applicable key establishment or key derivation mechanisms claimed under FCS_CKM.1/AKG, FCS_CKM.5, FCS_CKM_EXT.8, or FCS_CKM_EXT.3.

3.2. Protection of the TSF (FPT)

3.2.1. FPT_ITT.1 Basic Internal TSF Data Transfer Protection

278. 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.

3.2.1.1. TSS

279. 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 inter-component communication. 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.

3.2.1.2. Guidance Documentation

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

3.2.1.3. Tests

281. The evaluator shall perform the following tests:

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

b. Test 2: The evaluator shall verify that for each communication channel established between distributed TOE components, the channel data is not sent in plaintext.

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

The evaluator shall ensure that, for each different pair of non-equivalent TOE 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 network link layer.

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

For a non-virtualized TOE, the interruption shall not be performed at the virtual node (e.g., virtual switch) and must be physical in nature.

282. Further assurance activities are associated with the specific protocols identified in the ST.

3.3. Trusted Path/Channels (FTP)

3.3.1. FTP_TRP.1/Join Trusted Path

3.3.1.1. TSS

283. The evaluator shall examine the TSS to determine that the methods of joining components to the TOE are identified, along with how communications with those newly-joined components 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.

284. 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).

285. 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.

3.3.1.2. Guidance Documentation

286. 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 clarifies which TOE 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.

287. In the case of a distributed TOE that relies on the operational environment to provide security for some aspects of the registration channel security, 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 TOE components to the distributed 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 shall consult NIST 800-57 Table 2 for further guidance on comparable algorithm strength.

3.3.1.3. Tests

288. 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 verifying 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 TOE components from the Internet during registration might be inconsistent with the need for a TOE 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 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 TOE 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 the communication channel can be enabled by a Security Administrator for all the TOE components identified in the guidance documentation as capable of having the Security Administrator initiate the channel.

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.

289. Further assurance activities are associated with the specific protocols.

3.4. Communication (FCO)

3.4.1. FCO_CPC_EXT.1 Component Registration Channel Definition

290. 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 TOE component from successfully communicating with other TOE components (in a way that enables it to participate as part of the TOE) before it has been properly authenticated and becomes a functional part of the distributed 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).

i. What stops anybody other than a Security Administrator from carrying out this step?

ii. How does the Security Administrator know that they are enabling the intended TOE 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 TOE component from successfully joining if the Security Administrator has not carried out the enablement step; or, equivalently, how does the distributed TOE ensure that an action by an authentic Security Administrator is required before a component can successfully join?

d. What stops a TOE 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 does the registration process, and its resulting secure

channel, ensure that the transmitted data is protected from disclosure and that it 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 TOE component from successfully communicating with other TOE components if the Security Administrator has carried out the disablement step?

3.4.1.1. TSS

291. (Note: Section 3.4.1 lists questions for which the evaluator shall determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

292. 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 type of channel being used
- Second type: the TSS (with support from the operational guidance if selected in FTP_TRP.1.3/Join) provides 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.

293. The evaluator shall verify 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. If the

registration channel is also to be used for other communications then FTP_TRP.1/Join cannot be present in the ST.

3.4.1.2. Guidance Documentation

294. (Note: Section 3.4.1 lists questions for which the evaluator shall determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

295. 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 will be prevented from communicating with the component that is being removed from the TOE (preventing the remaining components from either attempting to initiate communications with the disabled component, or from responding to communications from the disabled component).

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

297. 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 that they 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 encryption key of equivalent length)

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

298. 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).

299. 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.

3.4.1.3. Tests

300. (Note: Section 3.4.1 lists questions for which the evaluator shall determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

301. The evaluator shall perform the following tests:

a. Test 1a: the evaluator shall confirm that a TOE component that is not currently a member of the distributed TOE cannot communicate with any other component of the TOE until the non-member 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).

b. Test 1b: the evaluator shall confirm that after enablement, a TOE component can only communicate with the other TOE components that it has been enabled for. This includes testing that the enabled communication is successful for the enabled TOE component pair, and that communication remains unsuccessful with any other TOE 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 separate registered-components channel must not allow communications until after the enablement step has been completed.

302. The evaluator shall repeat Tests 1a and 1b for each different type of enablement process that can be used in the distributed 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 perform 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.

    i. If the ST uses the first type of communication channel in the selection list in FCO_CPC_EXT.1.2 then the evaluator shall test the channel via the Evaluation Activities for FTP_ITC.1 or FPT_ITT.1 according to the second selection list in FCO_CPC_EXT.1.2 (which is contained within the first type), therefore the evaluator shall ensure that the test coverage for these SFRs includes their usage in the registration process.

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

    iii. 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:

    i. 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.

    ii. 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 (see AGD_PRE.1 refinement item 2 in (see the requirements on Preparative Procedures in 3.4.1.2), 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_SAR.1 Audit Review

4.1.1.1. TSS

303. There are no TSS evaluation activities for this component.

4.1.1.2. Guidance Documentation

304. The evaluator shall review the AGD for the procedure on how to review the audit records.

4.1.1.3. Tests

305. The evaluator shall verify that the audit records provide all of the information specified in FAU_GEN.1 and that this information is suitable for human interpretation. The evaluation activity for this requirement is performed in conjunction with the evaluation activity for FAU_GEN.1.

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

306. 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.

4.1.3. FAU_STG_EXT.4 Protected Local Audit Event Storage for Distributed TOEs and FAU_STG_EXT.5 Protected Remote Audit Event Storage for Distributed TOEs

4.1.3.1. TSS

307. The evaluator shall examine 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.

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

4.1.3.2. Guidance Documentation

309. 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.

310. 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.

4.1.3.3. Tests

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

a. 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 an appropriate interface), or indirectly after transmission to a central audit log storage location.

b. 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 to the external audit server. 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 shall induce audit record transmission, then review the packet capture around the time of transmission and verify that no audit data has been transmitted in the clear.

c. 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 have been 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 shall induce audit record transmission, then review the packet capture around the time of transmission and verify that no audit data is transmitted in the clear.

312. 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_COP.1/AEAD Cryptographic Operation – Authenticated Encryption with Associated Data

4.2.1.1. TSS

313. The evaluator shall examine the TSS to ensure that it describes the construction of any IVs, nonces, and tags in conformance with the relevant specifications.

314. If a CCM mode algorithm is selected, then the evaluator shall examine the TOE summary specification to confirm that it describes how the nonce is generated and that the same nonce is never reused to encrypt different plaintext pairs under the same key.

315. If a GCM mode algorithm is selected, then the evaluator shall examine the TOE summary specification to confirm that it describes how the IV is generated and that the same IV is never reused to encrypt different plaintext pairs under the same key. The evaluator shall also confirm that for each invocation of GCM, the length of the plaintext is at most (2^{32})-2 blocks

4.2.1.2. Guidance Documentation

316. There are no additional Guidance evaluation activities for this component.

4.2.1.3. Tests

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

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

AES-CCM

319. To test the TOE's implementation of AES-CCM authenticated encryption functionality the evaluator shall perform the Algorithm Functional Tests described below using the following input parameters:

- Key Size [128, 256] bits
- Associated data size [0-65536] bits in increments of 8
- Payload size [0-256] bits in increments of 8
- IV/Nonce size [64-104] bits in increments of 8
- Tag size [32-128] bits in increments of 16

Algorithm Functional Test

320. Unless otherwise specified, the following tests should use random data, a tag size of 128 bits, IV/Nonce size of 104 bits, payload size of 256 bits, and associated data size of 256 bits. If any of these values are not supported, any supported value may be used. The evaluator shall compare the output from each test case against results generated by a known-good implementation with the same input parameters.

