www.GossamerSec.com

Assurance Activity Report for Cisco Embedded Services 9300 & 3300 Series Switches (ESS9300 & ESS3300)

Version 0.3
09/29/23

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Document: AAR-VID11394

Revision History

Revision	Date	Authors	Summary
Version 0.1	08/09/23	Miller	Initial draft
Version 0.2	09/20/23	Gossamer	Addressed ECR comments
Version 0.3	09/29/23	Gossamer	Addressed ECR comments

The TOE Evaluation was Sponsored by:

Cisco Systems, Inc,
170 West Tasman Drive
San Jose, CA 95134

Evaluation Personnel:

- Cody Cummins
- Julia Miller

Common Criteria Versions:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

Common Evaluation Methodology Versions:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017

Table of Contents

1. Introduction

This document presents evaluations results of the Cisco Embedded Services 3300 Series and 9300 Series Switches (ESS) NDcPP22e/MACsecEP12 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 CAVP Certificates

The TOE has the following CAVP certificates:

Algorithm	Description	Supported Mode	Module	CAVP Cert. #	SFR
AES	Symmetric encryption/decryption Keyed hashing Compliant with: International Organization of Standards/International Electrotechnical Organization (ISO/IEC) 18033-3 AES-KW – National Institute of Standards and Technology (NIST) Special Publication (SP) 800-38F CBC – ISO/IEC 10116 CMAC – NIST SP800-38B GCM – ISO/IEC 19722	AES Key-Wrap (KW) CBC (128, and 256) CMAC (128) GCM (128)	IC2M Rel5a	A1462	FCS_COP.1/DataEncryption FCS_COP.1(1)/KeyedHash:CMAC FCS_COP.1(5) Cryptographic Operation (MACsec AES Data Encryption/Decryption)
			MACsec	4544 (ESS3300) 4848 (ESS9300)	
SHS (SHA-1, SHA-256, SHA-512)	Cryptographic hashing services	Byte Oriented	IC2M Rel5a	A1462	FCS_COP.1/Hash
HMAC (HMAC-SHA-1, SHA-256, SHA-512)	Keyed hashing services and software integrity test	Byte Oriented	IC2M Rel5a	A1462	FCS_COP.1/KeyedHash

Algorithm	Description	Supported Mode	Module	CAVP Cert. #	SFR
Key Agreement Scheme-Shared Secret Computation (KAS-SSC)	Key agreement scheme shared secret computation Compliant with NIST SP800-56Arev3	FFCDH Primitive (P-256 and P-384 curves)	IC2M Rel5a	A1462	FCS_CKM.2
Key Agreement Scheme (KAS)	Diffie-Hellman (DH Group 14 Key agreement scheme) Compliant with NIST SP800-56Arev3	Finite Field Cryptography (FFC)	IC2M Rel5a	A1462	FCS_CKM.2
DRBG	Deterministic random bit generation services in accordance with ISO/IEC 18031:2011 and NIST SP800-90A	CTR_DRBG (AES 256)	IC2M Rel5a	A1462	FCS_RBG_EXT.1
RSA	Signature verification Compliant with FIPS PUB 186-2 and FIPS PUB 186-4 Key Transport Compliant with FIPS PUB 186-4	PKCS#1 v.1.5, 2048 bit key, FIPS 186-4 Key Gen (2048 bit key)	IC2M Rel5a	A1462	FCS_CKM.1 FCS_COP.1/SigGen
Component Validation List (CVL)	Key derivation function for SSH protocol Compliant with NIST SP800-135	SSH Key Derivation Function (KDF)	IC2M Rel5a	A1462	FCS_CKM.2 FCS_SSHS_EXT.1
CVL	Key derivation function for IKEv2 protocol Compliant with NIST SP800-135	Internet Key Exchange (IKEv2) KDF	IC2M Rel5a	A1462	FCS_CKM.2 FCS_IPSEC_EXT.1

1.2 Platform Equivalence

The TOE is the Cisco Embedded Services 3300 and 9300 Series Switches, both running Internetworking Operating System (IOS)-XE 17.9.

- ESS3300 models: ESS-3300-NCP, **ESS-3300-CON**, ESS-3300-24T-NCP, ESS-3300-24T-CON
- ESS9300 models: **ESS-9300-10X-E**

The evaluator performed full testing on the following platforms (**bolded** in the list above) and microarchitectures.

- ESS3300 – Xilinx ZU3EG (ARMv8 Cortex A53)
    - MACsec: Broadcom BCM54194
- ESS9300 – CrayCore CPU integrated in DopplerGS ASIC (ARMv8 Cortex A53)
    - MACsec: MSC MACsec embedded in ASICs v1.1

For this set of products, there is one software image that is used across all of these devices in each series. This means that all ESS3300 devices share the same image and the ESS9300 has its own image. Therefore, the software image is exactly the same on every single ESS3300 model in the evaluated configuration. All Security Functionality with the exception of MACsec encryption is implemented in the one software image for each series which leverages the IOS Common Cryptographic Module (IC2M). The TOE supports MACsec using the Broadcom BCM54194 chip for all ESS33000 series devices and the proprietary Unified Access Data Plane (UADP) Application-Specific Integrated Circuit (ASIC) for ESS9300 device. The MACsec Controller (MSC) is embedded within the ASICs that are utilized by the ESS9300 device.

During testing, the TOE was enclosed within a Cisco developed hardened enclosure. It is a specially designed enclosure used for Cisco internal testing purposes only. It has no compute capabilities and is not a commercially available product. In addition, the enclosure used for testing contains the ESS3300/ESS9300 board including the integrated multi-pin BTB interface connector with pins dedicated for power input, ethernet ports, and RS-232 console ports. All TOE components also have a USB console port and USB interfaces for FLASH memory but all use of USB functionality is excluded from the evaluated configuration.

While there are different models in the ESS3300 series, they differ only in ways that do not affect the enforcement of the security functionality. The evaluation team tested against the ESS-3300-CON. The ESS-3300-NCP-E is identical with the exception of the main board not having a cooling plate. There also exists the ESS-3300-24T-NCP-E and ESS-3300-24T-CON-E which are identical to each other with the exception of the main board of the ESS-3300-24T-NCP-E not having a cooling plate. These 2 additional ESS3300 SKUs differ from the ESS-3300-CON and ESS-3300-NCP-E due to having an additional expansion board. This expansion board uses the identical SAMTEC SEAF-40-05.0-S-06-2-A-K 240-pin connector as the main ESS3300 board and therefore can be deemed equivalent and require no additional testing. Therefore the different characteristics affect only non-TSF relevant functionality (such as throughput, processing speed, number and type of network connections supported, number of

concurrent connections supported, and amount of storage). There exists only one ESS9300 model in the evaluation (ESS-9300-10X-E).

All TOE security functions with the exception of MACsec encryption are implemented in software and the TOE security behavior is the same on all the devices for each of the SFRs defined by the Security Target. These SFRs are instantiated by the same version of the TOE software and in the same way on every platform. As mentioned above the MACsec encryption is the only functionality not implemented by software. Instead, it uses the Broadcom BCM54194 chip which is the same on all ESS3300 series devices. As a result of the equivalency information above, only one platform of the ESS3300 series was tested (ESS-3300-CON) alongside the ESS9300 device (ESS-9300-10X-E).

1.3 References

The following evidence was used to complete the Assurance Activities:

- Cisco Embedded Services 9300 & 3300 Series Switches (ESS9300 & ESS3300) Security Target, Version 0.7, September 29, 2023 (**ST**)
- Cisco Embedded Services 3300 Series and 9300 Series Switches (ESS3300 & ESS9300) CC Configuration Guide, Version 0.7, September 29, 2023 (**Admin Guide**)

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profiles and describes the findings in each case.

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 (FAU_GEN.1) of the ST states each of the events is specified in the audit record is in enough detail to identify the user for which the event is associated, when the event occurred, where the event occurred, the outcome of the event, and the type of event that occurred such as generating keys, including the type of key.

Component Guidance Assurance Activities: 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).

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.

Section “Security Relevant Events” of the Admin Guide contains two tables containing al the audit records. Table 6 “General Auditable Events” contains all of the relevant syslog messages that are produced by the TOE, including a description of each audit function. Table 7 “Auditable Administrative Events” shows the administrative actions related to TSF data. Additionally, the use of each command is covered in each of the Component Guidance Assurance Activities in this AAR.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

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.

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

The evaluator created a list of the required audit events. The evaluator then collected the audit events when running the other security functional tests described by the protection profiles. For example, the required event for FPT_STM.1 is discontinuous change to time. The evaluator collected audit records when modifying the clock using administrative commands. The evaluator then recorded the relevant audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.2 User identity association (NDcPP22e:FAU_GEN.2)

2.1.2.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

See NDcPP22e:FAU_GEN.1.

Component Guidance Assurance Activities: The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

See NDcPP22e:FAU_GEN.1.

Component Testing Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

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.

See NDcPP22e:FAU_GEN.1.1.

2.1.3 Protected Audit Event Storage (NDcPP22e:FAU_STG_EXT.1)

2.1.3.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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

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.

The evaluator shall examine the TSS to ensure that it details the behavior 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.

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

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

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.

Section 6.1 (FAU_STG_EXT.1) of the ST states the TOE is a standalone TOE configured to export syslog records to a specified, external syslog server in real-time. The TOE protects communications with an external syslog server via IPsec. If the IPsec connection fails, the TOE will store audit records on the TOE when it discovers it can no longer communicate with its configured syslog server. When the connection is restored, the TOE will transmit the buffer contents when connected to the syslog server.

For audit records stored internally to the TOE the audit records are stored in a circular log file where the TOE overwrites the oldest audit records when the audit trail becomes full. The size of the logging files on the TOE is configurable by the administrator with the minimum value being 4096 (default) to 2,147,483,647 bytes of available disk space.

Only Authorized Administrators are able to clear the local logs, and local audit records are stored in a directory that does not allow administrators to modify the contents.

Component Guidance Assurance Activities: 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.

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

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

Section “IPsec Overview” of the Admin Guide provides instructions for how to establish the trusted channel to the audit server, including generating a crypto key pair for IPsec, creating trustpoints, configuring the IKEv2 profile, configuring IPsec transform sets, configuring SA lifetimes, configuring crypto maps, access control lists, and the reference identifier.

Section “Logging Configuration” of the Admin Guide provides instructions for using the “logging host” command to enable the TOE to transmit audit data to a specified external audit server. This section also provides instructions for how to configure the size of the local logging buffer. This section states that if the local storage space for audit data is full, the TOE will overwrite the oldest audit record to make room for the new audit record.

Section “Logging Protection” of the Admin Guide states that when am audit event is generated, it is simultaneously sent to the external server and the local store.

Component Testing Assurance Activities: Testing of the trusted channel mechanism for audit will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. The evaluator shall perform the following additional 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: The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.3. Depending on the configuration this means that the evaluator has to check the content of the audit data when the audit data is just filled to the maximum and then verifies that

1) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option 'drop new audit data' in FAU_STG_EXT.1.3).

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

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

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

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

Test 1: The successful establishment of the IPsec syslog connection for both TOE devices was demonstrated in FTP_ITC.1. In each case, the TOE initiated the connection without administrator intervention. The use of IPsec ensured that no audits were viewed in cleartext. The audits collected as part of FAU_GEN.1 throughout testing were gathered from the remote syslog server running rsyslog version 8.16.0, thus demonstrating that audits were successfully received by the remote syslog server.

Test 2: The option “overwrite previous audit records” is selected in FAU_STG_EXT.1.3. The logs are stored in a local buffer which has a configurable size. The evaluator issued the “show logging” command and viewed the contents of the TOE’s log buffer which was at full capacity. The evaluator then performed other commands to generate audit activity and issue the “show logging” command again. The evaluator observed that the new audit records were generated and verified that the oldest audit records had been overwritten.

Test 3: Not applicable. The TOE does not claim FAU_STG_EXT.1/LocSpace.

Test 4: Not applicable. The TOE is not distributed.

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.2.1.1 NDcPP22e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FCS_CKM.1) of the ST identifies RSA schemes that meet FIPS PUB 186-4 and FFC schemes that meet NIST SP 800-56A Revision 3, RFC 3526, and RFC 7919 for asymmetric key generation and key establishment. The TOE uses RSA keys of size 2048 bits or greater in support of device authentication for SSH remote administration and for IPsec sessions. For IPsec, the RSA keys are also used to generate certificate signing requests (CSRs) in which the public key is associated with an X.509 certificate. For key establishment, IPsec uses FFC safe prime groups (DH Group 14) for key establishment, while SSH is mapped to RSA establishment that meets RSAES-PKCS1-v1_5.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all cryptographic protocols defined in the Security Target.

Section “Enabling FIPS Mode” of the Admin Guide provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Administration Protocols” of the Admin Guide provides instructions for generating an RSA 3072 bit key for SSH, configuring the SSH server key exchange algorithm (Diffie-hellman group 14 sha1), configuring the host key and user public key algorithms and configuring ssh rekey settings.

Section “Generate a Key Pair” of the Admin Guide provides instructions for generating an RSA 2048/3072 bit key for use with IPsec.

Section “Securely Connecting to a Certificate Authority for Certificate Signing” of the Admin Guide includes instructions for mapping the RSA key generated for IPsec to a configured trustpoint and for generating a CSR.

Section “IKEv2 Transform Sets” of the Admin Guide provides instructions for configuring dh group 14 in the ikev2 proposal and also for defining an ikev2 keyring and configuring pre-shared keys. The evaluated configuration allows authentication of the IPsec peer using pre-shared keys or X.509 certificates.

Component Testing Assurance Activities: Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

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

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

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

a) Random Primes:

- Provable primes

- Probable primes

b) Primes with Conditions:

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

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

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

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

- Primes q and p shall both be provable primes

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

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

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

- len(q) bit output of RBG where 1 <=x <= q-1

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

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

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

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

Testing for FFC Schemes using safe-prime groups is done as part of testing in CKM.2.1.

(TD0580 applied)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.

2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.2.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. It is sufficient to provide the scheme, SFR, and service in the TSS.

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:

Scheme | SFR | Service

RSA | FCS_TLSS_EXT.1 | Administration

ECDH | FCS_SSHC_EXT.1 | Audit Server

ECDH | FCS_IPSEC_EXT.1 | Authentication Server

The information provided in the example above 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.

(TD0580 applied)

Section 6.1 (FCS_CKM.2) of the ST contains a table mapping the key establishment schemes to each service.

Scheme	SFR	Service
RSA	FCS_SSHS_EXT.1	Remote Administration
FFC/DH	FCS_SSHS_EXT.1	
RSAES-PKCS1	FCS_SSHS_EXT.1	
RSA	FCS_IPSEC_EXT.1	Trusted channel to syslog and RADIUS servers
FFC/DH	FCS_IPSEC_EXT.1	

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

See FCS_CKM.1 where this activity has been performed.

Component Testing Assurance Activities: Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses RSAES-PKCS1-v1_5.

FFC Schemes using 'safe-prime' groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

(TD0580 applied)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1. Gossamer tested using a known implementation for DH14 to ensure correctness for the TOE's use of DH14.

2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.3.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for2). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

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).

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.

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.4.

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.

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

Section 6.1 (FCS_CKM.4) of the ST states the TOE meets all requirements specified in FIPS 140-2 for destruction of keys and Critical Security Parameters (CSPs) when no longer required for use. Section 7 (Annex A: Key Zeroization) of the ST provides a table that identifies all keys, provides a description of each key including where it is stored, and identifies how it is cleared. The ST does not identify any configuration needed to ensure keys are cleared.

Component Guidance Assurance Activities: 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.

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 [Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table)] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

Section “Zeroize Private Key” of the Admin Guide provides instruction for zeroizing private keys stored in NVRAM that are generated for SSH or IPsec using the “crypto key zeroize” command. Other keys stored in SDRAM are zeroized when no longer in use, zeroized with a new value of the key, or zeroized on power-cycle. The Admin Guide does not identify any configurations or circumstances that do not strictly conform to the key destruction requirements or any situation where key destruction may be delayed at the physical layer.

Component Testing Assurance Activities: None Defined

2.2.4 Cryptographic Operation (AES-CMAC Keyed Hash Algorithm) (MACsecEP12:FCS_COP.1(1)/KeyedHashCMAC)

2.2.4.1 MACsecEP12:FCS_COP.1(1).1/KeyedHashCMAC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it specifies the following values used by the AES-CMAC function: key length, hash function used, block size, and output MAC length used.

(TD0466 applied, supersedes TD0134 and TD0357)

Section 6.1 (FCS_COP.1(1)/KeyedHashCMAC Cryptographic Operation (AES-CMAC Keyed Hash Algorithm)) of the ST states the TOE implements AES_CMAC keyed hash function for message authentication as described in NIST SP800-38B. The key length, hash function used, block size, message digest and output MAC length used are as follows:

- AES-128 (hash function and key length)
- Block Sizes: Full (block size)
- Message Length: 0-256 bits (output MAC length)

Component Guidance Assurance Activities: No additional guidance review activities are required. (TD0466 applied, supersedes TD0134 and TD0357)

No additional guidance review activities are required.

Component Testing Assurance Activities: CMAC Generation Test

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 8 arbitrary key-plaintext tuples that will result in the generation of a known MAC value when encrypted. The evaluator will then verify that the correct MAC was generated in each case.

CMAC Verification Test

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 20 arbitrary key-MAC tuples that will result in the generation of known messages when verified. The evaluator will then verify that the correct message was generated in each case.

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:

o 16

o 32

- Message length: values will include the following:

o 0 (optional)

o Largest value supported by the implementation (no greater than 65536)

o Two values divisible by 16

o Two values not divisible by 16

- CMAC length

o Smallest value supported by the implementation (no less than 1)

o 16

o Any supported CMAC length between the minimum and maximum values

(TD0466 applied, supersedes TD0134 and TD0357)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.

2.2.5 Cryptographic Operation (MACsec AES Data Encryption/Decryption) (MACsecEP12:FCS_COP.1(5))

2.2.5.1 MACsecEP12:FCS_COP.1.1(5)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the supported AES modes that are required for this EP in addition to the ones already required by the NDcPP. (TD0466 applied, supersedes TD0134 and TD0357)

Section 6.1 (FCS_COP.1(5) Cryptographic Operation) of the ST states the TOE provides symmetric encryption and decryption capabilities using AES in AES Key Wrap and GCM mode (128 bits) as described in AES as specified in ISO/IEC 18033-3, AES Key Wrap in CMAC mode as specified in NIST SP80-38F, GCM as specified in ISO/IEC 19772. AES is implemented in the MACsec protocol.

Component Guidance Assurance Activities: No additional guidance review activities are required. (TD0466 applied, supersedes TD0134 and TD0357)

No additional guidance review activities are required.

Component Testing Assurance Activities: The evaluator shall perform testing for AES-GCM as required by the NDcPP.

In addition to the tests specified in the NDcPP for this SFR, the evaluator shall perform the following tests for AES Key Wrap in accordance with the NIST 'Key Wrap Validation System':

KW-AE Test

To test the authenticated encryption capability of AES KW, the evaluator shall provide the TSF, for each key length, five sets of 100 messages and keys. Each set of messages and keys shall correspond to one of five plaintext message lengths (detailed below). The evaluator shall have the TSF encrypt the messages with the associated key. The evaluator shall verify that the correct ciphertext was generated in each case.

KW-AD Test

To test the authenticated decryption capability of AES KW, the evaluator shall provide the TSF, for each key length, five sets of 100 ciphertext values and keys. Each set of ciphertexts and keys shall correspond to one of five plaintext message lengths (detailed below). For each set of 100 cyphertext values, 20 shall not be authentic (i.e. fail authentication). The evaluator shall have the TSF decrypt the ciphertext messages with the associated key. The evaluator will then verify the correct plaintext was generated or the failure to authenticate was correctly detected.

The messages in each set for both tests shall be the following lengths:

- two lengths that are non-zero multiples of 128 bits (two semiblock lengths)

- two that are odd multiples of the semiblock length (64 bits)

- the largest supported plaintext length less than or equal to 4096 bits

(TD0466 applied, supersedes TD0134 and TD0357)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.

2.2.6 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS_COP.1/DataEncryption)

2.2.6.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FCS_COP.1/DataEncryption) of the ST states the TOE provides symmetric encryption and decryption capabilities using AES in CBC mode (128 and 256 bits) as described in ISO/IEC 18033-3 and ISO/IEC 10116, respectively. AES is implemented in the SSH and IPsec protocols.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected mode(s) and key size(s) defined in the Security Target supported by the TOE for data encryption/decryption.

Section “Enabling FIPS Mode” of the Admin Guide provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Administration Protocols” of the Admin Guide provides instructions for configuring the supported encryption algorithms used in SSH: aes256-cbc and aes128-cbc.

Section “IKEv2 Transform Sets” of the Admin Guide provides instructions for configuring the supported encryption algorithms (aes-cbc-128 and aes-cbc-256) in the ikev2 proposal.

Section “IPsec Transforms and Lifetimes” of the Admin Guide provides instructions for configuring the IPsec ESP encryption algorithms: esp-aes 128 and esp-aes 256.

Section “MACsec and MKA Configuration” of the Admin Guide provides instructions for configuring the macsec-cipher-suite, gcm-aes-128 in the MKA policy and the cryptographic algorithm, aes-128-cmac in the MACsec PSK keychain.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

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

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

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

CT[1] = AES-CBC-Encrypt(Key, IV, PT)

PT = IV

else:

CT[i] = AES-CBC-Encrypt(Key, PT)

PT = CT[i-1]

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

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

AES-GCM Test

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

128 bit and 256 bit keys

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

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

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

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

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

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-CTR Known Answer Tests

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Due to the fact that Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

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

KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 5 plaintext values for each selected keysize and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros.

KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 5 key values for each selected keysize and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value.

KAT-3 To test the encrypt functionality, the evaluator shall supply a set of key values for each selected keysize as described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values. A set of 128 128-bit keys, a set of 192 192-bit keys, and/or a set of 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N].

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384 ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128].

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10 (test shall be performed using AES-ECB mode). For each i the evaluator shall choose a key and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key using a known good implementation. The evaluator shall perform this test using each selected keysize.

AES-CTR Monte-Carlo Test

The evaluator shall test the encrypt functionality using 100 plaintext/key pairs. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

Input: PT, Key

for i = 1 to 1000:

CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation. The evaluator shall perform this test using each selected keysize.

There is no need to test the decryption engine.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.

2.2.7 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash)

2.2.7.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.1 (FCS_COP.1/Hash) of the ST states that the TOE provides cryptographic hashing services using SHA-1, SHA-256 and SHA-512 as specified in ISO/IEC 10118-3:204 (with key sizes and message digest sizes of 160, 256, and 512 bits respectively).

The TOE provides keyed-hashing message authentication services using HMAC-SHA-1 and HMAC-SHA-256 that operates on 512-bit blocks and HMAC-SHA-512 operating on 1024-bit blocks of data, with key sizes and message

digest sizes of 160-bits, 256 bits and 512 bits respectively as specified in ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”.

For IKE Internet Security Association and Key Management Protocol (ISAKMP) hashing, administrators configure the SHA and message digest to be used with remote IPsec endpoints.

SHA-256 hashing is used for verification of software image integrity. The TOE uses HMAC-SHA1 message authentication as part of the RADIUS Key Wrap functionality.

For IPsec Security Association (SA) authentication integrity options administrators can select any of esp-sha-hmac (HMAC-SHA-1), esp-sha256-hmac (HMAC-SHA-256), or esp-sha512-hmac (HMAC_SHA-512) with message digest sizes of 160 and 256 and 512 bits respectively to be part of the IPsec SA transform-set to be used with remote IPsec endpoints.

Section 6.1 (FCS_SSHS_EXT.1) of the ST indicates that the TOE supports host key authentication with rsa-sha2-256 and rsa-sha2-512 public key algorithms. The TSF’s SSH transport implementation supports the following MAC algorithms: hmac-sha2-256 and hmac-sha2-512. The TSF’s SSH key exchange implementation supports the following key exchange algorithm: diffie-hellman-group14-sha1.

Section 6.1 (FPT_APW_EXT.1) of the ST states that all passwords are stored using a SHA-2 hash.

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

Section “Enabling FIPS Mode” of the Admin Guide provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Administration Protocols” of the Admin Guide provides instructions for configuring the supported key exchange algorithm: diffie-hellman-group14-sha1, the supported MAC algorithms: hmac-sha2-512, hmac-sha2-256, the supported host key algorithms: rsa-sha2-256, rsa-sha2-512 and the supported user/client public key algorithm: ssh-rsa.

Section “IKEv2 Transform Sets” of the Admin Guide provides instructions for configuring the IPsec supported integrity algorithms (sha1, sha256, sha512) in the ikev2 proposal.

Section “IPsec Transforms and Lifetimes” of the Admin Guide provides instructions for configuring the IPsec HMAC algorithms: esp-sha-hmac, esp-sha256-hmac, esp-sha512-hmac.

Section “MACsec and MKA Configuration” of the Admin Guide provides instructions for configuring the macsec-cipher-suite, gcm-aes-128 in the MKA policy and the cryptographic algorithm, aes-128-cmac in the MACsec PSK keychain.

Component Testing Assurance Activities: The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number

of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.

2.2.8 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS_COP.1/KeyedHash)

2.2.8.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FCS_COP.1/KeyedHash) of the ST states the TOE provides keyed-hashing message authentication services using HMAC-SHA-1 and HMAC-SHA-256 that operates on 512-bit blocks and HMAC-SHA-512 operating on 1024-bit blocks of data, with key sizes and message digest sizes of 160-bits, 256 bits and 512 bits respectively as specified in ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”.

For IPsec Security Association (SA) authentication integrity options administrators can select any of esp-sha-hmac (HMAC-SHA-1), esp-sha256-hmac (HMAC-SHA-256), or esp-sha512-hmac (HMAC_SHA-512) with message digest sizes of 160 and 256 and 512 bits respectively to be part of the IPsec SA transform-set to be used with remote IPsec endpoints.

Section 6.1 (FCS_SSHS_EXT.1) of the ST states that the TSF’s SSH transport implementation supports the following MAC algorithms: hmac-sha2-256, hmac-sha2-512.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the values used by the HMAC function: key length, hash function used, block size, and output MAC length used defined in the Security Target supported by the TOE for keyed hash function.

Section “Enabling FIPS Mode” of the Admin Guide provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Administration Protocols” of the Admin Guide provides instructions for configuring the supported MAC algorithms: hmac-sha2-256, hmac-sha2-512.

Section “IPsec Transforms and Lifetimes” of the Admin Guide provides instructions for configuring the IPsec HMAC algorithms: esp-sha-hmac, esp-sha256-hmac, esp-sha512-hmac.

Section “MACsec and MKA Configuration” of the Admin Guide provides instructions for configuring the macsec-cipher-suite, gcm-aes-128 in the MKA policy and the cryptographic algorithm, aes-128-cmac in the MACsec PSK keychain.

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.

2.2.9 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS_COP.1/SigGen)

2.2.9.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and key size supported by the TOE for signature services.

Section 6.1 (FCS_COP.1/SigGen) of the ST states the TOE provides cryptographic signature services using an RSA Digital Signature Algorithm with key size of 2048 or 3072 as specified in FIPS PUB 186-4.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected cryptographic algorithm and key size defined in the Security Target supported by the TOE for signature services.

Section “Enabling FIPS Mode” of the Admin Guide provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enables in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Administration Protocols” of the Admin Guide provides instructions for generating an RSA 3072 bit key for use with SSH remote administration.

Section “Generate a Key Pair” of the Admin Guide provides instructions for generating an RSA 3072 bit key for use with IPsec.

Section “Creation of the Certificate Signing Request” of the Admin Guide includes instructions for mapping the RSA key generated for IPsec to a configured trustpoint, importing the CA certificate and generating a certificate signing request.

Section “MACsec and MKA Configuration” of the Admin Guide provides instructions for configuring pre-shared keys for use with MACsec.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

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

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

Signature Verification Test

For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (d, e). Each private key d is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, e, messages, or signatures are altered so that signature verification should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key e values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.

2.2.10 IPsec Protocol - per TD0633 (NDcPP22e:FCS_IPSEC_EXT.1)

2.2.10.1 NDcPP22e:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: 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.

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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states when a packet matches a permit entry in a particular access list, and the corresponding crypto map entry is tagged connections are established, if necessary. If the crypto map entry is tagged as ipsec-isakmp, IPsec is triggered. If there is no SA that the IPsec can use to protect this traffic to the peer, IPsec uses IKE to negotiate with the remote peer to set up the necessary IPsec SAs on behalf of the data flow. The negotiation uses information specified in the crypto map entry as well as the data flow information from the specific access list entry.

Once established, the set of SAs (outbound to the peer) is then applied to the triggering packet and to subsequent applicable packets as those packets exit the Controller. “Applicable” packets are packets that match the same access list criteria that the original packet matched. For example, all applicable packets could be encrypted before being forwarded to the remote peer. The corresponding inbound SAs are used when processing the incoming traffic from that peer.

Access lists associated with IPsec crypto map entries also represent the traffic that the Controller needs protected by IPsec. Inbound traffic is processed against crypto map entries. If an unprotected packet matches a permit entry in a particular access list associated with an IPsec crypto map entry, that packet is dropped because it was not sent as an IPsec-protected packet. The traffic matching the permit ACLs would then flow through the IPsec tunnel and be classified as “PROTECTED”. Traffic that does not match a permit ACL in the crypto map, but that is not disallowed by other ACLs on the interface is allowed to BYPASS the tunnel. Traffic that does not match a permit ACL and is also blocked by other non-crypto ACLs on the interface would be DISCARDED. Rules applied to an access control list can be applied to either inbound or outbound traffic.

Guidance Assurance Activities: 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 â€“ 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.

Section “Information Flow Policies” of the Admin Guide provides instructions for configuring access lists and applying these access lists to interfaces using crypto map sets. When a packet matches a permit entry in a particular access list, the method of security in the corresponding crypto map is applied. If the crypto map entry is tagged as ipsec-isakmp, IPsec is triggered and packets will be encrypted/decrypted (Protect). This section also provides examples of how to configure and apply the access lists such that specific traffic will be dropped (Discard) of will flow through unencrypted (Bypass).

Testing Assurance Activities: The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

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

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

Test 1: The TOE uses access-lists in order to control the flow of traffic. If traffic is destined for an IP address associated with the IPsec tunnel, then the traffic will be protected. Otherwise, the access-lists are screened in order to determine whether the traffic is allowed through unprotected or dropped. The evaluator sent a series of packets to the TOE. The evaluator sent packets to the IPsec tunnel interface that matched the rules to be encrypted (Protect) and observed that the traffic was encrypted by IPsec. The evaluator then sent the packets that did not match IPsec traffic selectors but which matched permit rules on the access list applied to the interface and observed that the traffic was sent in plaintext (Bypass). The evaluator then sent packets that matched rules to be denied and observed that the packets were dropped (Discard). The evaluator also sent packets that did not match any of the specific rules and observed that they were denied by default.

Test 2: To test the priority behavior of the SPD rules the evaluator configured the access list on the interface to discard packets on the subnet first followed by a bypass rule for the specific host. The evaluator sent packets from the defined host and viewed that they were denied as that rule took priority. The evaluator then flipped the priorities of the access list on the interface so the specific bypass rule was first. The evaluator sent packets from the defined host and viewed that they were accepted in plaintext as that rule took priority.