Variable Associated Data Test

321. For each claimed key size, and for each supported associated data size from 0 through 256 bits in increments of 8 bits, the TOE must be tested by encrypting 10 test cases using all random data. In addition, for each key size, the TOE must

be tested by encrypting 10 cases with associated data lengths of 65536 bits, if supported.

Variable Payload Test

322. For each claimed key size, and for each supported payload size from 0 through 256 bits in increments of 8 bits, the TOE must be tested by encrypting 10 test cases using all random data.

Variable Tag Test

323. For each claimed key size, and for each supported tag size from 32 through 128 bits in increments of 16 bits, the TOE must be tested by encrypting 10 test cases using all random data.

Decryption Verification Test

324. For each claimed key size, for each supported associated data size from 0 through 256 bits in increments of 8 bits, for each supported payload size from 0 through 256 bits in increments of 8 bits, for each supported IV/Nonce size from 64 through 104 bits in increments of 8 bits, and for each supported tag size from 32 through 128 bits in increments of 16 bits, the TOE must be tested by decrypting 10 test cases using all random data.

AES-GCM

325. To test the TOE's implementation of AES-GCM authenticated encryption functionality the evaluator shall perform the Encryption Algorithm Functional Tests and Decryption Algorithm Functional Tests as described below using the following input parameters:

- Key Size [128, 256] bits
- Associated data size [0-65536] bits
- Payload size [0-65536] bits
- IV size [96] bits
- Tag size [96, 104, 112, 120, 128] bits

Encryption Algorithm Functional Tests

326. The evaluator shall generate 15 test cases using random data for each combination of the above parameters as follows:

- Each claimed key size,

- Each supported tag size,
- Four supported non-zero payload sizes, such that two are multiples of 128 bits and two are not multiples of 128 bits,
- Four supported non-zero associated data sizes, such that two are multiples of 128 bits and two are not multiples of 128 bits, and
- An associated data size of zero, if supported.

327. Note that the IV size is always 96 bits.

328. The evaluator shall compare the output from each test case against results generated by a known- good implementation with the same input parameters.

Decryption Algorithm Functional Tests

329. The evaluator shall test the authenticated decrypt functionality of AES-GCM by supplying 15 test cases for the supported combinations of the parameters as described above. For each parameter combination the evaluator shall introduce an error into either the Ciphertext or the Tag such that approximately half of the cases are correct and half the cases contain errors.

4.2.2. FCS_COP.1/KeyEncap Cryptographic Operation - Key Encapsulation

4.2.2.1. TSS

330. The evaluator shall ensure that the TSS documents that the selection of the key size is sufficient for the security strength of the key encapsulated.

331. The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed and used in accordance with the relevant standards.

4.2.2.2. Guidance Documentation

332. There are no additional Guidance evaluation activities for this component.

4.2.2.3. Tests

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

334. The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests

executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

ML-KEM Key Encapsulation

335. To test the TOE's implementation of ML-KEM key encapsulation/decapsulation, the evaluator shall perform the Encapsulation Test and the Decapsulation Test using the following input parameters:
    - Encapsulation Parameters:
        - Parameter set [ML-KEM-1024]
        - Previously generated encapsulation key (ek)
        - Random value (m) [32 bytes]
    - Decapsulation Parameters:
        - Parameter set [ML-KEM-1024]
        - Previously generated decapsulation key (dk)
        - Previously generated ciphertext (c) [32 bytes]

Encapsulation Test

336. For each supported parameter set the evaluator shall generate 25 test cases consisting of an encapsulation key ek and random value m. For each test case the valuator shall require the implementation under test to generate the corresponding shared secret k and ciphertext c. To determine correctness, the evaluator shall compare the resulting values with those generated using a known-good implementation using the same inputs.

Encapsulation Key Check (if supported)

337. The evaluator shall generate 10 encapsulation keys such that:
    - Five of the encapsulation keys are valid, and
    - Five of the encapsulation keys are modified such that a value in the noisy linear system is encoded into the key as a value greater than Q.

338. The evaluator shall invoke the TOE's Encapsulation Key Check functionality to determine the validity of the 10 keys. The unmodified keys should be determined valid, and the modified keys should be determined invalid.

Decapsulation Key Check (if supported)

339. The evaluator shall generate 10 decapsulation keys such that:
    - Five of the decapsulation keys are valid, and
    - Five of the decapsulation keys are modified such that the concatenated values ek||H(ek) will no longer match by modifying H(ek) to be a different value.

340. The evaluator shall invoke the TOE's Decapsulation Key Check functionality to determine the validity of the 10 keys. The unmodified keys should be determined valid, and the modified keys should be determined invalid.

Decapsulation Test

341. For each supported parameter set the evaluator shall use a single previously generated decapsulation key dk and generate 10 test cases consisting of valid and invalid ciphertexts c. For each test case the evaluator shall require the implementation under test to generate the corresponding shared secret k whether or not the ciphertext is valid. To determine correctness, the evaluator shall compare the resulting values with those generated using a known-good implementation using the same inputs.

4.2.3. FCS_COP.1/KeyWrap Cryptographic Operation - Key Wrapping

4.2.3.1. TSS

342. The evaluator shall ensure that the TSS documents that the selection of the key size is sufficient for the security strength of the key wrapped.

343. The evaluator shall examine the TSS to ensure that it describes the construction of any IVs, nonces, and MACs in conformance with the relevant specifications.

4.2.3.2. Guidance Documentation

344. There are no additional Guidance evaluation activities for this component.

4.2.3.3. Tests

345. For tests of AES-GCM and AES-CCM, see testing for FCS_COP.1/AEAD.

346. The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as

close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

AES-KW

347. To test the TOE's ability to wrap keys using AES in Key Wrap mode the evaluator shall perform the Algorithm Functional Tests using the following input parameters:
    - Key size [256] bits
    - Keyword cipher type [cipher, inverse]
    - Payload sizes [128-4096] bits by 64s

Algorithm Functional Test

348. The evaluator shall generate 100 encryption test cases using random data for each combination of claimed key size, keyword cipher type, and six supported payload sizes such that the payload sizes include the minimum, the maximum, two that are divisible by 128, and two that are not divisible by 128.
349. The results shall be compared with those generated by a known-good implementation using the same inputs.
350. The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.

AES-KWP

351. To test the TOE's ability to wrap keys using AES in Key Wrap with Padding mode with padding the evaluator shall perform the Algorithm Functional Tests using the following input parameters:
    - Key size [256] bits
    - Keyword cipher type [cipher, inverse]
    - Payload sizes [8-4096] bits by 8s

Algorithm Functional Test

352. The evaluator shall generate 100 encryption test cases using random data for each combination of claimed key size, keyword cipher type, and six supported payload sizes such that the payload sizes include the minimum, the maximum, two that are divisible by 128, and two that are not divisible by 128.

353. The results shall be compared with those generated by a known-good implementation using the same inputs.

354. The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.

4.2.4. FCS_COP.1/SKC Cryptographic Operation - Symmetric Key Cryptography

4.2.4.1. TSS

355. The evaluator shall examine the TSS to ensure that it describes the construction of any IVs, tweak values, and counters in conformance with the relevant specifications.

356. If XTS-AES is claimed then the evaluator shall examine the TSS to verify that the TOE creates full-length keys by methods that ensure that the two key halves are different and independent.

357. The evaluator shall examine the TSS to ensure it identifies the key size(s) and mode(s) supported by the TOE for data AES encryption/decryption.

4.2.4.2. Guidance Documentation

358. The evaluator shall verify that the AGD guidance instructs the administrator on how to configure the TOE to use the selected mode(s) and key size(s) for AES encryption/decryption.

4.2.4.3. Tests

AES-CBC

359. To test the TOE's ability to encrypt/decrypt data using AES in CBC mode, the evaluator shall perform Algorithm Functional Tests and Monte Carlo Tests using the following input parameters:

- Key size [128, 256] bits
- Direction [encryption, decryption]

Algorithm Functional Test

360. Algorithm Functional Tests are designed to verify the correct operation of the logical components of the algorithm implementation under normal operation using different block sizes. For AES-CBC, there are two types of AFTs:

Known-Answer Tests

361. For each combination of direction and claimed key size, the TOE must be tested using the GFSBox, KeySbox, VarTxt, and VarKey test cases listed in Appendixes B through E of The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), NIST, 15 November 2002.

Multi-Block Message Tests

362. For each combination of direction and claimed key size, the TOE must be tested against 10 test cases consisting of a random IV, random key, and random plaintext/ciphertext. The plaintext/ciphertext starts with a length of 16 bytes and increases by 16 bytes for each test case until reaching 160 bytes.

Monte Carlo Tests

363. Monte Carlo tests are intended to test the implementation under strenuous conditions. The TOE must process the test cases according to the following algorithm once for each combination of direction and key size:

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

```
            PT[0] = CT[j-1]
        }
```

where AES_KEY_SHUFFLE is defined as:

```
If ( keylen = 128 )
        Key[i+1] = Key[i] xor MSB(CT[j], 128)
If ( keylen = 192 )
        Key[i+1] = Key[i] xor (LSB(CT[j-1], 64) || MSB(CT[j], 128))
If ( keylen = 256 )
        Key[i+1] = Key[i] xor (MSB(CT[j-1], 128) || MSB(CT[j], 128))
```

The above pseudocode is for encryption. For decryption, swap all instances of CT and PT.

The initial IV, key, and plaintext/ciphertext should be random.

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

XTS-AES

364. To test the TOE's ability to encrypt/decrypt data using AES in XTS mode, the evaluator shall perform the Single Data Unit Test and the Multiple Data Unit Test using the following input parameters:
    - Direction [encryption, decryption]
    - Key size [512] bits
    - Tweak value format [128-bit hex string, data unit sequence number]

Single Data Unit Test

365. For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. The payload size and data unit size must be equal.

Multiple Data Unit Test

366. For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random

payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. Likewise, the data unit size is determined randomly for each test case from supported values within the range [128-65535] bits. The payload size and data unit size must not be equal.

367. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

AES-CTR

368. To test the TOE's ability to encrypt/decrypt data using AES in CTR mode, the evaluator shall perform the Algorithm Functional Test and the Counter Test using the following input parameters:

- Direction [encryption, decryption]
- Key size [128, 256] bits

Algorithm Functional Tests

369. Algorithm Functional Tests are designed to verify the correct operation of the logical components of the algorithm implementation under normal operation using different block sizes. For AES-CTR, there are three types of AFTs:

Known-Answer Tests

370. For each combination of direction and claimed key size, the TOE must be tested using the GFSBox, KeySbox, VarTxt, and VarKey test cases listed in Appendixes B through E of The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), NIST, 15 November 2002.

Single Block Message Tests

371. For each combination of direction and claimed key, the evaluator shall generate 10 test cases with a data size of 128 bits.

Partial Block Message Tests

372. Monte Carlo tests are intended to test the implementation under strenuous conditions. The TOE must process the test cases according to the following algorithm once for each combination of direction and key size:

373. For each combination of direction and claimed key, the evaluator shall generate five test cases such that the data size is not a multiple of 128 bits.

374. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Counter Test

375. The evaluator shall generate a single message of 1000 blocks (128000 bits) and either encrypt or decrypt it. Back-compute the IVs used. Verify that they are unique and increasing (encryption) or decreasing (decryption).

4.2.5. FCS_COP.1/CMAC Cryptographic Operation - CMAC

4.2.5.1. TSS

376. The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and 128 bit key size supported by the TOE for CMAC.

4.2.5.2. Guidance Documentation

377. There are no additional Guidance evaluation activities for this component.

4.2.5.3. Tests

CMAC Generation Test

378. To test the generation capability of AES-CMAC, the evaluator shall provide to the TSF, for each key length-message length-CMAC length tuple (in bytes), a set of eight arbitrary key-plaintext tuples that will result in the generation of a known MAC value when encrypted. The evaluator shall then verify that the correct MAC was generated in each case.

CMAC Verification Test

379. To test the verification capability of AES-CMAC, the evaluator shall provide to the TSF, for each key length-message length-CMAC length tuple (in bytes), a set of 20 arbitrary key-MAC tuples that will result in the generation of known messages when verified. The evaluator shall then verify that the correct message was generated in each case.

380. The following information should be used by the evaluator to determine the key length-message length-CMAC length tuples that should be tested:

Key length: Values will include the following:
16
Message length: Values will include the following:
0 (optional)
Largest value supported by the implementation (no greater than 65536)
Two values divisible by 16
Two values not divisible by 16
CMAC length:
Smallest value supported by the implementation (no less than 1)
381. 16
Any supported CMAC length between the minimum and maximum values

4.2.6. FCS_RBG.2 Random Bit Generation (External Seeding - VS platform)

381. The evaluator shall examine the entropy documentation required by FCS_RBG.1 to verify that it identifies, for each DRBG function implemented by the TOE, the TSF external interface used to seed the TOE's DRBG. The evaluator shall verify that this includes the amount of sampled data and the min-entropy rate of the sampled data such that it can be determined that sufficient entropy can be made available for the highest strength keys that the TSF can generate (e.g., 256 bits). If the seed data cannot be assumed to have full entropy (e.g., the min-entropy of the sampled bits is less than 1), the evaluator shall ensure that the entropy documentation describes the method by which the TOE estimates the amount of entropy that has been accumulated to ensure that sufficient data is collected and any conditioning that the TSF applies to the output data to create a seed of sufficient size with full entropy.

4.2.6.1. TSS

FCS_RBG.2.1

382. There are no additional TSS evaluation activities for this component.

4.2.6.2. Guidance Documentation

FCS_RBG.2.1

383. There are no additional Guidance evaluation activities for this component.

4.2.6.3. Tests

FCS_RBG.2.1

384. There are no test activities for this component.

4.2.7. FCS_RBG.3 Random Bit Generation (Internal Seeding - Single Source)

385. The evaluator shall examine the entropy documentation required by FCS_RBG.1 to verify that it identifies, for each DRBG function implemented by the TOE, the TSF entropy source used to seed the TOE's DRBG. The evaluator shall verify that this includes the amount of sampled data and the min-entropy rate of the sampled data such that it can be determined that sufficient entropy can be made available for the highest strength keys that the TSF can generate (e.g., 256 bits). If the seed data cannot be assumed to have full entropy (e.g., the min-entropy of the sampled bits is less than 1), the evaluator shall ensure that the entropy documentation describes the method by which the TOE estimates the amount of entropy that has been accumulated to ensure that sufficient data is collected and any conditioning that the TSF applies to the output data to create a seed of sufficient size with full entropy.