2.2.10.2 NDcPP22e:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

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

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

This test was performed as part of NDcPP22e:FCS_IPSEC_EXT.1.1-t1, which demonstrates the TOE protecting a packet, discarding a packet, allowing the packet to bypass the tunnel, and dropping the packet if a packet field is modified and does not match any rule.

2.2.10.3 NDcPP22e:FCS_IPSEC_EXT.1.3

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

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states the TOE supports both transport and tunnel mode for IPsec communications between the TOE and an external audit and authentication server. This is consistent with FCS_IPSEC_EXT.1.3.

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

Section “IPsec Transforms and Lifetimes” of the Admin Guide explains how to configure the connection in both transport mode and tunnel mode.

Testing Assurance Activities: The evaluator shall perform the following test(s) based on the selections chosen:

Test 1: If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN

peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

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

Test 1 and Test 2: The TOE supports both tunnel mode and transport mode. The evaluator configured a test peer to require only tunnel mode or only transport mode. In each case, the evaluator then attempted to establish an IPsec connection between the test peer and the TOE and observed via logs and packet captures that the connection was successful in each case.

2.2.10.4 NDcPP22e:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4)/KeyedHash Cryptographic Operations (for keyed-hash message authentication) and if the SHA-based HMAC function truncated output is utilized it must also be described.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states IPsec protocol ESP is implemented using the cryptographic algorithms AES-CBC-128, and AES-CBC-256 (specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC-SHA1, HMAC-SHA-256, and HMAC-SHA-512.

Guidance Assurance Activities: The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

Section “Remote Administration Protocols” of the Admin Guide explains how to configure the TOE to use AES-CBC-128 and AES-CBC-256 encryption key algorithms to secure and control SSH sessions.

Testing Assurance Activities: 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.

The evaluator configured the TOE to use AES-CBC-128 and AES-CBC-256, and verified via logs and packet captures that the connection was successfully established for each algorithm.

2.2.10.5 NDcPP22e:FCS_IPSEC_EXT.1.5

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

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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states the TOE supports IKEv2 session establishment. This is consistent with the requirement which only claims IKEv2.

Guidance Assurance Activities: 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 for the following test (if selected).

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.

Section "IKEv2 Transform Sets" of the Admin Guide explains how to configure the IPsec with IKEv2 functionality for the TOE. The TOE does not support NAT traversal.

Testing Assurance Activities: Tests are performed in conjunction with the other IPsec evaluation activities.

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

b) Test 2: If NAT traversal is selected within the IKEv2 selection, the evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

Test 1: Not applicable. IKEv1 is not selected.

Test 2: Not applicable. NAT traversal is not selected within the IKEv2 selection.

2.2.10.6 NDcPP22e:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: 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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states that the TSF's implementation of the IPsec standard (in accordance with the RFCs noted in the SFR) uses the Encapsulating Security Payload (ESP) protocol to provide authentication and encryption supporting AES-CBC-128 (RFC3602), AES-CBC-256 (RFC3602), and with a SHA-based HMAC (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512) to implement IKEv2 and the IPsec protocol ESP as defined in RFC 4303.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement.

Section “IKEv2 Transform Sets” of the Admin Guide provides instructions for configuring the encryption algorithms, aes-cbc-128 and aes-cbc-256, in the IKEv2 proposal.

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

The evaluator configured IKEv2 profiles (AES-CBC-128 and AES-CBC-256) on the TOE. The evaluator then confirmed that the TOE could establish a session with each algorithm and that the tunnel successfully established with the selected algorithm.

2.2.10.7 NDcPP22e:FCS_IPSEC_EXT.1.7

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

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states the TOE supports configuration of session lifetimes for both Phase 1 SAs and Phase 2 SAs using the following command “lifetime.” The time values for Phase 1 SAs can be limited from 2 minutes up to 24 hours and for Phase 2 SAs up to 8 hours. The Phase 2 SA lifetimes can also be configured by an Administrator based on number of bytes. The TOE supports Diffie-Hellman Group 14.

Guidance Assurance Activities: 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.

Section “IKEv2 Transform Sets” of the Admin Guide provides instructions for configuring the lifetime in seconds in the IKEv2 profile. This includes the ability to configure the values from 120 to 86400 seconds which is equivalent to the values (2 minutes to 24 hours) specified in the requirement.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that

both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

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.

Test 1: Not applicable. The TOE does not support data size based rekey limits for the IKEv2 SA.

Test 2: For this test, the evaluator configured the TOE to have a 24 hour IKE and 8 hour ESP limits. The evaluator established an IPsec connection between the ESS3300 instance of the TOE and the test server and then waited for over 24 hours before terminating the test. The evaluator observed through logs and packet captures that the connection was successful and that the TOE rekeyed well before the configured time limits were reached, ensuring that the keys were renegotiated successfully and there was no data loss during the rekey. The evaluator then configured the TOE to have a 1 hour IKE and 30 minute ESP limits. The evaluator established an IPsec connection between the ESS9300 instance of the TOE and the test server and then waited for over 1 hour before terminating the test. The evaluator observed through logs and packet captures that the connection was successful and that the TOE rekeyed well before the configured time limits were reached, ensuring that the keys were renegotiated successfully and there was no data loss during the rekey.

2.2.10.8 NDcPP22e:FCS_IPSEC_EXT.1.8

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

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states that the TOE supports configuration of session lifetimes for both Phase 1 SAs and Phase 2 SAs using the following command “lifetime.” The time values for Phase 1 SAs can be limited from 2 minutes up to 24 hours and for Phase 2 SAs up to 8 hours. The Phase 2 SA lifetimes can also be configured by an Administrator based on number of bytes. The TOE supports Diffie-Hellman Group 14.

Guidance Assurance Activities: 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.

Section “IPsec Transforms and Lifetimes” of the Admin Guide provides instructions for configuring the IPsec security association lifetime, which can be configured based on time in seconds. This includes the ability to configure the values from 120 to 28800 seconds, which is equivalent to the values (2 minutes to 8 hours) specified in the requirement. Instructions are also included for configuring the IPsec security association lifetime by volume in kilobytes with a range of 2560-4294967295.

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

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

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.

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.

Test 1: The evaluator configured a max lifetime of 2560 KB on the TOE and established a connection with a test peer. The Phase 2 SA timed out after the packet size was exceeded and the TOE rekeyed the negotiation successfully.

Test 2: This test was performed as part of the FCS_IPSEC_EXT.1.7 test 2 where the evaluator observed that the TOE rekeyed before the configured time limits for the Phase 1 SA and the Phase 2 SA.

2.2.10.9 NDcPP22e:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: 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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states the TSF generates the secret value 'x' used In the IKEv2 Diffie-Hellman key exchange ("x" in g^x mod p) using the NIST approved DRBG specified in FCS_RBG_EXT.1 and having possible length of 256 or 384 bits. The TOE generates nonces used in IKEv2 exchanges, of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash. When a random number is needed for a nonce, the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^{128}. The nonce is likewise generated using the AES-CTR DRBG.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.10 NDcPP22e:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: 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.

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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states the TSF generates the secret value 'x' used In the IKEv2 Diffie-Hellman key exchange ("x" in g^x mod p) using the NIST approved DRBG specified in FCS_RBG_EXT.1 and having possible length of 224 bits. The TOE generates nonces used in IKEv2 exchanges, of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash. When a random number is needed for a nonce, the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^{128}. The nonce is likewise generated using the AES-CTR DRBG.

Guidance Assurance Activities: None Defined

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

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

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

See TSS assurance activity above.

2.2.10.11 NDcPP22e:FCS_IPSEC_EXT.1.11

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

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states the TOE supports Diffie-Hellman Group 14.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement.

Section "IKEv2 Transform Sets" of the Admin Guide explains how to select DH Group 14 (2048-bit MODP) for IKE.

Testing Assurance Activities: 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.

The evaluator made a successful IPsec connection to an IPsec peer using the claimed DH group 14. The evaluator was able to capture DH group 14 using a packet capture to ensure the correct DH group was used.

2.2.10.12 NDcPP22e:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: 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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states that the resulting potential strength of the symmetric key will be 128 or 256 bits of security depending on the algorithms negotiated between the two IPsec peers. As part of this negotiation, the TOE verifies that the negotiated phase 2 symmetric algorithm key strength is at most as large as the negotiated phase 1 key strength as configured on the TOE and peer via an explicit check.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator simply follows the guidance to configure the TOE to perform the following tests.

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

b) Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish 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.

Test 1: The evaluator made an IPsec connection to an IPsec peer using each of the claimed hash functions identified in the requirements. The evaluator verified via packet capture and logs that the connection was successful with each of the claimed functions.

Test 2: This test is met by guidance in the Admin Guide which instructs the user not to configure an ESP encryption algorithm with greater strength than the algorithm configured for the IKEv2 SA. The evaluator configured the TOE as instructed and then attempted to establish a connection with a test server configured with an ESP algorithm with greater strength than the IKEv2 SA algorithm and observed that the attempt failed.

Test 3: The evaluator attempted to establish a connection with an unsupported algorithm/hash combination. The connection attempt failed.

Test 4: This test was performed as part of Test 3 above.

2.2.10.13 NDcPP22e:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(2)/SigGen Cryptographic Operations (for cryptographic signature).

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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states that the TOE supports authentication of IPsec peers using pre-shared keys, and RSA X.509 certificates. During IKE establishment, IPsec peers authenticate each other by creating and exchanging a hash value that includes the pre-shared key. The TOE will compare the received hash value to its computed hash and determine if it matches. If it does, pre-shared key authentication is successful; otherwise, pre-shared key authentication fails. For peer authentication using RSA certificates, the TOE validates the presented identifier provided supporting the following fields and types: SAN: IP address, SAN: Fully Qualified Domain Name (FQDN).

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

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.

The evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA and ensure a valid certificate for that CA is loaded into the TOE and marked 'trusted'.

Section “IPsec Overview” of the Admin Guide explains how to set up the TOE to use certificates with RSA signatures and public keys.

Section “IKEv2 Transform Sets” of the Admin Guide explains how to configure IPsec to use pre-shared keys and states that X.509 v3 certificates are also supported for authentication of IPsec peers. In IKEv2 these pre-shared keys are specific to the peer. Pre-shared keys on the TOE must be at least 22 characters in length and can be composed of any combination of upper and lower case letters, numbers, and special characters (that include: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”). The TOE supports pre-shared keys up to 128 bytes in length. While longer keys increase the difficulty of brute-force attacks, but longer keys increase processing time.

Section “Configuring Certificate Chain Validation” of the Admin Guide explains 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’.

Testing Assurance Activities: For efficiency sake, the testing is combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1.

This testing is combined and performed as part of testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), FCS_IPSEC_EXT.1.1, and FIA_PSK_EXT.1 where IPsec connections were successfully established using pre-shared keys and RSA certificates.

2.2.10.14 NDcPP22e:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: 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.

Section 6.1 (FCS_IPSEC_EXT.1) of the ST states for peer authentication using RSA certificates, the TOE validates the presented identifier provided supporting the following fields and types: SAN: IP address, SAN: Fully Qualified Domain Name (FQDN).

Certificate maps provide the ability for a certificate to be matched with a given set of criteria. The Administrator is instructed in the CC Configuration Guide to specify one or more certificate fields together with their matching criteria and the value to match. In the evaluated configuration, the field name must specify the SAN (alt-subject-name) field. Match criteria should be "eq" for equal. SAN example: alt-subject-name eq <peer.cisco.com>.

The TOE will reject the IKE connection in any of these situations: 1) If the data ID Payload for any of those ID Types does not match the peer's certificate exactly; 2) If an ID Payload is not provided by the peer; 3) If multiple ID Types are provided in the ID Payload.

Guidance Assurance Activities: 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.

Section "Configure Reference Identifier" of the Admin Guide provides instructions for configuration of the peer reference identifier through use of a certificate map. Certificate maps provide the ability for a certificate to be matched with a given set of criteria. You can specify which fields within a certificate should be checked and which values those fields may or may not have. The certificate map is associated with the IPsec trustpoint. This section further states that in the evaluated configuration, the field name must specify the SAN field of the peer's certificate. This includes specifying either the FQDN of the peer or the IP address of the peer in the SAN.

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

The evaluator shall perform the following tests:

Test 1: (conditional) For each CN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes CN checking over SAN (through explicit configuration of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the SAN so it contains an incorrect identifier of the correct type (e.g. the reference identifier on the TOE is example.com, the CN=example.com, and the SAN:FQDN=otherdomain.com) and verify that IKE authentication succeeds.

Test 2: (conditional) For each SAN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes SAN checking over CN (through explicit specification of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the CN so it contains an incorrect identifier formatted to be the same type (e.g. the

reference identifier on the TOE is DNS-ID; identify certificate has an identifier in SAN with correct DNS-ID, CN with incorrect DNS-ID (and not a different type of identifier)) and verify that IKE authentication succeeds.

Test 3: (conditional) For each CN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with the CN so it contains the valid identifier followed by ''. 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.

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

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

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

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

Test 5: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 6: (conditional) If the TOE supports DN identifier types, to demonstrate a bit-wise comparison of the DN, the evaluator shall create the following valid certificates and verify that the IKE authentication fails when each certificate is presented to the TOE:

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

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

Test 1: Not applicable. The TOE does not support CN identifiers.

Test 2: For the first part of this test, the evaluator alternately configured a test peer to use an authentication certificate with the correct SAN: IP address and SAN: FQDN. The evaluator then attempted to establish an IPsec connection between the TOE and the test peer and confirmed that the connection was successful. CN identifiers are not supported by the TOE so the second part of the test is not applicable.

Test 3: Not applicable. The TOE does not support CN identifier types.

Test 4: For this test, the evaluator alternately configured the TOE to look for each of the supported SAN reference identifiers (SAN: IP Address and SAN: FQDN). The evaluator then configured the test peer to use a certificate that would present an incorrect SAN reference identifier and a correct CN reference identifier as CN checking is not prioritized over SAN (CN is not supported). In each case, the evaluator attempted to establish an IPsec connection to the TOE and then expected the TOE to reject the connection.

Test 5: Not applicable. DN identifier types are not supported.

Test 6: Not applicable. The TOE does not support DN and CN identifier types.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.11 MACsec (MACsecEP12:FCS_MACSEC_EXT.1)

2.2.11.1 MACsecEP12:FCS_MACSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 MACsecEP12:FCS_MACSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.3 MACsecEP12:FCS_MACSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.4 MACsecEP12:FCS_MACSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the ability of the TSF to implement MACsec in accordance with IEEE 802.1AE-2006. The evaluator shall also determine that the TSS describes the ability of the TSF to derive SCI values from peer MAC address and port data and to reject traffic that

does not have a valid SCI. Finally, the evaluator shall check the TSS for an assertion that only EAPOL, MACsec Ethernet frames, and MAC control frames are accepted by the MACsec interface. (per TD0553)

Section 6.1 (FCS_MACSEC_EXT.1) of the ST states that the TOE implements MACsec in compliance with Institute of Electrical and Electronics Engineers (IEEE) Standard 802.1AE-2006. The MACsec connections maintain confidentiality of transmitted data and takes measures against frames transmitted or modified by unauthorized devices. In addition, the TOE implementation provides configuration options and management of the MACsec functionality.

The Secure Channel Identifier (SCI) is composed of a globally unique 48-bit Message Authentication Code (MAC) address and the Secure System Address (port). The SCI is part of the SecTAG if the Secure Channel (SC) bit is set and will be at the end of the tag. Any MAC Protocol Data Units (MPDUs) during a given session that contain an SCI other than the one used to establish that session is rejected.