4.2.7.1. TSS

FCS_RBG.3.1

386. There are no additional TSS evaluation activities for this component.

4.2.7.2. Guidance Documentation

FCS_RBG.3.1

387. There are no additional Guidance evaluation activities for this component.

4.2.7.3. Tests

FCS_RBG.3.1

388. There are no test activities for this component.

4.2.8. FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)

389. The evaluator shall examine the entropy documentation required by FCS_RBG.1 to verify that it identifies, for each DRBG function implemented by

the TOE, each TSF entropy source used to seed the TOE's DRBG. The evaluator shall verify that this includes the amount of sampled data and the min-entropy rate of the sampled data from each data source.

4.2.8.1. TSS

FCS_RBG.4.1

390. There are no additional TSS evaluation activities for this component.

4.2.8.2. Guidance Documentation

FCS_RBG.4.1

391. There are no additional Guidance evaluation activities for this component.

4.2.8.3. Tests

FCS_RBG.4.1

392. There are no test activities for this component.

4.2.9. FCS_RBG.5 Random Bit Generation (Combining Entropy Sources)

393. Using the entropy sources specified in FCS_RBG.4, the evaluator shall examine the entropy documentation required by FCS_RBG.1 to verify that it describes the method by which the various entropy sources are combined into a single seed. This should include an estimation of the rate at which each noise source outputs data and whether this is dependent on any system-specific factors so that each source's relative contribution to the overall entropy is understood. The evaluator shall verify that the resulting combination of sampled data and the min-entropy rate of the sampled data is described in sufficient detail to determine that sufficient entropy can be made available for the highest strength keys that the TSF can generate (e.g., 256 bits). If the seed data cannot be assumed to have full entropy (e.g., the min-entropy of the sampled bits is less than 1), the evaluator shall ensure that the entropy documentation describes the method by which the TOE estimates the amount of entropy that has been accumulated to ensure that sufficient data is collected and any conditioning that the TSF applies to the output data to create a seed of sufficient size with full entropy.

4.2.9.1. TSS

FCS_RBG.5.1

394. There are no additional TSS evaluation activities for this component.

4.2.9.2. Guidance Documentation

FCS_RBG.5.1

395. There are no additional Guidance evaluation activities for this component.

4.2.9.3. Tests

FCS_RBG.5.1

396. There are no test activities for this component.

4.2.10. FCS_COP.1/XOF Extendable-Output Function

4.2.10.1. TSS

FCS_COP.1.1/XOF

397. There are no additional TSS evaluation activities for this component.

4.2.10.2. Guidance Documentation

FCS_COP.1.1/XOF

398. There are no additional Guidance evaluation activities for this component.

4.2.10.3. Tests

FCS_COP.1.1/XOF

399. To test SHAKE-128 or SHAKE-256 the evaluator shall collect the input sizes allowed by the implementation, [0-65536] and the output sizes allowed by the implementation, [16-65536] and compare the results against a known good implementation.

400. For the variable length inputs, the evaluator shall test the following:

    a. Every message length up to the rate (1600 – 2*{128, 256})

    b. About 100 random message lengths greater than the rate

c. All have an output length of {128, 256}

401. For variable length outputs, the evaluator shall test the following:

a. Minimum output length,

b. Maximum output length,

c. 510 other random output lengths (for a total of 512 tests)

d. All have an input length of {128, 256}

4.2.11. FCS_IPSEC_EXT.1 IPsec Protocol

4.2.11.1. TSS

FCS_IPSEC_EXT.1.1

402. 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.

403. 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.1.2

404. None.

FCS_IPSEC_EXT.1.3

405. The evaluator shall check 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

406. The evaluator shall examine the TSS to verify that the selected algorithms are implemented. In addition, the evaluator shall ensure that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1/KeyedHash Cryptographic Operations (for keyed-hash message authentication) and if the SHA-based HMAC function truncated output is utilised it must also be described.

FCS_IPSEC_EXT.1.5

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

408. 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

409. 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

410. 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.1.7.

FCS_IPSEC_EXT.1.8

411. 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.1.8.

FCS_IPSEC_EXT.1.9

412. 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

413. 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.

414. 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

415. 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 shall check to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

FCS_IPSEC_EXT.1.12

416. 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

417. The evaluator shall ensure 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/SigVer Cryptographic Operation - Signature Verification.

418. 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

419. 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.

4.2.11.2. Guidance Documentation

FCS_IPSEC_EXT.1.1

420. 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.1.2

421. None.

FCS_IPSEC_EXT.1.3

422. 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

423. The evaluator shall check the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

FCS_IPSEC_EXT.1.5

424. 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).

425. 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

426. The evaluator shall ensure that the guidance documentation describes the configuration of all selected algorithms in the requirement.

FCS_IPSEC_EXT.1.7

427. 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, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g., configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours. 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

428. 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, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g., configure a time value of 7h 45min to ensure the actual rekey is

performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 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.1.9 and FCS_IPSEC_EXT.1.10

429. None.

FCS_IPSEC_EXT.1.11

430. The evaluator shall ensure that the guidance documentation describes the configuration of all algorithms selected in the requirement.

FCS_IPSEC_EXT.1.12

431. None.

FCS_IPSEC_EXT.1.13

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

433. 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.

434. The evaluator shall 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

435. 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.

4.2.11.3. Tests

FCS_IPSEC_EXT.1.1

436. The evaluator shall use the guidance documentation to configure the TOE to perform 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 shall perform 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 shall observe 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 shall ensure 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 behaviour is exhibited, and is consistent with both the TSS and the guidance documentation.

FCS_IPSEC_EXT.1.2

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

438. The evaluator shall use the guidance documentation to configure the TOE to perform 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 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 shall 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 shall send the packet and observe that the packet was dropped.

FCS_IPSEC_EXT.1.3

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

a. Test 1: If tunnel mode is selected, the evaluator shall use 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 shall configure 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 shall observe (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 shall use 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 shall configure 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 shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator shall observe (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

440. 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

441. 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 shall 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 7296, Section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

FCS_IPSEC_EXT.1.6

442. 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 shall confirm the algorithm was that used in the negotiation.

FCS_IPSEC_EXT.1.7

443. When testing this functionality, the evaluator shall 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."

444. 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 a 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 no later than 24 hours

for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the Phase 1 SA lifetime on the TOE. The evaluator shall establish a 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

445. When testing this functionality, the evaluator shall 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."

446. 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 a 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 no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE. The evaluator shall establish a 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.1.9

447. None.

FCS_IPSEC_EXT.1.10

448. The following tests shall be performed.

a. Test 1: If IKEv1 is supported, Configure the TOE to use IKEv1:

- Test 1.a: If "according to the security strength associated with the negotiated Diffie- Hellman group" has been selected, for each supported authentication methods and DH groups, demonstrate the TOE uses phase 1 and phase 2 nonces meeting the strength requirement defined in NIST SP 800-57 for the appropriate DH group.
- Test 1.b: If "at least 128-bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash" has been selected, for each supported authentication method and PRF hash, demonstrate the TOE uses phase 1 and phase 2 nonces suitable for the selected PRF.

b. Test 2: If IKEv2 is supported, configure the TOE to use IKEv2:

- For each supported DH group, demonstrate the TOE uses IKE_SA_INIT and CREATE_CHILD_SA nonces suitable for the selected PRF.

FCS_IPSEC_EXT.1.11

449. 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

450. The evaluator shall follow 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 a 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 a SA for ESP (assumes the proper parameters were 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

451. For efficiency sake, the testing is combined with the testing for FIA_X509_EXT.1 and FIA_X509_EXT.2 from the Functional Package for X.509 (for IPsec connections), and FCS_IPSEC_EXT.1.1.

FCS_IPSEC_EXT.1.14

452. In the context of the tests below, a valid certificate is a certificate that passes FIA_X509_EXT.1 (from the Functional Package for X.509) validation checks but does not necessarily contain an authorised subject.

453. The evaluator shall perform the following tests:

a. 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.

b. 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.

c. Test 3 [conditional]: For each CN/identifier type combination selected, the evaluator shall:

    i. 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.

    ii. 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.

d. Test 4 [conditional]: For each SAN/identifier type combination selected, the evaluator shall:

    i. 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.

    ii. 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.

e. 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.

f. 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:

    i. Duplicate the CN field, so the otherwise authorised DN contains two identical CNs.

    ii. Append '\0' to a non-CN field of an otherwise authorised DN.

4.2.12. FCS_NTP_EXT.1 NTP Protocol

4.2.12.1. TSS

FCS_NTP_EXT.1.1

454. The evaluator shall examine the TSS to ensure it 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.

455. 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.

FCS_NTP_EXT.1.2, FCS_NTP_EXT.1.3, and FCS_NTP_EXT.1.4

456. None.

4.2.12.2. Guidance Documentation

FCS_NTP_EXT.1.1

457. The evaluator shall examine the guidance documentation to ensure it provides the Security 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

458. For each of the secondary selections made in the ST, the evaluator shall examine the guidance document to ensure it instructs the Security 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 Security Administrator how to configure the TOE to use the chosen option(s).

FCS_NTP_EXT.1.3

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

FCS_NTP_EXT.1.4

460. None.

4.2.12.3. Tests

FCS_NTP_EXT.1.1

461. 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

462. 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 shall configure the TOE and NTP server so the TOE can synchronize time using a claimed message digest algorithm. The evaluator shall modify the response(s) from the NTP server so the response(s) contains a MAC that was generated by a different message digest algorithm and confirm the TOE does not synchronize to this time source. Other than the invalid MAC, the NTP response(s) must be valid (e.g., key ID, key value used, timestamps).

Note: Since the algorithm is not identified in the server response, this tests an incorrect algorithm and an invalid MAC.

The evaluator shall use a packet sniffer to capture the network traffic between the TOE and the NTP server. The evaluator shall use 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

463. 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.

FCS_NTP_EXT.1.4

464. The evaluator shall perform the following tests:

a. Test 1: The evaluator shall confirm the TOE supports configuration of at least three (3) NTP time sources. The evaluator shall configure at least three NTP servers to support periodic time updates to the TOE. The evaluator shall confirm the TOE is configured to accept NTP packets that would result in the timestamp being updated from each of the NTP servers. The evaluator shall check that the time stamp is updated after receipt of the NTP packets. The purpose of this test to verify that the TOE can be configured to synchronize with multiple NTP servers. It is up to the evaluator to determine that the multi-source update of the time information is appropriate and consistent with the behaviour prescribed by the RFC 1305 for NTPv3 and RFC 5905 for NTPv4.

b. Test 2: (The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers.) The evaluator shall confirm that the TOE would not synchronize to other, not explicitly configured time sources by sending an otherwise valid but unsolicited NTP Server response indicating different time from the TOE's current system time. This rogue time source needs to be configured in a way (e.g., degrade or disable valid and configured NTP servers) that could plausibly result in unsolicited updates becoming a preferred time source if they are not discarded by the TOE. The TOE is not mandated to respond in a detectable way or audit the occurrence of such unsolicited updates. It is up to the evaluator to craft and transmit unsolicited updates in a way that would be consistent with the behaviour of a correctly-functioning NTP server.

4.3. Identification and Authentication (FIA)

4.3.1. FIA_AFL.1 Authentication Failure Management

4.3.1.1. TSS

465. 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.

466. 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).

4.3.1.2. Guidance Documentation

467. 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.

468. 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.

4.3.1.3. Tests

469. 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.

4.3.2. FIA_UAU.7 Protected Authentication Feedback

4.3.2.1. TSS

470. None

4.3.2.2. Guidance Documentation

471. The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

4.3.2.3. Tests

472. 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.

4.3.3. FIA_PMG_EXT.1 Password Management

4.3.3.1. TSS

473. The evaluator shall check that the TSS:

a. lists the supported special character(s) for the composition of administrator passwords.

b. to ensure that the minimum_password_length parameter is configurable by a Security Administrator.

c. lists the range of values supported for the minimum_password_length parameter. The listed range shall include the value of 15.

4.3.3.2. Guidance Documentation

474. 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.

4.3.3.3. Tests

475. The evaluator shall perform the following tests.

a. Test 1: The evaluator shall compose passwords that 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.

b. Test 2: The evaluator shall compose passwords that do not meet the requirements in some way. For each password, the evaluator shall verify that the TOE does not support the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that the TOE enforces the allowed characters and the minimum length listed in the requirement and justify the subset of those characters chosen for testing.

4.3.4. FIA_PSK_EXT.1 Pre-Shared Key Composition

4.3.4.1. TSS

476. If "generate" is selected, the evaluator shall confirm that this process uses the RBG specified in FCS_RBG.1 and the output matches the size required in FIA_PSK_EXT.1.2.

4.3.4.2. Guidance Documentation

477. The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure the mandatory_or_not flag per RFC 8784.

4.3.4.3. Tests

478. The evaluator shall attempt to establish a connection and confirm that the connection requires the selected factors in the PSK to establish the connection in alignment with table 1 from RFC 8784.

4.4. Protection of the TSF (FPT)

4.4.1. FPT_APW_EXT.1 Protection of Administrator Passwords

4.4.1.1. TSS

479. 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.

4.4.1.2. Guidance Documentation

480. None

4.4.1.3. Tests

481. None

4.4.2. FPT_TUD_EXT.2 Trusted Update Based on Certificates

4.4.2.1. TSS

482. 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.

483. The evaluator shall verify that the TSS describes how the TOE reacts if X.509v3 certificates are used for trusted updates and the Security Administrator attempts to perform the trusted update using an expired certificate.

484. The TSS shall describe the point at which revocation checking is performed and describe whether the Security Administrator can manually provide revocation information. 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.509v3 certificate only when it is loaded onto the device.

4.4.2.2. Guidance Documentation

485. The evaluator shall verify that the guidance documentation describes how the TOE reacts if X.509v3 certificates are used for trusted updates and the administrator attempts to perform the trusted update using an expired certificate. The evaluator shall verify any Security Administrator actions related to revocation checking, both accepting or rejecting certificates and manually providing revocation information. The description shall correspond to the description in the TSS.

4.4.2.3. Tests

486. The evaluator shall perform the following tests:

a. Test 1: The evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1 (in the Functional Package for X.509) and a check for the Code Signing purpose in the extendedKeyUsage.

b. Test 2: 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.

c. Test 3: 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.509v3 certificate only when it is loaded onto the device.

4.5. Security management (FMT)

4.5.1. FMT_MOF.1/Services Management of Security Functions Behaviour

4.5.1.1. TSS

487. For distributed TOEs, see Section 2.4.1.1.

488. For non-distributed TOEs, the evaluator shall verify that the TSS lists the services that the Security Administrator is able to start and stop and how that operation is performed.

4.5.1.2. Guidance Documentation

489. For distributed TOEs, see Section 2.4.1.2.

490. For non-distributed TOEs, the evaluator shall also verify that the Guidance Documentation describes how the TSS lists the services that the Security Administrator is able to start and stop and how that operation is performed.