Only Extensible Authentication Protocol over LAN EAPOL (Physical Address Extension (PAE) EtherType 88-8E), MACsec frames (EtherTYpe 88-E5) and MACsec control frames (EtherType 88-08) are permitted. All others are rejected.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test 1: The evaluator shall successfully establish a MACsec channel between the TOE and a MACsec-capable peer in the Operational Environment and verify that the TSF logs the communications. The evaluator shall capture the traffic between the TOE and the Operational Environment to determine the SCI that the TOE uses to identify the peer. The evaluator shall then configure a test system to capture traffic between the peer and the TOE to modify the SCI that is used to identify the peer. The evaluator then verifies that the TOE does not reply to this traffic and logs that the traffic was discarded.

Test 2: The evaluator shall send Ethernet traffic to the TOE's MAC address that iterates through the full range of supported EtherType values (refer to http://standards.ieee.org/develop/regauth/ethertype/eth.txt) and observes that traffic for all EtherType values is discarded by the TOE except for the traffic which has an EtherType value of 88-8E, 88-E5 or 8808. Note that there are a large number of EtherType values so the evaluator is encouraged to execute a script that automatically iterates through each value.

(per TD0553)

Test 1: The evaluator configured MACsec between the TOE and a peer. The evaluator captured the SCI value that is used to negotiate the successful connection. The evaluator sent a MACsec encrypted ping packet from the peer to the TOE. The TOE responded with a MACsec reply, indicating that it accepted the peer's MACsec packet. The evaluator then sent a packet with an incorrect SCI value. The evaluator observed that the TOE rejected the incorrect packet and did not send a reply.

Test 2: The evaluator configured MACsec between the TOE and a peer. The evaluator then configured the peer test device to send Ethernet traffic that iterates through the range of supported EtherType values. The evaluator

confirmed from packet captures that the TOE does not respond to any EtherTypes except for 88-8E or 8-E5. MACsec and EAPOL EtherTypes can be seen throughout other MACsec tests.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.12 MACsec Integrity and Confidentiality (MACsecEP12:FCS_MACSEC_EXT.2)

2.2.12.1 MACsecEP12:FCS_MACSEC_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 MACsecEP12:FCS_MACSEC_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.3 MACsecEP12:FCS_MACSEC_EXT.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the methods that the TOE implements to provide assurance of MACsec integrity, including the confidentiality offset(s) used, the use of an ICV (including the supported length), and generating the ICV with the SAK, using the SCI as the most significant bits of the IV and the 32 least significant bits of the PN as the IV.

Section 6.1 (FCS_MACSEC_EXT.2) of the ST states the TOE implements the MACsec requirement for integrity protection with the confidentiality offsets of 0, 30 and 50 through the CLI command of “mka policy <policyname>, confidentiality-offset” commands. An offset value of 0 does not offset the encryption and offset values of 30 and 50 offset the encryption by 30 and 50 characters respectively.

An Integrity Check Value (ICV) that is 16 bytes in length is derived with the Secure Association Key (SAK) and is used to provide assurance of the integrity of MPDUs. The TOE derives the ICV from a CAK using KDF, using the SCI as the most significant bits of the Initialization Vector (IV) and the 32 least significant bits of the PN as the IV.

Component Guidance Assurance Activities: If any integrity verifications are configurable such as the confidentiality offset(s) used or the mechanism used to derive an ICK, the evaluator shall verify that instructions for performing these functions are documented.

Section "MACsec and MKA Configuration" of the Admin Guide provides instructions for configuring the "confidentiality-offset" in the MKA policy. It also provides instructions for configuring the MACsec PSK which includes configuring the AES-CMAC algorithm and the key-string. The key-string is the CAK that is used for ICV validation by the MKA protocol. The CAK is not used directly, but derives two further keys from the CAK using the AES cipher in CMAC mode. The derived keys include the ICV Key (ICK) used to verify the integrity of MPDUs and to prove the transmitter of the MKPDU possesses the CAK.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall transmit MACsec traffic to the TOE from a MACsec-capable peer in the Operational Environment. The evaluator shall verify via packet captures and/or audit logs that the frame bytes after the MACsec Tag values in the received traffic is not obviously predictable.

Test 2: The evaluator shall transmit valid MACsec traffic to the TOE from a MACsec-capable peer in the operational environment that is routed through a test system set up as a man-in-the-middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure.

Test 1: A valid test MACsec connection was performed in MACsecEP12:FCS_MACSEC_EXT.1-t1. The evaluator viewed that none of the frame bytes after MACsec Tag value were obviously predictable, indicating that the data was successfully encrypted.

Test 2: The evaluator configured MACsec between the TOE and a peer. The evaluator attempted to send a MACsec encrypted ICMP packet in which the encrypted packet contained a modified byte in the end of the data payload. This effectively creates an invalid packet in which the ICV does not match the entire packet. The evaluator observed that the TOE rejected the incorrect packet and did not send a reply.

2.2.13 MACsec Randomness (MACsecEP12:FCS_MACSEC_EXT.3)

2.2.13.1 MACsecEP12:FCS_MACSEC_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.2 MACsecEP12:FCS_MACSEC_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the method used to generate SAKs and nonces and that the strength of the CAK and the size of the CAK's key space are provided.

Section 6.1 (FCS_MACSEC_EXT.3) of the ST states each SAK is generated using the KDF specified in IEEE 802.1X-2010 section 6.2.1 using the following transform–- KS-nonce = a nonce of the same size as the required SAK, obtained from an RNG each time an SAK is generated.

The CAK is based on AES cipher in CMAC mode, with key sizes of 128 and 256 bits. Each of the keys used by MKA is derived from the CAK. The key string is the CAK that is used for ICV validation by the MKA protocol. The CAK is not used directly but derives two further keys from the CAK using the AES cipher in CMAC mode.

The derived keys, which are derived via key derivation function as defined in SP800-108 KDF (CMAC) are tied to the identity of the CAK, and thus restricted to use with that particular CAK. These are the ICV Key (ICK) used to verify the integrity of MPDUs and to prove that the transmitter of the MKPDU possesses the CAK, and the Key Encrypting Key (KEK) used by the Key Server, elected by MKA, to transport a succession of SAKs, for use by MACsec, to the other member(s) of a CA.

The size of the key is based on the configured AES key sized used. If using AES 128-bit CMAC mode encryption, the key string will be 32-bit hexadecimal in length. If using 256-bit encryption, the key string will be 64-bit hexadecimal in length.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Testing of the TOE's MACsec capabilities and verification of the DRBG is sufficient to demonstrate that this SFR has been satisfied.

Please see test results for other MACsec test cases, which show the TOE’s MACsec capabilities. Please see FCS_RBG_EXT.1 for verification of the DRBG.

2.2.14 MACsec Key Usage (MACsecEP12:FCS_MACSEC_EXT.4)

2.2.14.1 MACsecEP12:FCS_MACSEC_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.2 MACsecEP12:FCS_MACSEC_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.3 MACsecEP12:FCS_MACSEC_EXT.4.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.4 MACsecEP12:FCS_MACSEC_EXT.4.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.5 MACsecEP12:FCS_MACSEC_EXT.4.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the SAK is wrapped prior to being distributed using the AES implementation specified in this EP.

Section 6.1 (FCS_MACSEC_EXT.4) of the ST states MACsec peer authentication is achieved only using pre-shared keys. The SAKs are distributed between these peers using AES Key Wrap. Prior to distribution of the SAKs between these peers, the TOE uses AES Key Wrap GCM with a key size of 128 or 256 bits in accordance with AES as specified in ISO 18033-3, AES Key Wrap in CMAC mode as specified in NIST SP 800-38F, and GCM as specified in ISO 19772.

Component Guidance Assurance Activities: If the method(s) of peer authentication is configurable, the evaluator shall verify that the guidance provides instructions on how to configure this. The evaluator shall also verify that the method of specifying a lifetime for CAKs is described.

Section “MACsec and MKA Configuration” of the Admin Guide provides instructions for configuring the MACsec PSK in a key chain and includes specifying the key-string and the lifetime for CAKs.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: For each supported method of peer authentication in FCS_MACSEC_EXT.4.1, the evaluator shall follow the operational guidance to configure the supported method (if applicable). The evaluator shall set up a packet sniffer between the TOE and a MACsec-capable peer in the Operational Environment. The evaluator shall then initiate a connection between the TOE and the peer such that authentication occurs and a secure connection is established. The evaluator shall wait 1 minute and then disconnect the TOE from the peer and stop the sniffer. The evaluator shall use the packet captures to verify that the secure channel was established via the selected mechanism and that the EtherType of the first data frame sent between the TOE and the peer is 88-E5.

Test 2: The evaluator shall capture traffic between the TOE and a MACsec-capable peer in the Operational Environment. The evaluator shall then cause the TOE to distribute a SAK to that peer, capture the MKPDUs from that operation, and verify the key is wrapped in the captured MKPDUs. (TD0273 applied)

Test 3: Removed by TD0273 (duplicate of FMT_SMF.1 Test 3).

Test 1: The evaluator configured MACsec between the TOE and a peer. The evaluator started a packet capture and then stopped the packet capture after 1 minute. The evaluator verified that the first data frame sent between the TOE and the test peer has an EtherType value of x088E5. At the same time, the evaluator ensured that the TOE is the MKA server by ensuring that the test peer's MKA priority is set to the maximum value (255) and that the TOE's MKA priority is set to a higher priority. The evaluator then analyzed the packet capture and verified that the TOE sends an MKPDU packet to the test peer that contains an AES wrapped SAK.

Test 2: This test was performed as part of Test 1.

2.2.15 MACsec Key Agreement (MACsecEP12:FCS_MKA_EXT.1)

2.2.15.1 MACsecEP12:FCS_MKA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.2 MACsecEP12:FCS_MKA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The test below requires the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with. Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the Key Server and principal actor. The evaluator shall then perform the following test:

The evaluator shall use a peer device to send traffic to the TOE, arbitrarily inducing artificial delays in their transmission using a man-in-the-middle setup. The evaluator shall observe that traffic delayed longer than 2.0 seconds is rejected.

The evaluator enabled data delay protection in the TOE in order to execute this test. The evaluator configured MACsec between the TOE and a peer and initiated a connection. The evaluator then introduced a packet delay from the MACsec peer using the Linux tc command. The evaluator reviewed the packet capture and saw several packets that were delayed by at least 3 seconds. The evaluator observed the TOE rejecting the traffic and issuing an audit message of the rejection.

2.2.15.3 MACsecEP12:FCS_MKA_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.4 MACsecEP12:FCS_MKA_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the methods that the TOE implements to provide assurance of MKA integrity, including the use of an ICV and the ability to use a KDF to derive an ICK.

Section 6.1 (FCS_MKA_EXT.1) of the ST states the TOE implements Key Agreement Protocol (MKA) in accordance with IEEE 802.1X-2010 and 802.1Xbx-2014. The data delay protection is enabled for MKA as a protection guard against an attack on the configuration protocols that MACsec is designed to protect by alternately delaying and delivering their PDUs. The Delay protection does not operate if and when MKA operation is suspended. An MKA Lifetime Timeout limit of 6.0 seconds and Hello Timeout limit of 2.0 seconds is enforced by the TOE.

The TOE discards MKPDUs that do not satisfy the requirements listed under FCS_MKA_EXT.1.8. All valid MKPDUs that meet the requirements as defined under FCS_MKA_EXT.1.8 are decoded in a manner conformant to IEEE 802.1x-2010 Section 11.11.4.

On successful peer authentication, a connectivity association is formed between the peers and a secure Connectivity Association Key Name (CKN) is exchanged. After the exchange, the MKA ICV is validated with a Connectivity Association Key (CAK), which is effectively a secret key.

For the Data Integrity Check, MACsec uses MKA to generate an Integrity Check Value (ICV) for the frame arriving on the port. If the generated ICV is the same as the ICV in the frame, then the frame is accepted; otherwise, it is dropped. The key string is the Connectivity Association Key (CAK) that is used for ICV validation by the MKA protocol.

Section 6.1 (FCS_MACSEC_EXT.2) of the ST states that the TOE derives the ICV from a CAK using KDF.

Section 6.1 (FCS_MACSEC_EXT.3) of the ST states that each of the keys used by MKA is derived from the CAK. The key string is the CAK that is used for ICV validation by the MKA protocol. The CAK is not used directly but derives two further keys from the CAK using the AES cipher in CMAC mode. The derived keys, which are derived via key derivation function as defined in SP800-108 KDF (CMAC), are tied to the identity of the CAK, and thus restricted to use with that particular CAK. These are the ICV Key (ICK) used to verify the integrity of MPDUs and to prove that the transmitter of the MKPDU possesses the CAK, and the Key Encrypting Key (KEK) used by the Key Server, elected by MKA, to transport a succession of SAKs, for use by MACsec, to the other member(s) of a CA.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall transmit MKA traffic (MKPDUs) to the TOE from an MKA-capable peer in the Operational Environment. The evaluator shall verify via packet captures and/or audit logs that the last 16 octets of the MKPDUs in the received traffic do not appear to be predictable.

Test 2: The evaluator shall transmit valid MKA traffic to the TOE from an MKA-capable peer in the operational environment that is routed through a test system set up as a man-in-the-middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure.

Test 1: A successful MACsec connection was established in MACsecEP12:FCS_MACSEC_EXT.1-t1 above. The evaluator observed the ICV in the MKPDU packets and determined they were not obviously predictable.

Test 2: The evaluator disabled data replay protection in the TOE in order to execute this test. The evaluator configured MACsec between the TOE and a peer. The evaluator captured a previously sent MKPDU packet and modified the last byte in the packet. The evaluator attempted to send the modified packet to the TOE. The evaluator observed the TOE successfully rejecting the packet and reporting an error in the audit log.

2.2.15.5 MACsecEP12:FCS_MKA_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The tests below require the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with.

Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the Key Server and principal actor (peer). The evaluator shall then perform the following tests using a traffic sniffer to capture this traffic.

Test 1: The evaluator shall send a fresh SAK that includes both peers as active participants. The evaluator shall start an MKA session between the TOE and the two active participant peers and send MKPDUs. The evaluator shall verify from packet captures that MKPDUs are sent at least once every half-second.

Test 2: Disconnect one of the peers. Using a man-in-the-middle device, arbitrarily introduce an artificial delay in sending a fresh SAK following the change in the Live Peer List. Repeat Test 1 delaying a fresh SAK for MKA Lifetime traffic and observe that the timeout of 6.0 seconds is enforced by the TSF.

Test 1: The TOE was configured with delay protection enabled. The evaluator set up two MACsec peers to connect to the TOE. The evaluator ensured that the TOE is the Key Server by setting an MKA priority that is lower than the two MACsec peers. The evaluator then started an MKA session between the TOE and the two active participant peers, also taking a packet capture of the session. The evaluator analyzed the packet capture and noted that the TOE sends MKPDUs at least once every half-second.

Test 2: Not applicable as the TOE does not support Group CAKs.

2.2.15.6 MACsecEP12:FCS_MKA_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.7 MACsecEP12:FCS_MKA_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.8 MACsecEP12:FCS_MKA_EXT.1.8

TSS Assurance Activities: The evaluator shall verify that the TSS describes the TOE's compliance with IEEE 802.1X-2010 and 802.1Xbx-2014 for MKA, including the values for MKA and Hello timeout limits and support for data delay protection. The evaluator shall also verify that the TSS describes the ability of the PAE of the TOE to establish unique CAs with individual peers and group CAs using a group CAK such that a new group SAK is distributed every time the group's membership changes. The evaluator shall also verify that the TSS describes the invalid MKPDUs that are discarded automatically by the TSF in a manner that is consistent with the SFR, and that valid MKPDUs are decoded in a manner consistent with IEEE 802.1X-2010 section 11.11.4.

Section 6.1 (FCS_MKA_EXT.1) of the ST states the TOE implements Key Agreement Protocol (MKA) in accordance with IEEE 802.1X-2010 and 802.1Xbx-2014. The data delay protection is enabled for MKA as a protection guard against an attack on the configuration protocols that MACsec is designed to protect by alternately delaying and delivering their PDUs. The Delay protection does not operate if and when MKA operation is suspended. An MKA Lifetime Timeout limit of 6.0 seconds and Hello Timeout limit of 2.0 seconds is enforced by the TOE.