4.5.1.3. Tests

491. The evaluator shall perform the following tests:

a. Test 1: 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.

When there is an implementation where no other users other than the Security Administrator can be defined, the user might not be able to get to the point where the attempt to enable/disable this service/these services can be executed without user authentication. 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 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.

4.5.2. FMT_MOF.1/AutoUpdate Management of Security Functions Behaviour

4.5.2.1. TSS

492. For distributed TOEs, see Section 2.4.1.1.

493. For non-distributed TOEs, the evaluator shall ensure the TSS describes how checking for automatic updates or automated updates (whichever is supported by the TOE) are performed to include required configuration settings to enable and disable automated checking or automated updates.

4.5.2.2. Guidance Documentation

494. For distributed TOEs, see Section 2.4.1.2.

495. For non-distributed TOEs, the evaluator shall also ensure the TSS describes how checking for automatic updates or automated updates (whichever is supported by the TOE) are performed to include required configuration settings to enable and disable automated checking or automated updates (whichever is supported by the TOE).

4.5.2.3. Tests

496. The evaluator shall perform the following tests:

a. Test 1: 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. = When there is an implementation where no other users than the Security Administrator can be defined, the user might not be able to get to the point where the attempt to enable/disable automatic checking for updates can be executed without user authentication. 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 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.

4.5.3. FMT_MOF.1/Functions Management of Security Functions Behaviour

4.5.3.1. TSS

497. For distributed TOEs, see Section 2.4.1.1.

498. For non-distributed TOEs, the evaluator shall ensure that the TSS, for each administrative function identified, details how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, and audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE).

4.5.3.2. Guidance Documentation

499. For distributed TOEs, see Section 2.4.1.2.

500. For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, and audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE) are performed to include required configuration settings.

4.5.3.3. Tests

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

501. The evaluator shall perform the following tests:

a. Test 1: 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 must fail.

When there is an implementation where no other users than the Security Administrator can be defined, the user might not be able to get to the point where the attempt to modify the security related parameters can be executed without user authentication. 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 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.

   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.

If 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection

502. The evaluator shall perform the following tests:

a. Test 1: 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 must fail.

   When there is an implementation where no other users than the Security Administrator can be defined, the user might not be able to get to the point where the attempt can be executed without user authentication. 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.1.4, FAU_STG_EXT.1.5 and FAU_STG_EXT.2.

b. Test 2: 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 shall be confirmed. The term 'handling of audit data'

refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.4, FAU_STG_EXT.1.5 and FAU_STG_EXT.2.

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 must be tested.

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

503. The evaluator shall perform the following tests:

a. Test 1: 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.

When there is an implementation where no other users than the Security Administrator can be defined, the user might not be able to get to the point where the attempt can be executed without user authentication. 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 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.

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 must be tested.

If in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection

504. The evaluator shall perform the following tests:

a. Test 1: 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.

When there is an implementation where other users than the Security Administrator can be defined, the user might not be able to get to the point where the attempt can be executed without user authentication. 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 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 Security Administrator authentication shall be successful.

4.5.4. FMT_MTD.1/CryptoKeys Management of TSF Data

4.5.4.1. TSS

505. For distributed TOEs, see Section 2.4.1.1.

506. For non-distributed TOEs, the evaluator shall ensure the TSS lists the keys the Security Administrator is able to manage to include the options available (e.g., generating keys, importing keys, modifying keys or deleting keys) and names the operations that are performed.

4.5.4.2. Guidance Documentation

507. For distributed TOEs, see Section 2.4.1.2.

508. For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the operations are performed on the keys the Security Administrator is able to manage.

4.5.4.3. Tests

509. The evaluator shall perform the following tests:

a. Test 1: 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.

When there is an implementation where other users than the Security Administrator can be defined, the user might not be able to get to the point where the attempt can be executed without user authentication. 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 try to perform at least one of the related actions with prior authentication as Security Administrator. This attempt should be successful.

4.6. TOE Access (FTA)

4.6.1. FTA_SSL_EXT.1 TSF-initiated Session Locking

4.6.1.1. TSS

510. The evaluator shall examine the TSS to determine that it details whether local administrative session locking or termination is supported and the related inactivity time period settings.

4.6.1.2. Guidance Documentation

511. 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.

4.6.1.3. Tests

512. The evaluator shall perform the following test:

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

5. Evaluation Activities for SARs

513. The sections below specify EAs for the SARs included in the related cPPs (see Section 1.1). 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.

514. 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 Evaluation Activities for TOE Summary Specification (ASE_TSS.1) for All TOEs

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

5.1.2. Additional Evaluation Activities for TOE Summary Specification (ASE_TSS.1) for Distributed TOEs

516. 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.

ASE_TSS.1 element	Evaluator Action
	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.

517. Note that additional Evaluation Activities for the TSS in the case of a distributed TOE are defined in Annex B.4.

5.2. ADV: Development

5.2.1. Basic Functional Specification (ADV_FSP.1)

518. The EAs for this assurance component focus on understanding the interfaces (e.g., application programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the 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.

519. 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.

520. 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.

521. 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.1-4, 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 method of use for each SFR-supporting and SFR-enforcing TSFI is given.	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.1-3 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.	Work unit ADV_FSP.1-4 from the [CEM] states: “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-

CEM ADV_FSP.1 Work Units	Evaluation Activities
	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 and confirm that all security-relevant interfaces are described. The evaluator shall also confirm that all internal (FPT_ITT.1), administrative (FTP_TRP.1/Admin), and trusted channel (FTP_ITC.1) requirements unambiguously trace to documented interfaces.
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

CEM ADV_FSP.1 Work Units	Evaluation Activities
	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

522. *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.*

523. 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). Explicitly labelling TSFI as security relevant or non-security relevant is not necessary. A TSFI is implicitly security relevant if it is used to satisfy an evaluation activity, or if it is identified in the ST or guidance documentation as adhering to the security policies (as presented in the SFRs). 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. According to the description above 'security relevant' corresponds to the combination of 'SFR-enforcing' and 'SFR-supporting' as defined in CC Part 3, page 36 last paragraph.

524. The set of TSFI that are provided as evaluation evidence are contained in the Security Target and the guidance documentation.

5.2.1.2. Evaluation Activity

525. *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

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

527. The evaluator shall use the provided documentation to identify and examine a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

528. It should be noted that there may be some SFRs that do not have a TSFI that is explicitly "mapped" to invoke the desired functionality. For example, generating a random bit string or destroying a cryptographic key that is no longer needed are capabilities that may be specified in SFRs, but are not invoked by an interface.

529. The required EAs define the design and interface information required to meet ADV_FSP.1. If the evaluator is unable to perform an EA, then the evaluator shall conclude that an adequate functional specification has not been provided.

5.3. AGD: Guidance Documents

530. 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 Security Administrators and users (as appropriate) as part of the TOE.

531. Note that additional EAs for the guidance documentation in the case of a distributed TOE are defined in Annex B.4.

5.3.1. Operational User Guidance (AGD_OPE.1)

532. 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. For the related evaluation activities, the evaluation evidence documents Security Target, AGD documentation (user guidance) and functional specification documentation (if provided) shall be used as input documents. Each input document is subject to ALC_CMS.1-2 requirements.