The TOE discards MKPDUs that do not satisfy the requirements listed under FCS_MKA_EXT.1.8. All valid MKPDUs that meet the requirements as defined under FCS_MKA_EXT.1.8 are decoded in a manner conformant to IEEE 802.1x-2010 Section 11.11.4.

On successful peer authentication, a connectivity association is formed between the peers and a secure Connectivity Association Key Name (CKN) is exchanged. After the exchange, the MKA ICV is validated with a Connectivity Association Key (CAK), which is effectively a secret key.

For the Data Integrity Check, MACsec uses MKA to generate an Integrity Check Value (ICV) for the frame arriving on the port. If the generated ICV is the same as the ICV in the frame, then the frame is accepted; otherwise, it is dropped. The key string is the Connectivity Association Key (CAK) that is used for ICV validation by the MKA protocol.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation provides instructions on how to configure the TOE to act as the Key Server in an environment with multiple MACsec-capable devices.

Section “MACsec and MKA Configuration” of the Admin Guide provides instructions for configuring the key-server priority in the MKA policy to ensure that the TOE can act as the Key Server when connecting with MACsec peers.

Testing Assurance Activities: The tests below require the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with. Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the Key Server and principal actor. The evaluator shall then perform the following tests:

Test 1 [conditional]: If the TOE supports group CAKs, the evaluator shall perform the following steps:

1. Load one PSK onto the TOE and device B and a second PSK onto the TOE and device C. This defines two pairwise CAs.

2. Generate a group CAK for the group of 3 devices using ieee8021XKayCreateNewGroup.

3. Observe via packet capture that the TOE distributes the group CAK to the two peers, protected by AES key wrap using their respective PSKs.

4. Verify that B can form a SA with C and connect securely.

5. Disable the KaY functionality of device C using ieee8021XPaePortKayMkaEnable.

6. Generate a group CAK for the TOE and B using ieee8021XKayCreateNewGroup and observe they can connect.

7. The evaluator shall have B attempt to connect to C and observe this fails.

8. Re-enable the KaY functionality of device C.

9. Invoke ieee8021XKayCreateNewGroup again.

10. Verify that both the TOE can connect to C and that B can connect to C.

Test 2: The evaluator shall start an MKA session between the TOE and the two environmental MACsec peers and then perform the following steps:

1. Send an MKPDU to the TOE's individual MAC address from a peer. Verify the frame is dropped and logged.

2. Send an MKPDU to the TOE that is less than 32 octets long. Verify the frame is dropped and logged.

3. Send an MKPDU to the TOE whose length in octets is not a multiple of 4. Verify the frame is dropped and logged.

4. Send an MKPDU to the TOE that is one byte short. Verify the frame is dropped and logged.

5. Send an MKPDU to the TOE with unknown Agility Parameter. Verify the frame is dropped and logged.

Test 1: Not applicable. The TOE does not support Group CAKs.

Test 2: The evaluator set up two MACsec peers to connect to the TOE. The evaluator ensured that the TOE is the Key Server by setting an MKA priority that is lower than the two MACsec peers. The evaluator then started an MKA session between the TOE and the two active participant peers, also taking a packet capture of the session. The evaluator then sent five modified MKA packets according to the conditions identified in this test case. In each case the TOE detected the failure and rejected the packet.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.16 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.16.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

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

The Entropy description is provided in a separate (non-ST) document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section 6.1 (FCS_RBG) of the ST states the TOE implements a NIST-approved AES-CTR DRBG, as specified in NIST SP800-90A seeded by an entropy source that accumulates entropy from a TSF-hardware based noise source. The DRBG is seeded with a minimum of 256 bits of entropy, which is at least equal to the greatest security strength of the keys and hashes that it will generate.

Component Guidance Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

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

The Entropy description is provided in a separate (non-ST) document that has been delivered to NIAP for approval. Nota that the entropy analysis has been accepted by NIAP/NSA.

Section “Enabling FIPS Mode” of the Admin Guide provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography including the proper DRBG methods. Otherwise, there is no further configuration required for configuring RNG functionality.

Component Testing Assurance Activities: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for

each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

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

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

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

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

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

The TOE has been CAVP tested. Refer to the section entitled “CAVP Certificates” earlier in this document.

2.2.17 SSH Server Protocol - per TD0631 (NDcPP22e:FCS_SSHS_EXT.1)

2.2.17.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.17.2 NDcPP22e:FCS_SSHS_EXT.1.2

TSS Assurance Activities: The evaluator shall check to ensure that the TSS contains a list of supported public key algorithms that are accepted for client authentication and that this list is consistent with signature verification algorithms selected in FCS_COP.1/SigGen (e.g., accepting EC keys requires corresponding Elliptic Curve Digital Signature algorithm claims).

The evaluator shall confirm that the TSS includes the description of how the TOE establishes a user identity when an SSH client presents a public key or X.509v3 certificate. For example, the TOE could verify that the SSH client's presented public key matches one that is stored within the SSH server's authorized_keys file.

If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, then the evaluator shall confirm its role in the authentication process is described in the TSS. (TD0631 applied)

Section 6.1 (FCS_SSHS_EXT.1) of the ST states the TOE provides the following authentication methods:

- Public key algorithms for authentication: RSA Signature Verification (rsa-sha2-256, rsa-sha2-512).
- Local password-based authentication for administrative users accessing the TOE through SSHv2, and optionally supports deferring authentication to a remote AAA server.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test objective: The purpose of these tests is to verify server supports each claimed client authentication method.

Test 1: For each supported client public-key authentication algorithm, the evaluator shall configure a remote client to present a public key corresponding to that authentication method (e.g., 2048-bit RSA key when using ssh-rsa public key). The evaluator shall establish sufficient separate SSH connections with an appropriately configured remote non-TOE SSH client to demonstrate the use of all applicable public key algorithms. It is sufficient to observe the successful completion of the SSH Authentication Protocol to satisfy the intent of this test.

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

Test 3: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication succeeds when the correct password is provided by the connecting SSH client.

Test 4: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication fails when the incorrect password is provided by the connecting SSH client.

(TD0631 applied)

Test 1: The TOE supports only ssh-rsa for client public key authentication. The evaluator generated an RSA 2048-bit key pair on the test server and then configured an admin user on the TOE with the public key. The evaluator then attempted to login to the TOE using this ssh-rsa public key and observed that the login was successful.

Test 2: The evaluator next demonstrated an unsuccessful login using an unrecognized public key. The TOE claims support for ssh-rsa public keys only.

Test 3: The evaluator attempted to connect to the TOE using a SSH client alternately using the correct and incorrect password. The evaluator found that only the correct password would yield a successful SSH session.

Test 4: This was performed as part of Test 3.

2.2.17.3 NDcPP22e:FCS_SSHS_EXT.1.3

TSS Assurance Activities: The evaluator shall check that the TSS describes how 'large packets' in terms of RFC 4253 are detected and handled.

Section 6.1 (FCS_SSHS_EXT.1) of the ST states that SSHv2 connections will be dropped if the TOE receives a packet larger than 35,000 bytes. Large packets are detected by the SSHv2 implementation and dropped internal to the SSH process.

Guidance Assurance Activities: None Defined

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

The evaluator created and sent a 262,150-byte packet to the TOE, which is larger than the maximum packet size of 35,000 bytes. The TOE rejected the packet and the connection was closed.

2.2.17.4 NDcPP22e:FCS_SSHS_EXT.1.4

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

Section 6.1 (FCS_SSHS_EXT.1) of the ST states that the TSF's SSH transport implementation supports the following encryption algorithms: aes128-cbc and aes256-cbc. This is consistent with the algorithms specified in the requirement.

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

Section "Remote Administration Protocols" of the Admin Guide provides instructions for configuring the supported encryption algorithms used in SSH: aes128-cbc and aes256-cbc.

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

The evaluator connected to the TOE using an SSH client alternately using the claimed encryption algorithms. The evaluator confirmed that the supported algorithms resulted in successful connections. In each case the evaluator viewed the Server: Key Exchange Init packet and saw that no additional ciphers beyond those that were claimed were seen as supported.

2.2.17.5 NDcPP22e:FCS_SSHS_EXT.1.5

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the SSH server's host public key algorithms supported are specified and that they are identical to those listed for this component. (TD0631 applied)

Section 6.1 (FCS_SSHS_EXT.1) of the ST indicates that the TOE supports password based authentication as well as SSH host key authentication with rsa-sha2-256 and rsa-sha2-512 and client public key authentication with ssh-rsa.

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

Section “Remote Administration Protocols” of the Admin Guide provides instructions for configuring the supported host key algorithms: rsa-sha2-256, rsa-sha2-512, and the client public key authentication algorithm: ssh-rsa.

Testing Assurance Activities: Test objective: This test case is meant to validate that the TOE server will support host public keys of the claimed algorithm types.

Test 1: The evaluator shall configure (only if required by the TOE) the TOE to use each of the claimed host public key algorithms. The evaluator will then use an SSH client to confirm that the client can authenticate the TOE server public key using the claimed algorithm. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Has effectively been moved to FCS_SSHS_EXT.1.2.

Test objective: This negative test case is meant to validate that the TOE server does not support host public key algorithms that are not claimed.

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

(TD0631 applied)

Test 1: The evaluator established an SSH connection with the TOE using each claimed host key algorithm. The connection was successful.

Test 2: The evaluator attempted to connect to the TOE using a host public key algorithm that is not included in the ST selection and observed that the connection failed.

2.2.17.6 NDcPP22e:FCS_SSHS_EXT.1.6

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

Section 6.1 (FCS_SSHS_EXT.1) of the ST states that the TSF's SSH transport implementation supports the following MAC algorithms: hmac-sha2-256, hmac-sha2-512. This is consistent with the requirement.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the 'none' MAC algorithm is not allowed).

Section "Remote Administration Protocols" of the Admin Guide provides instructions for configuring the supported MAC algorithms: hmac-sha-256, hmac-sha2-512.

Testing Assurance Activities: Test 1 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall establish an SSH connection using each of the algorithms, except 'implicit', specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

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

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

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

Test 1: The evaluator established an SSH connection with the TOE using each of the claimed integrity algorithms. The evaluator observed a successful connection using each claimed integrity algorithm.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using the HMAC-MD5 algorithm. The connection attempt failed.

2.2.17.7 NDcPP22e:FCS_SSHS_EXT.1.7

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

Section 6.1 (FCS_SSHS_EXT.1) of the ST states that the TSF's SSH key exchange implementation supports the following key exchange algorithm: diffie-hellman-group14-sha1. This is consistent with the requirement.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Section “Remote Administration Protocols” of the Admin Guide provides instructions for configuring the supported key exchange algorithm: diffie-hellman-group14-sha1.

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

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

Test 1: The evaluator performed this as part of test 2 where the evaluator attempted to establish an SSH connection with the TOE using diffie-hellman-group1-sha1 key exchange. The connection attempt failed.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using each allowed key exchange method: diffie-hellman-group14-sha1. The connection succeeded.

2.2.17.8 NDcPP22e:FCS_SSHS_EXT.1.8

TSS Assurance Activities: The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

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

Section 6.1 (FCS_SSHS_EXT.1) of the ST states the TSF’s SSH implementation will perform a rekey after no longer than one hour or no more than one gigabyte of data has been transmitted with the same session key. Both thresholds are checked. Rekeying is performed upon reaching whichever threshold is met first. The Administrator can configure lower rekey values if desired. The minimum value is 10 minutes. The minimum volume value is 100 kilobytes.

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

Section “Remote Administration Protocols” of the Admin Guide provides instructions for configuring time-based and volume-based rekey values. SSH connections with the same session keys cannot be used longer than one hour, and with no more than one gigabyte of transmitted data. Values can be configured to be lower it desired. The

minimum time value is 10 minutes. The minimum volume value is 100 kilobytes. When configuring an SSH rekey time or volume interval, the TOE will begin re-key based upon the first threshold reached.

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

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

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

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8).

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

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

If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1(3)/Functions).

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

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

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

The evaluator configured the rekey data limit to 100 KB. The evaluator attempted to connect to the TOE using a SSH client sending data and confirmed that a rekey happened at the 100 KB threshold. Next the evaluator

configured the rekey time limit to 10 minutes. The evaluator attempted to connect to the TOE using an SSH client and confirmed that a rekey happened when the configured threshold was reached.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 IDENTIFICATION AND AUTHENTICATION (FIA)

2.3.1 AUTHENTICATION FAILURE HANDLING (MACSECEP12:FIA_AFL.1)

2.3.1.1 MACSECEP12:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 MACSECEP12:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Refer to NDcPP22e:FIA_AFL.1 where this activity is met.

Component Guidance Assurance Activities: The evaluator shall also examine the operational guidance to ensure that instructions for configuring the authentication failure threshold and the TOE's response to the threshold being met (if configurable), 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 trusted path used to access the TSF (see FTP_TRP.1), all must be described.

Refer to NDcPP22e:FIA_AFL.1 where this activity is met.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which remote administrators access the TOE: Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE. The evaluator shall test that once the limit is reached for a given remote administrator account, subsequent attempts with valid credentials are not successful.

Test 2: [conditional] If the TSS indicates that administrative action is necessary to re-enable an account that was locked out due to excessive authentication failures, the evaluator shall perform the steps in Test 1 to lock out an account, follow the operational guidance to manually re-enable the locked out administrator account, and observe that it is once again able to successfully log in.

Test 3: [conditional] If the TSS indicates that an administrator-configurable time period must elapse in order to automatically re-enable an account that was locked out due to excessive authentication failures, the evaluator shall perform the steps in Test 1 to lock out an account, follow the operational guidance to configure a time period of their choosing, and observe through periodic login attempts that the account cannot successfully log in until the configured amount of time has elapsed. The evaluator shall then repeat this test for a different time period of their choosing.

Test 1: This test is covered by the testing for NDcPP22e:FIA_AFL.1-t1 where the number of unsuccessful logins exceeded the configured threshold and the account was locked out for a specified time period.

Test 2: The evaluator verified that they could login as a user after the lockout occurred and the account was manually unlocked by an administrator in the test case NDcPP22e:FIA_AFL.1-t1.

Test 3: Not applicable. Lockout for an Administrator defined period of time action is not claimed.

2.3.2 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.3.2.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.2.2 NDcPP22e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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).

Section 6.1 (FIA_AFL.1) of the ST states the TOE provides the privileged administrator the ability to specify the maximum number of unsuccessful authentication attempts before privileged administrator or non-privileged administrator is locked out through the administrative CLI using a privileged CLI command. While the TOE supports a range from 1-25, in the evaluated configuration, the maximum number of failed attempts is recommended to be set to 3.

When a privileged administrator or non-privileged administrator attempting to log into the administrative CLI reaches the administratively set maximum number of failed authentication attempts, the user will not be granted access to the administrative functionality of the TOE until a privileged administrator resets the user's number of failed login attempts through the administrative CLI. Administrator lockouts are not applicable to the local console.

Component Guidance Assurance Activities: 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.

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.

Section “User Lockout” of the Admin Guide provides instructions for specifying the value for maximum number of failed login attempts and the time period to lock the offending account.

Section “Remote Administration Protocols” of the Admin Guide provides instructions for configuring the switch for SSH public key authentication. This is necessary to avoid a potential situation where password failures by remote Administrators lead to no Administrator access until the account is manually unlocked. During the defined lockout period, the TOE provides the ability for the Administrator account to login remotely using SSH public key authentication.

Component Testing Assurance Activities: 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 authorization attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorization attempt using valid credentials results in successful access.