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

5.3.1.1. Evaluation Activity

534. *The evaluator shall verify the Operational guidance documentation is distributed to Security Administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that Security 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

535. *The evaluator shall verify 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

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

5.3.1.4. Evaluation Activity

537. *The evaluator shall verify 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

538. In addition, the evaluator shall verify that the following requirements are also met.

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

b. The evaluator shall verify that this process includes 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).

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 EAs.

5.3.2. Preparative Procedures (AGD_PRE.1)

539. The evaluator will perform 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.

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

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

5.3.2.1. Evaluation Activity:

542. *The evaluator shall examine the Preparative procedures to ensure they include a description of how the Security 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).*

543. The documentation should be in an informal style and should be written with sufficient detail and explanation that it 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

544. *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

545. *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

546. *The evaluator shall examine the preparative procedures to ensure they include instructions on how to manage the TSF as a product and as a component of the larger Operational Environment in a manner that allows to preserve integrity of the TSF.*

547. *The intent of this requirement is to ensure there exists adequate preparative procedures (guidance in most cases) to put the TSF in a secure state (i.e., evaluated configuration). AGD_PRE.1 lists general requirements, the specific assurance activities implementing it are performed as part of FMT_SMF.1, FMT_MTD.1 and FMT_MOF.1 series of SFRs.*

5.3.2.5. Evaluation Activity

548. In addition, the evaluator shall verify that the following requirements are also met.

549. 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 mandate that they shall be changed.

5.4. ALC: Life-cycle Support

5.4.1. Labelling of the TOE (ALC_CMC.1)

550. 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)

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

5.4.3. Basic Flaw Remediation (ALC_FLR.1) (optional)

552. When evaluating the developer's procedures regarding basic flaw remediation, the evaluator performs the work units as presented in the CEM.

5.4.4. Flaw Reporting Procedures (ALC_FLR.2) (optional)

553. When evaluating the developer's flaw reporting procedures, the evaluator performs the work units as presented in the CEM.

5.4.5. Systematic Flaw Remediation (ALC_FLR.3) (optional)

554. When evaluating the developer's procedures regarding systematic flaw remediation, the evaluator performs the work units as presented in the CEM.

5.5. ATE: Tests

5.5.1. Independent Testing – Conformance (ATE_IND.1)

555. 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.

556. 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.

557. The evaluator shall consult Annex B when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

558. Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in Annex B.4.

5.6. AVA: Vulnerability Assessment

5.6.1. Vulnerability Survey (AVA_VAN.1)

559. 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 shall follow 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.

560. 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.

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	The evaluator shall perform the CEM activity as specified.

CEM AVA_VAN.1 Work Units	Evaluation Activities
with the configuration under evaluation as specified in the ST.	The calibration of test resources specified in the last paragraph of the work unit AVA_VAN.1-1 of the [CEM] applies to the tools listed in Annex 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 Annex 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 Annex A.1, and documentation as specified in Annex 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 Annex A.2.
AVA_VAN.1-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential	The CEM work unit is captured in Annex A.3; there are no substantive differences.

CEM AVA_VAN.1 Work Units	Evaluation Activities
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.	

CEM AVA_VAN.1 Work Units	Evaluation Activities
AVA_VAN.1-7 The evaluator ***shall conduct*** penetration testing.	The evaluator shall perform the CEM activity as specified.
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 Annex 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 Annex 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.
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);	Replace the CEM work unit with the reporting called for in Annex A.3.

CEM AVA_VAN.1 Work Units	Evaluation Activities
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.	

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

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

5.6.1.1. Evaluation Activity (Documentation):

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

563. *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.*

564. The developer shall provide documentation identifying the list of software and hardware components[7] that compose the TOE. Hardware components should identify compute-capable hardware components, at a minimum that must include the processor, and where applicable, discrete crypto ASICs, TPMs, etc. used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or

cryptographic implementations, (independently identifiable and reusable components are not limited to the list provided in the example). 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 vulnerability hypotheses during their analysis.

565. 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, Table 2]

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.

5.6.1.2. Evaluation Activity:

566. The evaluator shall formulate hypotheses in accordance with process defined in Annex A. The evaluator shall document the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Annex A.3. The evaluator shall perform vulnerability analysis in accordance with Annex A.2. The results of the analysis shall be documented in the report according to Annex A.3.

6. Required Supplementary Information

567. 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.

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

7. References

[CC1] Common Criteria for Information Technology Security Evaluation,
Part 1: Introduction and General Model
CCMB-2022-11-001, Version CC:2022 Revision 1, November 2022

[CC2] Common Criteria for Information Technology Security Evaluation,
Part 2: Security Functional Components,
CCMB-2022-11-002, Version CC:2022 Revision 1, November 2022

[CC3] Common Criteria for Information Technology Security Evaluation,
Part 3: Security Assurance Components,
CCMB-2022-11-003, Version CC:2022 Revision 1, November 2022

[CC4] Common Criteria for Information Technology Security Evaluation,
Part 4: Framework for the specification of evaluation methods and activities
CCMB-2022-11-004, Version CC:2022 Revision 1, November 2022

[CEM] Common Methodology for Information Technology Security Evaluation,
Evaluation Methodology,
CCMB-2022-11-006, Version CC:2022, Revision 1, November 2022

[FIPS 140-2] FIPS PUB 140-2, Security Requirements for cryptographic modules, May 25, 2001

[FIPS 140-3] FIPS PUB 140-3, Security Requirements for cryptographic modules, March 22 2019

[ISO/IEC 19790:2015]ISO/IEC 19790:2012/Cor 1:2015 Security requirements for cryptographic Modules, August 2012

[ISO/IEC 19790:2025] ISO/IEC 19790:2025 Security requirements for cryptographic Modules, February 2025

[FIPS 186-5] FIPS PUB 186-5, Digital Signature Standard (DSS), February 2023

[NDcPP] collaborative Protection Profile for Network Devices,
Version 4.0, 25 November 2025

[NIST SP800-56A] NIST Special Publication SP800-56A Revision 3: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, Apr 2018

[NIST SP800-90A] NIST Special Publication SP800-90A Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 24 2015

[SHAVS] The Secure Hash Algorithm Validation System (SHAVS), Updated: May 21, 2014

Annex A: Vulnerability Analysis

A.1. Sources of Vulnerability Information

569. 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 hypotheses 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 Annex 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 Annex 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 Annex 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 Annex 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 Annex A.1.4.

A.1.1. Type 1 Hypotheses – Public-Vulnerability-Based

570. The list of public sources of vulnerability information selected by the iTC is given in Annex A.4.

571. The evaluators shall perform a search on the sources listed in Annex A.4 to determine a list of potential flaw hypotheses 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.

572. According to Section 5.6.1.1, the developer shall provide documentation identifying the list of software and hardware components that compose the TOE. The evaluator shall independently verify this list for completeness by comparing it to the security functionality defined in the TSS of the ST and ensuring that all expected components are accounted for. Hardware components should identify at a minimum the processors used by the TOE. Software components that are in the scope of this requirement include libraries, frameworks, operating system and other major components that are independently identifiable and reusable (i.e., can be present in other products) components. The evaluator shall use the components list and determine that the TOE and its components are free of unmitigated vulnerabilities. It is expected that all remotely exploitable vulnerabilities present in the network device shall be considered as part of vulnerability assessment ("network device" is used to refer to the entire device and is not limited to the claimed security functionality). The search criteria to be used when searching the sources shall include:

    - The list of software and hardware components that compose the TOE
    - The TOE name (including model information as appropriate)

      As the search terms can contain proprietary information and there is a possibility that this information could be used by attackers to identify potential attack surfaces, there is no expectation that search terms containing proprietary information are published in any public-facing document.