Test 1 & 2: The evaluator configured a limit on failed local authentication attempts (i.e., 3 failures). The evaluator then performed the same number of login attempts using incorrect credentials than the configured limit. The evaluator observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator then logged in as admin and unlocked the locked user account and observed that the user could login successfully with the correct password.

2.3.3 Password Management (NDcPP22e:FIA_PMG_EXT.1)

2.3.3.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it contains the lists of the supported special character(s) and minimum and maximum number of characters supported for administrator passwords.

Section 6.1 (FIA_PMG_EXT.1) of the ST states the TOE supports the local definition of users with corresponding passwords. The passwords can be composed of any combination of upper and lower case letters, numbers, and special characters that include: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)” and other special characters listed in the table below. Minimum password length is settable by the Authorized Administrator, and can be configured for minimum password lengths of 8 to 16 characters and maximum of 127 characters.

Special Character	Name
	Space

;	Semicolon
:	Colon
“	Double Quote
‘	Single Quote
|	Vertical Bar
+	Plus
-	Minus
=	Equal Sign
.	Period
,	Comma
/	Slash
\	Backslash
<	Less Than
>	Greater Than
_	Underscore
`	Grave accent (backtick)
~	Tilde
{	Left Brace
}	Right Brace

Component Guidance Assurance Activities: 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.

Section “Passwords” of the Admin Guide provides instructions for defining a common criteria policy that can be applied to each local account and that will ensure that passwords contain a minimum of 8 characters or greater.

Component Testing Assurance Activities: The evaluator shall perform the following tests.

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.

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.

Test 1 & 2: The evaluator performed attempts to set passwords of varying lengths and characters to demonstrate that passwords comply with a minimum length and support the claimed set of characters.

2.3.4 Extended: Pre-Shared Key Composition (MACsecEP12:FIA_PSK_EXT.1)

2.3.4.1 MACsecEP12:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.2 MACsecEP12:FIA_PSK_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it describes the process by which the bit-based pre-shared keys are generated (if the TOE supports this functionality), and confirm that this process uses the RBG specified in FCS_RBG_EXT.1.

The TOE does not support bit-based pre-shared key generation.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on the composition of strong pre-shared keys, and (if the selection indicates keys of various lengths can be entered) that it provides information on the range of lengths supported

The evaluator shall confirm the operational guidance contains instructions for either entering bit-based pre-shared keys for each protocol identified in the requirement, or generating a bit-based pre-shared key (or both). The evaluator shall also examine the TSS to ensure it describes the process by which the bit-based pre-shared keys are

generated (if the TOE supports this functionality), and confirm that this process uses the RBG specified in FCS_RBG_EXT.1.

Section "IKEv2 Transform Sets" of the Admin Guide provides instructions for configuring the IKEv2 keyring for pre-shared key authentication. The key can be entered as character strings or hex values. The TOE supports pre-shared keys up to 128 bytes in length. The recommendation for a strong pre-shared key is a minimum of length of 22 characters composed of any combination of upper and lower case letters, numbers, and special characters (that include: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")").

Section "MACsec and MKA Configuration" of the Admin Guide provides instructions for configuring a keychain and a pre-shared key in the form of a HEX string via the key-string command in the key chain. When specifying the value of the key identifier, the Administrator must ensure the length does not exceed 64 hex digits (32 bytes).

Component Testing Assurance Activities: The evaluator shall also perform the following tests for each protocol (or instantiation of a protocol, if performed by a different implementation on the TOE). Note that one or more of these tests can be performed with a single test case.

Test 1 [conditional]: If the TOE supports pre-shared keys of multiple lengths, the evaluator shall use the minimum length; the maximum length; a length inside the allowable range; and invalid lengths beyond the supported range (both higher and lower). The minimum, maximum, and included length tests should be successful, and the invalid lengths must be rejected by the TOE.

Test 2 [conditional]: If the TOE does not generate bit-based pre-shared keys, the evaluator shall obtain a bit-based pre-shared key of the appropriate length and enter it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

Test 3 [conditional]: If the TOE does generate bit-based pre-shared keys, the evaluator shall generate a bit-based pre-shared key of the appropriate length and use it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

These tests were performed for both MACsec and IPsec.

Test 1: The evaluator attempted to establish a connection using pre-shared keys of valid and invalid lengths and confirmed that pre-shared keys with valid lengths resulted in successful connections, while the attempt to configure pre-shared keys of invalid lengths was unsuccessful.

Test 2: The TOE does not generate bit based pre-shared keys. The evaluator entered a pre-shared key according to instructions in the guide and demonstrated a successful connection.

Test 3: Not applicable. The TOE does not generate bit-based keys.

2.3.5 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.3.5.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: 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.

There are no preparatory steps needed to ensure authentication data is not revealed. Section 6.1 (FIA_UAU.7) of the ST states that when a user enters their password at the local console or via a remote session, the TOE does not echo any characters as the password is entered.

Component Testing Assurance Activities: 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.

Test 1: This test was performed as part of the tests for FIA_UIA_EXT.1 where the evaluator observed that passwords are obscured on the console logins.

2.3.6 Password-based Authentication Mechanism (NDcPP22e:FIA_UAU_EXT.2)

2.3.6.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See NDcPP22e:FIA_UIA_EXT.1.

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

See NDcPP22e:FIA_UIA_EXT.1.

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

See NDcPP22e:FIA_UIA_EXT.1.

2.3.7 User Identification and Authentication (NDcPP22e:FIA_UIA_EXT.1)

2.3.7.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.7.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no unauthorized access to any TOE component can occur.

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

Section 6.1 (FIA_UIA_EXT.1) of the ST states that the TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed except for the login warning banner that is displayed prior to user authentication and any network packets as configured by the Authorized Administrator may flow through the switch.

Administrative access to the TOE is facilitated through the TOE's CLI. The TOE mediates all administrative actions through the CLI. Once a potential administrative user attempts to access the CLI of the TOE through either a directly connected console or remotely through an SSHv2 secured connection, the TOE prompts the user for a username and password. Only after the administrative user presents the correct authentication credentials will access to the TOE administrative functionality be granted. No access is granted to the administrative functionality of the TOE until an administrator is successfully identified and authenticated.

The TOE provides a local password-based authentication mechanism as well as RADIUS AAA server for remote authentication. The administrator authentication policies include authentication to the local user database or redirection to a remote authentication server. Interfaces can be configured to try one or more remote authentication servers, and then fail back to the local user database if the remote authentication servers are inaccessible.

The process for authentication is the same for administrative access whether administration is occurring via a directly connected console or remotely via SSHv2 secured connection.

At initial login, the administrative user is prompted to provide a username. After the user provides the username, the user is prompted to provide the administrative password associated with the user account. The TOE then either grants administrative access (if the combination of username and password is correct) or indicates that the login was unsuccessful. The TOE does not provide a reason for failure in the cases of a login failure.

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

The guidance contains configuration for the remote SSH interface in the "Remote Administration Protocols" section. The details of the configuration, preparatory steps, and claimed functionality are described in the assurance activities for the SSH protocol. As identified in the FIA requirements, the password and account lockout mechanisms are described in the guidance.

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

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A

information results in the ability to access the system, while providing incorrect information results in denial of access.

b) Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

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

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

The TOE offers the following user interfaces where authentication is provided:

- Local Console using local authentication with password
- SSH CLI using local authentication with password
- SSH CLI using local authentication with public key
- SSH CLI using RADIUS account

Test 1: Using each interface, the evaluator performed an unsuccessful and successful login of each type using bad and good credentials, respectively.

Test 2: Using each interface, the evaluator was able to observe the TOE displayed a banner to the user before login and that there were no other services available nor any configuration options offered to administrators to control services available prior to authentication.

Test 3: This test was performed as part of test 1. Using each interface, the evaluator found that, prior to login, no functions were available to the administrator with the exception of acknowledging the banner.

Test 4: Not applicable. The TOE is not a distributed TOE.

2.3.8 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/Rev)

2.3.8.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for

FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

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

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

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

c) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates - conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the peer certificate and revocation of the peer intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore the revoked certificate(s) used for testing shall not be a trust anchor.

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

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

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

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

h) The following tests are run when a minimum certificate path length of three certificates is implemented.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall conduct the following tests:

Test 8a: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a valid chain of EC certificates (terminating in a trusted CA certificate), where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

Test 8b: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a chain of EC certificates (terminating in a trusted CA certificate), where the intermediate certificate in the certificate chain uses an explicit format version of the Elliptic Curve parameters in the public key information field, and is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

Test 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

(TD0527 12/2020 update applied)

Test 1: The evaluator alternately configured the TOE to have and then not have the trusted root CA used by Strongswan on the test peer to anchor all of its certificates. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to succeed only when the trusted root CA was properly configured forming a valid certificate chain.

Test 2: The evaluator alternately configured Strongswan on a test peer to send an authentication certificate 1) that is valid, 2) that is expired, and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to succeed only if there are no expired certificates.

Test 3: The evaluator alternately configured Strongswan on a test peer to send an authentication certificate 1) that is valid, 2) that is revoked, and 3) issued by an intermediate CA that is revoked. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to succeed only if there are no revoked certificates.

Test 4: The evaluator alternately configured Strongswan on a test peer to send an authentication certificate 1) that is valid, 2) issued by an intermediate CA referring to a CRL revocation server where the signer lacks cRLSign, and 3) issued by an intermediate CA whose issuer CA refers to a CRL revocation server where the signer lacks cRLSign. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to succeed only if all retreived CRLs are signed using certificates with cRLSign.

Test 5: For this test, the evaluator alternately configured Strongswan on a test peer to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to succeed only if the certificate is not modified/corrupted.

Test 6: This test has been performed in FIA_X509_EXT.1.1/Rev-t5.

Test 7: This test has been performed in FIA_X509_EXT.1.1/Rev-t5.

Test 8: The evaluator alternately configured Strongswan on a test peer to send an authentication certificate issued by a Sub CA with a valid elliptic curve and an explicitly defined elliptic curve. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to be rejected in the second case.

2.3.8.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation). For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

a) Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate

without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

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

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

Test 1: For this test, the evaluator alternately configured a test peer to send an authentication certificate issued by a Sub CA with no basicConstraints and with basicConstraints but the CA Flag set to false. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and observed that the connection was rejected in each case.

Test 2: This was performed as part of Test 1.

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

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

Section 6.1 (FIA_X509_EXT.1/Rev) of the ST states the TOE uses X.509v3 certificates to support authentication for IPsec connections. The TSF determines the validity of certificates by ensuring that the certificate and the certificate path are valid in accordance with RFC 5280. The certificate path is validated by ensuring that all the CA certificates have the basicConstraints extension and the CA flag is set to TRUE and the certificate path must terminate with a trusted CA certificate. CRL revocation checking is supported by the TOE. Revocation checking is performed on the leaf and intermediate certificate(s) when authenticating a certificate chain provided by the remote peer. There are no functional differences if a full certificate chain or only a leaf certificate is presented.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for

extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

Section "IPsec Overview" of the Admin Guide states that the TOE uses X.509v3 certificates to support authentication for IPsec connections. The TOE determines validity of certificates by ensuring that the certificate and the certificate path are valid in accordance with RFC 5280. The certificate path is valid by ensuring that all the CA certificates have the basicConstraints extension and the CA flag is set to TRUE and the certificate path must terminate with a trusted CA certificate. OCSP is not supported; therefore, the OCSP Signing purpose (id-kp with OID 1.3.6.1.5.5.7.3.9) is trivially satisfied by the TOE. Revocation checking is performed on the leaf and intermediate certificate(s) when authenticating a certificate chain provided by the remote peer.

Component Testing Assurance Activities: None Defined

2.3.9 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2)

2.3.9.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.2 NDcPP22e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The evaluator shall examine the TSS to confirm that it describes the behaviour of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed.

Section 6.1 (FIA_X509_EXT.2) of the ST states the TOE determines which certificate to use based upon the trustpoint configured. The instructions for configuring trustpoints is provided in the CC Configuration Guide. If a network connection cannot be established to verify the revocation status of certificate for an external peer the connection will be rejected.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the configuration required in the operating environment so the TOE can use the certificates. The guidance documentation shall also include any required configuration on the TOE to use the certificates. The guidance document shall also describe the steps for the Security Administrator to follow if the connection cannot be established during the validity check of a certificate used in establishing a trusted channel.

Section "X.509 Certificates" of the Admin Guide explains how to use X.509 certificates with the TOE. It explains how to request a certificate, how to communicate with a certificate authority, how to load a certificate, and how to configure revocation. If the TOE does not have the applicable CRL and is unable to obtain one, the TOE will reject the peer's certificate.

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

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

The evaluator alternately configured a test peer to send an authentication certificate with valid/accessible revocation servers and an authentication certificate with revocation information referring to an inaccessible revocation server. In each case, the evaluator then attempted to make a connection between the test peer and the TOE expecting the connection to be successful when the revocation server is accessible and when the revocation server is not accessible only if that behavior is claimed for the TOE. The evaluator observed the certificate validation checking behavior in each case and confirmed that it was consistent with the actions selected in FIA_X509_EXT.2.2 in the ST.

2.3.10 X.509 Certificate Requests (NDcPP22e:FIA_X509_EXT.3)

2.3.10.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.10.2 NDcPP22e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If the ST author selects 'device-specific information', the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

The ST does not select ‘device-specific information’.

Section 6.1 (FIA_X509_EXT.3) of the ST states that a Certificate Request Message can be generated as specified by RFC 2986 and provide the following information in the request – Common Name (CN), Organization (O), Organizational Unit (OU), and Country. The TOE can validate the chain of certificates from the Root CA when the CA Certificate Response is received.

Component Guidance Assurance Activities: The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certification Request. If the ST author selects 'Common Name', 'Organization', 'Organizational Unit', or 'Country', the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the Certification Request.

Section “Creation of the Certificate Signing Request” of the Admin Guide provides instructions for creating the root CA and subordinate CA trustpoints, which includes establishing the subject-name to at least the Common Name, Organization, Organizational Unit, and Country. The certificate signing request is then generated from the subordinate CA trustpoint and includes the Common Name (CN), Organization (O), Organizational Unit (OU), and Country as specified in the requirement.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

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

b) Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the response message, and demonstrate that the function succeeds.

Test 1: The evaluator followed operational guidance to log into the TOE and then generated the CSR. The evaluator captured the generated request and ensured that it contains the information claimed in the ST.

Test 2: The evaluator attempted to import a certificate without a valid certification path and the import failed. The evaluator then attempted to import a certificate with a valid certification path and the import succeeded.

2.4 Security Management (FMT)

2.4.1 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate)

2.4.1.1 NDcPP22e:FMT_MOF.1.1/ManualUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

There are no specific requirements for non-distributed TOEs.

The TOE is not distributed.

Component Guidance Assurance Activities: 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).

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

Section “Secure Acceptance of the TOE” of the Admin Guide explains how to perform a manual update of the TOE. It explains the image on the Cisco website and as part of the install, the signature will be validated. If the signature is not correct, the device will not boot. Steps 7 and 9 provide instructions for how to download and verify an image prior to running it on the TOE.

Component Testing Assurance Activities: 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 should fail.

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

As can be seen in the FIA_UIA_EXT.1 test evidence, no functions are offered to users prior to a successful login. Any user that can login is considered an administrator and can perform TOE updates.

FTP_TUD_EXT.1 demonstrates the successful updating of the TOE by a trusted administrator.

2.4.2 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData)

2.4.2.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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.

Section 6.1 (FMT_MTD.1/CoreData) of the ST states the TOE provides the ability for Security Administrators to access TOE data, such as audit data, configuration data, security attributes, session thresholds, cryptographic keys and to perform manual updates to the TOE. Only Security Administrators can access the TOE's trust store. Each of the predefined and administratively configured roles has create (set), query, modify, or delete access to the TOE data, through with some privilege levels, the access is limited.

The TOE performs role-based authorization, using TOE platform authorization mechanisms, to grant access to the privileged and semi-privileged roles. For the purposes of this evaluation, the privileged level is equivalent to full administrative access to the CLI, which is the default access for IOS-XE privilege level 15; and the semi-privileged level equates to any privilege level that has a subset of the privileges assigned to level 15. Privilege levels 0 and 1 are defined by default and are customizable, while levels 2-14 are undefined by default and also customizable.

Component Guidance Assurance Activities: 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.

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.

Section "X.509 Certificates" of the Admin Guide explains how to use X.509 certificates with the TOE. It explains how to request a certificate, how to communicate with a certificate authority, and how to load a certificate.

Component Testing Assurance Activities: 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.

No additional testing was required as all management functions were demonstrated throughout the course of testing other SFRs.

2.4.3 Management of TSF Data (NDcPP22e:FMT_MTD.1/CryptoKeys)

2.4.3.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see chapter 2.4.1.1.

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 how that how those operations are performed.

Section 6.1 (FMT_MTD.1/CryptoKeys) of the ST states that only Security Administrators can access the TOE's trust store. This section describes that the Authorized Administrator generates RSA key pairs to be used in SSH protocol. This section also describes that the TOE Administrators can control (generate/delete) the following keys, IKE RSA Key Pairs and SSH RSA Key Pairs by following the instructions in the AGD. This section further explains that the TOE provides the ability for Security Administrators to access TOE data, such as audit data, configuration data, security attributes, session thresholds, cryptographic keys and to perform manual updates to the TOE. Each of the predefined and administratively configured roles has create (set), query, modify, or delete access to the TOE data, though with some privilege levels, the access is limited.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation 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 how that how those operations are performed.

Section "Remote Administration Protocols" of the Admin Guide provides instructions for generating an RSA 3072 bit key for SSH, configuring the SSH server key exchange algorithm (Diffie-Hellman group 14 sha1), configuring the host key and user public key algorithms, and configuring SSH rekey settings.

Section "Generate a Key Pair" of the Admin Guide provides instructions for generating an RSA 3072 bit key for use with IPsec.

Section “Creation of the Certificate Signing Request” of the Admin Guide includes instructions for mapping the RSA key generated for IPsec to a configured trustpoint, importing the CA certificate and generating a certificate signing request.

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

As can be seen in the FIA_UIA_EXT.1 test evidence, no functions are offered to users prior to a successful login. Any user that can login is considered an administrator and can perform TOE updates.

FIA_X509_EXT.3 demonstrates the successful installing of crypto keys by an authorized administrator.

2.4.4 Specification of Management Functions - per TD0652 (MACsecEP12:FMT_SMF.1)

2.4.4.1 MACsecEP12:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the ability of the TOE to provide the management functions defined in this SFR in addition to the management functions required by the base NDcPP.

The TSS activity for all management functions, including MACsec specific functions, was performed in NDcPP22e:FMT_SMF.1.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides instructions on how to perform each of the management functions defined in this SFR in addition to those required by the base NDcPP.

As addressed in the relevant SFRs throughout this document, section “MACsec and MKA Configuration” of the Admin Guide provides instructions for how to perform the management functions relevant to this SFR. This includes instructions for configuring and enabling a PSK based CAK using the ‘key chain’ command, configuring the key-string and the lifetime of a CAK and configuring the key-server priority in the MKA policy to ensure that the

TOE can act as the Key Server when connecting with MACsec peers. Section “User Lockout” of the Admin Guide provides instructions for configuring the number of failed authentication attempts that will cause an account to be locked out using the “aaa local authentication attempts max-fail [*number of failures*]” command. The number of failures is the number of consecutive failures that will trigger locking of the account. The minimum value is 1 and the maximum value it 65535.

Component Testing Assurance Activities: The evaluator shall set up an environment where the TOE can connect to two other MACsec devices, identified as devices B and C, with the ability of pre-shared keys to be distributed between them. The evaluator shall configure the devices so that the TOE will be elected key server and principal actor, i.e., has highest key server priority.

In addition to the tests specified in the NDcPP for this SFR, the evaluator shall follow the relevant operational guidance to perform the tests listed below. Note that if the TOE claims multiple management interfaces, the tests should be performed for each interface that supports the functions.

Test 1: The evaluator shall connect to the PAE of the TOE and install a PSK. The evaluator shall then specify a CKN and that the PSK is to be used as a CAK.

- Repeat this test for both 128-bit and 256-bit key sizes.

- Repeat this test for a CKN of valid length (1-32 octets), and observe success.

- Repeat this test again for CKN of invalid lengths zero and 33, and observe failure.

Test 2: The evaluator will test the ability of the TOE to enable and disable MKA participants using the management function specified in the ST. The evaluator shall install pre-shared keys in devices B and C, and take any necessary additional steps to create corresponding MKA participants. The evaluator shall disable the MKA participant on device C, then observe that the TOE can communicate with B but neither the TOE nor B can communicate with device C. The evaluator shall re-enable the MKA participant of device B and observe that the TOE is now able to communicate with devices B and C.

Test 3: For TOEs using only PSKs, the TOE should be the Key Server in both tests and only one peer (B) needs to be tested. The tests are:

Subtest a (Switch to unexpired CKN): TOE and Peer B have CKN1(10 minutes) and CKN2. CKN2 can either be configured with a longer overlapping lifetime (20 minutes) or be configured with a lifetime starting period of more than 10 minutes after the CKN1 start. The TOE and Peer B start using CKN1 and after 10 minutes, verify that the TOE expires SAK1. This can be verified by either 1) seeing the TOE immediately distribute a new SAK to the peer if the lifetime of CKN2 overlaps CKN1, or 2) by terminating the connection with CKN1 and distributing a new SAK once the lifetime period of CKN2 begins.

Subtest b (reject CA with expired CKN): TOE has CKN1(10 minutes). Peer B has CKN1(20 minutes). TOE and Peer B start using CKN1 and after 10 minutes, verify that the TOE rejects (or ignores) peer's request to use (or distribute a) SAK using CKN1.

Test 4: If 'Cause Key Server to generate a new group CAK...' is selected, the evaluator shall connect to the PAE of the TOE, set the management function specified in the ST (e.g., set ieee8021XKeyCreateNewGroup to true), and observe that the TOE distributes a new group CAK.

Test 1: The evaluator first configured CKNs with valid minimum and maximum lengths and established successful connections. The evaluator then attempted to configure a CKN length below the minimum and viewed the attempt failed as expected. Maximum CKN lengths are enforced by guidance.

Test 2: The evaluator configured the TOE to establish MACsec connections with peers B and C. The evaluator ensured that the TOE could establish a MACsec connection with peers B and C. The evaluator then issued a command in the TOE to disable MACsec on peer C. The evaluator observed that no client could communicate with peer C while the it was disabled. The evaluator then re-enabled MACsec on peer C and observed that communications with peer C were successful.

Test 3: The evaluator configured the TOE with CKN1, which expires in 10 minutes, and CKN2, which does not expire. The tester set up a valid MACsec channel between the TOE and the peer using CKN1. After 10 minutes, the evaluator analyzed the TOE logs and packet capture. The evaluator determined that a new SAK was distributed using CKN2.

Test 4: Not applicable. Group CAKs are not supported by the TOE.

2.4.5 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1)

2.4.5.1 NDcPP22e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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(1)/ManualUpdate, FMT_MOF.1(4)/AutoUpdate (if included in the ST), FIA_AFL.1, FIA_X509_EXT.2.2 (if included in the ST), FPT_TUD_EXT.1.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(2)/Services, and FMT_MOF.1(3)/Functions (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

(containing also requirements on Guidance Documentation and Tests)

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

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.

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.

Section 6.1 (FMT_SMF.1) of the ST provides a description of the management functions of the TOE. The TOE provides all the capabilities necessary to securely manage the TOE and the services provided by the TOE. The management functionality of the TOE is provided through the TOE CLI to perform these functions via SSHv2 secured connection, a terminal server, or at the local console. The specific management capabilities available from the TOE include:

- Local and remote administration of the TOE and the services provided by the TOE via the TOE CLI, as described above.
- The ability to manage the warning banner message and content which allows the Authorized Administrator the ability to define warning banner that is displayed prior to establishing a session (note this applies to the interactive (human) users, e.g., administrative users.
- The ability to set and modify the time limits of session inactivity.
- The ability to configure the number of failed administrator logon attempts that will cause the account to be locked until it is reset.
- Ability to re-enable an Administrator account.
- Ability to manually unlock a locked administrator account.
- The ability to update the IOS-XE software. The validity of the image is provided using SHA-256 and/or digital signature prior to installing the update.
- The ability to manage audit behaviour and the audit logs which allows the Authorized Administrator to configure the audit logs, view the audit logs, and to clear the audit logs.
- The ability to manage the cryptographic functionality which allows the Authorized Administrator the ability to identify and configure the algorithms used to provide protection of the data, such as generating the RSA keys to enable SSHv2.
- The ability to configure the IPsec functionality which supports the secure connections to the audit server and the remote authentication server.
- The ability to import the X.509v3 certificates and validate for use in authentication and secure connections.
- The ability to manage the Key Server and associated MKA participants.
- The ability to generate a PSK and install in the CAK cache.
- The ability to initiate the generation of a new CAK from the Key Server The ability to specify the lifetime of a CAK and to enable, disable or delete a PSK in the CAK cache of a device.
- The ability to configure and set the time clock.
- The ability to configure the reference identifiers.

Component Guidance Assurance Activities: See TSS Assurance Activities

See TSS Assurance Activities.

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

All TOE security functions are identified and have been tested as documented throughout this AAR.

2.4.6 Restrictions on Security Roles (NDcPP22e:FMT_SMR.2)

2.4.6.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.6.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.6.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.1 (FMT_SMR.2) of the ST states the TOE platform maintains privileged and semi-privileged administrator roles. The TOE performs role-based authorization, using TOE platform authorization mechanisms, to grant access to the semi-privileged and privileged roles. For the purposes of this evaluation, the privileged role is equivalent to full administrative access to the CLI, which is the default access for IOS-XE privilege level 15; and the semi-privileged role equates to any privilege level that has a subset of the privileges assigned to level 15. Privilege levels 0 and 1 are defined by default and are customizable, while levels 2-14 are undefined by default and are also customizable. Note: the levels are not hierarchical.

The TOE can and shall be configured to authenticate all access to the command line interface using a username and password. The TOE supports both local administration via a directly connected console cable and remote administration via SSH or IPsec over SSH.

Component Guidance Assurance Activities: 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.

See FIA_UIA_EXT.1 which identifies the instructions of the Admin Guide for administering the TOE both locally and remotely.

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

Testing throughout the course of the evaluation was performed using both the SSH and local hardware interfaces.

2.5 Protection of the TSF (FPT)

2.5.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1)

2.5.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.1 (FPT_APW_EXT.1) of the ST states the TOE is designed specifically to not disclose any keys stored in the TOE. The TOE stores all private keys in a secure directory that cannot be viewed or accessed, even by the Administrator. The TOE stores symmetric keys only in volatile memory. Pre-shared keys may be specified in the configuration file by the Administrator using a bit-based (hex) format. Only the Administrator may view the configuration file.

The TOE is designed specifically to not disclose any passwords stored in the TOE. All passwords are stored using a SHA-2 hash. 'Show' commands display only the hashed password.

The CC Configuration Guide instructs the Administrator to use the algorithm-type sha256 or scrypt sub-command when passwords are created or updated. The SHA256 sub-command is password type 8 while scrypt is password type 9. Both password types use SHA-2.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.2 Protection of CAK Data (MACsecEP12:FPT_CAK_EXT.1)

2.5.2.1 MACsecEP12:FPT_CAK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how CAKs are stored and that they are unable to be viewed through an interface designed specifically for that purpose. If these values are not stored in plaintext, the TSS shall describe how they are protected or obscured.

Section 6.1 (FPT_CAK_EXT.1) of the ST states during the setup and configuration of the TOE and the MACsec functionality, the Authorized Administrator issues the command – "service password–encryption". This prevents the CAK value from being shown in clear text to the administrators on the CLI when the "show run" output is displayed. In addition, CAK data is stored in secure directory that is not readily accessible to administrators.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.3 SelfTest Failure with Preservation of Secure State (MACsecEP12:FPT_FLS.1(2))

2.5.3.1 MACsecEP12:FPT_FLS.1.1(2)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it indicates that the TSF will shut down in the event that a self-test failure is detected. For TOEs with redundant failover capability, the evaluator shall examine the TSS to determine that it indicates that the failed components will shut down in the event that a self-test failure is detected. (TD0190 applied)

Section 6.1 (FPT_FLS.1(2)/SelfTest) of the ST states whenever a failure occurs (power-on self-tests, integrity check of the TSF executable image and/or the noise source health-tests) within the TOE that results in the TOE ceasing operation, the TOE securely disables its interfaces to prevent the unintentional flow of any information to or from the TOE and reloads.

If the failures persist, the TOE will continue to reload in an attempt to correct the failure. This functionally prevents any failure from causing an unauthorized information flow. There are no failures that circumvent this protection. If the rebooting continues, the Authorized Administrator should contact Cisco Technical Assistance Center (TAC).

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it describes the behavior of the TOE following a self-test failure and actions that an administrator should take if it occurs.

Section “Cryptographic Self-Tests” of the Admin Guide explains the TOE runs a suite of self-tests during initial start-up to verify correct operation of cryptographic modules. If any component reports failure for the POST, the system crashes and appropriate information is displayed on the local console. All ports are blocked from moving to forwarding state during the POST. If all components of all modules pass the POST, the system is placed in FIPS PASS state and ports are allowed to forward data traffic. If any of the tests fail, a message is displayed to the local console and the TOE component will automatically reboot. If the Administrator observes a cryptographic self-test failure, they should contact Cisco Technical Support. This section also provides a command for the Administrator to execute cryptographic self-tests for the Switch after the image is loaded.

Component Testing Assurance Activities: The following test may require the vendor to provide access to a test platform that provides the evaluator with the ability to modify the TOE internals in a manner that is not provided to end customers:

Test 1: The evaluator shall modify the TSF in a way that will cause a self-test failure to occur. The evaluator shall determine that the TSF shuts down and that the behavior of the TOE is consistent with the operational guidance. The evaluator shall repeat this test for each type of self-test that can be deliberately induced to fail.

For TOEs with redundant failover capability, the evaluator shall determine that the failed components shut down and the behavior of the TOE is consistent with the operational guidance. For each component, the evaluator shall repeat each type of self-test that can be deliberately induced to fail. (TD0190 applied)

The evaluator used special builds provided by the vendor to cause failure of the integrity check of the image and failure of each of the relevant power-on self-tests. In each case the TOE rebooted as expected due to the self-test failure.

2.5.4 Replay Detection (MACsecEP12:FPT_RPL.1)

2.5.4.1 MACsecEP12:FPT_RPL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.4.2 MACsecEP12:FPT_RPL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes how replay is detected for MPDUs and how replayed MPDUs are handled by the TSF.

Section 6.1 (FPT_RPL.1) of the ST states replayed data is discarded by the TOE and the attempt to replay data is logged. MDPUs are replay protected in the TOE. Also, the MKA frames are guarded against replay (If a MKPDU with duplicate MN (member number) and not latest MN comes along, then this MKPDU will be dropped and not processed further). Replayed data is discarded and logged by the TOE.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Before performing each test the evaluator shall successfully establish a MACsec channel between the TOE and a MACsec-capable peer in the Operational Environment sending enough traffic to see it working and verify the PN values increase for each direction

Test 1: The evaluator shall set up a MACsec connection with an entity in the Operational Environment. The evaluator shall then capture traffic sent from this remote entity to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the remote entity where the PN values in the SecTag of these packets are less than the lowest acceptable PN for the SA. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

The evaluator shall establish a MACsec connection between the TOE and a test system. The evaluator shall then capture traffic sent from test system to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in

order to impersonate the remote entity where the PN values in the SecTag of these packets are less than the lowest acceptable PN for the SA. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

Test 2: The evaluator will capture frames during a MKA session and record the lowest PN observed in a particular time range. The evaluator will then send a frame with a lower PN, and then verify that this frame is dropped. The evaluator will verify that the device logged this event.