573. 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

574. Annex 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.

575. 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 CB to determine the appropriate means of submitting the flaw for consideration by the iTC.

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

576. 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.

577. 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 CB to determine the appropriate means of submitting the flaw for consideration by the iTC.

A.1.4. Type 4 Hypotheses – Tool-Generated

578. The evaluator is recommended but not mandated to perform Fuzz Testing to generate Type 4 flaw hypotheses. Fuzz testing is a method where a set of randomly generated inputs is used to induce an observable fault. If this approach is selected, the evaluator shall e.g., identify a protocol implementing one or more SFR-enforcing TSFI(s) and conduct fuzz testing by varying selected fields in that protocol. For example, if the TOE's remote administrative interface is secured with TLS, Fuzz Testing of the SSL record header or the TLS handshake message would be appropriate. Any results that are unexpected (e.g., core dumps, inappropriate errors, or unplanned reboots) should be investigated and resolved. If Fuzz Testing is performed, the evaluator shall focus on security-relevant parts of the TOE that have not been subjected to Fuzz Testing multiple times (like publicly available libraries), but instead focus on custom parts of the TOE (if such information is available to the evaluator).

579. Additionally, the evaluator shall utilize an automated port scanning tool as part of the vulnerability assessment process. It is up to the evaluator to select the applicable tools, however there is an expectation that any utilized tool is actively maintained.

580. 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 activity.

581. 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 CB to determine the appropriate means of submitting the flaw for consideration by the iTC.

A.2. Process for Evaluator Vulnerability Analysis

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

- The evaluator shall refine each flaw hypothesis for the TOE and attempt to disprove it using the information provided by the developer or through penetration testing.
- During this process, the evaluator is free to interact 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).
- In case of a confirmed flaw, the CB should be informed.
- 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 shall prepare an appropriate set of materials as follows:
    a. The source documents used in formulating the hypothesis, and why it represents a potential compromise against a specific TOE function;
    b. An argument why the flaw hypothesis could not be proven or disproved by the evidence provided so far;
    c. The type of information required to investigate the flaw hypothesis further.
- The CB will then either approve or disapprove the request for additional information. If approved, the developer provides the requested evidence to disprove the flaw hypothesis (or, of course, acknowledge the flaw).
- For each hypothesis, the evaluator shall note whether the flaw hypothesis has been successfully disproved, successfully proven to have identified a flaw, or requires further investigation. It is important to have the results documented as outlined in Annex A.3 below.
- If the evaluator finds a flaw, the evaluator shall report these flaws to the developer. All reported flaws must be addressed as follows:

a. 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.

b. 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).

c. 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.

d. 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.

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

584. As indicated in Annex 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 Annex A.1) flaw hypotheses only. The fact that the iTC generates these candidate hypotheses indicates these must be addressed.

A.3. Reporting

585. 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.

586. 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 Annex A.1.1;
- A statement that the evaluators have examined the Type 1 flaw hypotheses specified in this Supporting Document in Annex 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 Annex A.1.2;

587. 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.

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

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

- A list of all of the flaw hypotheses generated (see AVA_VAN.1-4);
- The evaluator penetration testing effort, outlining the testing approach, configuration, depth and results (see 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 (see 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) (see 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 Annex 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 was/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. (see AVA_VAN.1-6, AVA_VAN.1-8).

A.4. Public Vulnerability Sources

590. 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://nvd.nist.gov/vuln/search

b. CISA Known Exploited Vulnerabilities Catalog:
https://www.cisa.gov/known-exploited-vulnerabilities-catalog/

c. Common Vulnerabilities and Exposures:
https://www.cve.org/

d. US-CERT (Carnegie Mellon University):
https://www.kb.cert.org/vuls/html/search

e. Tenable Network Security
https://www.tenable.com/plugins

f. Zero Day Initiative (Trend Micro)
https://www.zerodayinitiative.com/advisories/published/

g. Exploit Database (Offensive Security):
https://www.exploit-db.com/

h. Rapid7 Vulnerability and Exploit Database:
https://www.rapid7.com/db/?type=nexpose

Annex B: Network Device Equivalency Considerations

B.1. Introduction

591. This Annex 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 NDcPP is allowed.

592. 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.

593. Determination of equivalency between models can result in different testing outcomes:

- If a set of multiple different TOE instances are determined to be equivalent, testing may be performed on a single variation of the TOE. However, if the TOE variations have security-relevant functional differences, each of the TOE models that exhibits either functional or structural differences must be separately tested. Generally speaking, only the difference between each variation of TOE must be separately tested. Other equivalent functionality may be tested on a representative model and not across multiple platforms.
- If it is determined that a TOE operates the same regardless of the environment, testing may be performed on a single instance for all equivalent configurations. However, if the TOE is determined to provide environment-specific functionality, testing must take place in each environment for which a difference in functionality exists. Similar to the above scenario, only the functionality affected by environment differences must be retested.

594. If a developer disagrees with the evaluator's assessment of equivalency, the CB will arbitrate between the two parties as to whether equivalency exists.

B.2. Evaluator Guidance for Determining Equivalence

B.2.1. Strategy

595. When performing the equivalency analysis, the evaluator shall 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.

596. 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

597. The following table provides a description of how an evaluator shall 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

Factor	Same/Not Same	Evaluator Guidance
		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 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.

Factor	Same/Not Same	Evaluator Guidance
	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

Factor	Same/Not Same	Evaluator Guidance
		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 model 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 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

Factor	Same/Not Same	Evaluator Guidance
		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

Factor	Same/Not Same	Evaluator Guidance
		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

598. In addition to determining what to test, the evaluation results (and resulting Certification Report) must identify the actual TOE 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

599. In the case of a distributed TOE, the Security Target will identify an evaluated configuration that consists of a number of separate TOE 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'.

600. In addition to the minimum configuration, 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 TOE description in the ST may include constraints on how such components are added, including required and/or prohibited configurations of the components.

601. Extra instances of a TOE component must have the same hardware and software as the original component included in the evaluated configuration.

602. 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

603. The evaluator shall examine the TSS to confirm it identifies any extra/additional instances of specific TOE components allowed in the ST and what effects will occur when extra instances of distributed TOE components are added. The information in the TSS shall allow the evaluator to understand how a system with one specific TOE component behaves in comparison to a system with multiple instances of that same TOE component. The TSS also shall describe how the additional TOE component instances maintain the SFRs to determine it is consistent with the role the singular TOE component plays in the evaluated configuration, and that the additional TOE components cannot be used in a way that the security functionality would be corrupted or bypassed. In general, any additional TOE component must not have a negative impact on other already-present components that are already part of the TOE.

B.4.2. Evaluator Actions for Assessing the Guidance Documentation

B.4.2.1. Guidance Documentation

604. 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 components 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.

605. 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

606. The evaluator shall test the TOE in the minimum configuration as defined in the ST (and the guidance documentation).

607. 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 shall test these additional SFR cases that were not included in the minimum configuration.

608. In addition, the evaluator shall test the following aspects for each extra component that is identified as allowed in the distributed TOE:

- Communications: the evaluator shall follow the guidance documentation to confirm, by testing, that any additional connections introduced with the extra component(s) 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 shall confirm 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.

---

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

2. Where keys are stored encrypted or wrapped under another key, 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.

3. Where TRIM is used then the TSS and/or guidance documentation is/are 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 of size less than 982 bytes which would be completely contained in the master file table).

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.

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.

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 Annex 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.

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.