Test 1: The evaluator enabled replay protection on the TOE and set up a successful MACsec connection between the TOE and a test system. The evaluator captured MACsec traffic sent from the test system to the TOE and then attempted to send the same traffic, which contains an old packet number (PN). The TOE successfully detected the invalid PN and dropped the traffic along with reporting an audit log of the event. The evaluator then attempted the same test, only this time the evaluator sent MKA traffic that was already sent. The evaluator noted that the TOE successfully detected the invalid PN, dropped the traffic, and reported the error in an audit log.

Test 2: This was performed as part of test 1 above.

2.5.5 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1)

2.5.5.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FPT_SKP_EXT.1) of the ST states the TOE stores all private keys in a secure directory that cannot be viewed or accessed, even by the Administrator. The TOE stores symmetric keys only in volatile memory.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.6 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1)

2.5.6.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.6.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

If 'obtain time from the underlying virtualization system' is selected, the evaluator shall examine the TSS to ensure that it identifies the 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.

Section 6.1 (FPT_STM_EXT.1) of the ST states the TOE TSF implements a clock function to provide a source of date and time. The clock function is reliant on the system clock provided by the underlying hardware. All Switch models have a real-time clock (RTC) with battery to maintain time across reboots and power loss.

The TOE relies upon date and time information for the following security functions:

- To monitor local and remote interactive administrative sessions for inactivity (FTA_SSL_EXT.1, FTA_SSL.3);
- Validating X.509 certificates to determine if a certificate has expired (FIA_X509_EXT.1/Rev);
- To determine when SSH session keys have expired and to initiate a rekey (FCS_SSHS_EXT.1);
- To determine when IKEv2 SA lifetimes have expired and to initiate a rekey (FCS_IPSEC_EXT.1);
- To determine when iPsec Child SA lifetimes have expired and to initiate a rekey (FCS_IPSEC_EXT.1);
- To provide accurate timestamps in audit records (FAU_GEN.1.2).

Additionally, ‘obtain time from the underlying virtualization system’ is not selected for the TOE.

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

If the TOE supports obtaining time from the underlying VS, the evaluator shall verify the Guidance Documentation specifies any configuration steps necessary. If no configuration is necessary, 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, the evaluator shall ensure the Guidance Documentation informs the administrator of the maximum possible delay.

Section “Clock Management” of the Admin Guide explains how to set the time on the TOE.

The TOE does not support obtaining time from the underlying VS.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

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

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

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.

c) Test 3: [conditional] If the TOE obtains time from the underlying VS, 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 the setting the time on the VS and when the time is reflected on the TOE, the evaluator shall ensure this delay is consistent with the TSS and Guidance.

Test 1: The evaluator followed the guidance instructions to configure the time on the TOE. The evaluator read the time from the TOE using a date command and also found audit records confirming that the time was successfully changed.

Test 2: Not applicable. NTP is not supported.

Test 3: Not applicable. The TOE does not support obtaining time from the underlying VS.

2.5.7 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.5.7.1 NDCPP22E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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.

Section 6.1 (FPT_TST_EXT.1) of the ST explains the self-tests. The TOE runs a suite of self-tests during initial start-up to verify its correct operation. For testing of the TSF, the TOE automatically runs checks and tests at start-up, during resets and periodically during normal operation to ensure the TOE is operating correctly, including checks of image integrity and all cryptographic functions.

During the system bootup process (power on or reboot), all the Power on Startup Test (POST) components for all the cryptographic modules perform the POST for the corresponding component (hardware or software). The TOE performs the following tests:

- AES Known Answer Test: For the encrypt test, a known key is used to encrypt a known plain text value resulting in an encrypted value. This encrypted value is compared to a known encrypted value. If the encrypted texts match, the test passes; otherwise, the test fails. The decrypt test is just the opposite. In this test a known key is used to decrypt a known encrypted value. The resulting plaintext value is compared to a known plaintext value. If the decrypted texts match, the test passes; otherwise, the test fails.
- RSA Signature Known Answer Test (both signature/verification): This test takes a known plaintext value and Private/Public key pair and used the public key to encrypt the data. This value is compared to a known encrypted value. If the encrypted values, the test passes; otherwise, the test fails. The encrypted data is then decrypted using the private key. This value is compared to the original plaintext value. If the decrypted values match, the test passes; otherwise, the test fails.
- RNG/DRBG Known Answer Test: For this test, known seed values are provided to the DRBG implementation. The DRBG uses these values to generate random bits. These random bits are compared to known random bits. If the random bits match, the test passes; otherwise, the test fails.
- HMAC Known Answer Test: For each of the hash values listed, the HMAC implementation is fed known plaintext data and a known key. These values are used to generate a MAC. This MAC is compared to a known MAC. If the MAC values match, the test passes; otherwise, the test fails.
- Software Integrity Test: The Software Integrity Test is run automatically whenever the IOS-XE system image is loaded and confirms that the image file that's about to be loaded has maintained its integrity. The software contains a SHA-512 hash. This hash is compared to a pre-loaded hash. If the hash values match, the test passes; otherwise, the test fails.
- SHA-1/256/512 Known Answer Test: For each of the values listed, the SHA implementation is fed known data and a key. These values are used to generate a hash. This hash is compared to a known value. If the hash values match, the test passes; otherwise, the test fails.

If any component reports failure for the POST, the system crashes. Appropriate information is displayed on the screen and saved in the crashinfo file.

All ports are blocked during the POST. If all components pass the POST, the system is placed in FIPS PASS state and ports can forward data traffic.

If an error occurs during the self-test, a SELF_TEST_FAILURE system log is generated.

Example Error Message: _FIPS-2-SELF_TEST_IOS_FAILURE: "IOS crypto FIPS self-test failed at %s."

Explanation FIPS self test on IOS crypto routine failed.

These tests are sufficient to verify that the correct version of the TOE software is running as well as that the cryptographic operations are all performing as expected because any deviation in the TSF behaviour will be identified by the failure of a self-test.

Component Guidance Assurance Activities: 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.

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.

Section “Power-on Self-Tests Run During Bootup and Normal Operation” of the Admin Guide provides a summary of the self-tests. This list matches the ST. After the list of tests, the Admin Guide explains what to do if any of the tests fail:

- If possible, review the crashinfo file. This will provide additional information on the cause of the crash.
- Restart the TOE to perform POST and determine if normal operation can be resumed.
- If the problem persists, contact Cisco Technical Assistance via http://www.cisco.com/techsupport or 1 800 553-2447.
- If necessary, return the TOE to Cisco under guidance of Cisco Technical Assistance.

Component Testing Assurance Activities: 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 fulfill any of the SFRs.

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

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

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

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

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

During a reboot of the TOE, the evaluator confirmed that the TOE performed self tests to verify the firmware integrity and the cryptographic functions. The evaluator observed the output of these tests indicate that they were successful. The firmware integrity test passed and all other tests were successfully completed with no errors.

2.5.8 TRUSTED UPDATE (NDCPP22E:FPT_TUD_EXT.1)

2.5.8.1 NDCPP22E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.8.2 NDCPP22E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.8.3 NDCPP22E:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

Section 6.1 (FPT_TUD_EXT.1) of the ST states that an Authorized Administrator can query the software version running on the TOE and can initiate updates to (replacements of) software images. The current active version can be verified by executing the “show version” command from the TOE’s CLI. When software updates are made available by Cisco, an Administrator can obtain, verify the integrity of, and install the updates. The updates can be downloaded from software.cisco.com.

A digital signature mechanism is used to verify software files (to ensure they have not been modified from the originals distributed by Cisco) before loading. If the integrity check fails, the software is not loaded and the system reboots to attempt the test again. If the test continues to fail, the Authorized Administrator must contact Cisco. If the integrity check is successful, the software is loaded and the device continues with the bootup process.

To verify the digital signature prior to installation, the “show software authenticity file” command displays software authentication related information that includes image credential information, key type used for verification, signing information, and other attributes in the signature envelope, for a specific image file. If the output from the “show software authenticity file” command does not provide the expected output, contact Cisco TAC.

Once the integrity check is complete, the power-on self-tests are executed. If the power-on self-tests are successful, the TOE continues to load into an operational state. If a power-on self-test fails, the TOE automatically reboots to attempt to clear the error state. The TOE will continue to reboot until the error is cleared and the device is operational. If the error persists, the Authorized Administrator must contact Cisco.

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

The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall

include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

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

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 that takes place when applying updates.

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

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

Section “Secure Acceptance of the TOE” of the Admin Guide, steps 7 and 9, provide instructions for how to download and verify an image prior to running it on the TOE. The Admin Guide provides detailed instructions for each step in the process. It also says if the verification fails, the image will not be loaded.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

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

b) Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

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

2) An image that has not been signed

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

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

c) Test 3 [conditional]: If the TOE itself verifies a hash value over an image against a published hash value (i.e. reference value) that has been imported to the TOE from outside such that the TOE itself authorizes the installation of an image to update the TOE, the following test shall be performed (otherwise the test shall be omitted). If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

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

2) The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value to the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to

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

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

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

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

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

Test 1: The evaluator verified the current TOE version and then followed guidance procedures to install a new image to the TOE. The evaluator then verified the TOE version again and confirmed that it was updated successfully.

Test 2: For each device, the evaluator used legitimate updates that were modified in three ways:

1. Corrupted Image/Valid Signature - A few bytes in the update file is modified via a hex editor
2. Valid Image/No Signature - Update is missing a signature
3. Valid Image/Invalid Signature - Update's signature is corrupted

Attempts to update with each of these modified images failed as expected.

Test 3: Not applicable. The does not verify the integrity of updates using published hashes.

2.6 TOE ACCESS (FTA)

2.6.1 TSF-INITIATED TERMINATION (NDCPP22E:FTA_SSL.3)

2.6.1.1 NDcPP22e:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FTA_SSL.3) of the ST states the allowable inactivity timeout range is from 1 to 65,535 seconds. Administratively configurable timeouts are also available for the EXEC level access (access above level 1) through use of the “exec-timeout” setting.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

Section “Session Termination” of the Admin Guide discusses remote session termination. It provides instructions for setting termination values for inactive sessions.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following test:

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

The evaluator performed a test that demonstrates operation of inactivity timeouts over SSH with timeout values of 1 minute and 2 minutes. For each time period, the evaluator recorded the time immediately prior to login, logged in, and performed no further actions. When the TOE terminated the SSH session for inactivity the evaluator recorded the time again. The period of time observed matched the configured timeout values.

2.6.2 User-initiated Termination (NDcPP22e:FTA_SSL.4)

2.6.2.1 NDcPP22e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how the local and remote administrative sessions are terminated.

Section 6.1 (FTA_SSL.4) of the ST states an authorized administrator is able to exit out of both local and remote administrative sessions. Each administrator logged onto the TOE can manually terminate their session using the "exit" or "logout" command.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Section "Remote Administration Protocols" of the Admin Guide states that to terminate a remote or local session to the switch, use the "exit" or "logout" command at the User or Privilege EXEC prompt to terminate the session.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following tests:

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

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

Test 1: The evaluator logged in to the local console and then typed in the command "exit". The evaluator observed that the session ended and a login prompt was presented.

Test 2: The evaluator repeated this test using an SSH connection and observed that the session ended and the SSH connection was terminated.

2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.6.3.1 NDcPP22e:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.1 (FTA_SSL_EXT.1) of the ST states an Authorized Administrator can configure maximum inactivity times individually for both local and remote administrative sessions using the "session-timeout" setting applied to the console and virtual terminal (VTY) lines. The allowable inactivity timeout range is from 1 to 65,535 seconds. If a local user is inactive for a configured period, the session will be terminated and will require reidentification and authentication to login. If a remote user session is inactive for a configured period, the session will be terminated and will require re-identification and authentication to establish a new session.

Component Guidance Assurance Activities: 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.

Section “Session Termination” of the Admin Guide discusses remote session termination. It provides instructions for setting termination values for inactive sessions.

Component Testing Assurance Activities: The evaluator shall perform the following test:

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

The evaluator performed a test that demonstrates operation of inactivity timeouts on the local console with timeout values of 1 minute and 2 minutes. For each time period, the evaluator recorded the time immediately prior to login, logged in, and performed no further actions. When the TOE terminated the local console session for inactivity the evaluator recorded the time again. The period of time observed matched the configured timeout values.

2.6.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.6.4.1 NDcPP22e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FTA_TAB.1) of the ST states the Administrator can configure an access banner that describes restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE. The banner will display on the local console port and SSH interfaces prior to allowing any administrative access.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Section "Login Banners" of the Admin Guide explains how to configure the login banner.

Component Testing Assurance Activities: The evaluator shall also perform the following test:

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

The evaluator configured a banner and verified that the banner was displayed appropriately for console and SSH CLI logins.

2.7 Trusted path/channels (FTP)

2.7.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1)

2.7.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 NDcPP22e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure

communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Section 6.1 (FTP_ITC.1) of the ST states the TOE protects communications with peer or neighbor switches using keyed hash as defined in FCS_COP.1.1/keyedhash and cryptographic hashing functions FCS_COP.1.1/hash. This protects the data from modification of data by hashing that verify that data has not been modified in transit. In addition, encryption of the data as defined in FCS_COP.1.1/DataEncryption is provided to ensure the data is not disclosed in transit.

MACsec is used to secure communication channels between MACsec peers at Layer 2. The TOE protects communication between the TOE and the remote audit server using IPsec. This provides a secure channel to transmit log events. Communications between the TOE and the AAA server are secured using IPsec.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

Section “IPsec Overview” of the Admin Guide explains how to establish an IPsec connection. See FCS_IPSEC_EXT.1 for specific configuration details. Section “IPsec Session Interruption/Recovery” explains that if an IPsec session with a peer is unexpectedly interrupted, the connection will be broken. In these cases, no administrative interaction is required. The IPsec session will be reestablished (a new SA set up) once the peer is back online.

Component Testing Assurance Activities: 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.

The evaluator shall perform the following tests:

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

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

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

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

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

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

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

Further assurance activities are associated with the specific protocols.

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.

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.

The TOE utilizes IPsec for trusted channels protecting communication with an external audit server (syslog server) and an external authentication server (RADIUS).

A successful TOE IPsec connection supporting communication to an external audit server was established. Examining the packet capture from that test, we see that the IPsec connection between the TOE component and the external syslog server was established, the TOE initiated the connection, and Application data transferred is encrypted (i.e., not plaintext).

The evaluator began a packet capture off traffic between the TOE and external audit server. With the connection established, the evaluator physically disconnected the network between the TOE and the remote audit server. The evaluator left the network disconnected several minutes then reconnected the wiring. Because the TOE automatically reconnects broken IPsec connections, the evaluator waited for the syslog server to begin receiving audit data again and stopped the packet capture shortly after traffic began flowing after the disruption. The evaluator observed that no data was transmitted unprotected.

The evaluator also used the TOE to initiate an IPsec protected communication pathway to an external authentication server. Examination of the packet capture obtained during this activity showed that the connection was protected by IPsec, the TOE initiated the connection, and all application data was transferred encrypted (i.e., not plaintext). The evaluator also performed the same physical disruption test during this test and observed that no data was transmitted unprotected.

Upon completion of these activities, the resulting transcripts and packet captures were inspected. This data showed the following:

Test 1: The TOE support for IPsec protected syslog and RADIUS was demonstrated.

Test 2: The TOE initiated the IPsec connection for both syslog and RADIUS trusted channels.

Test 3: Syslog and RADIUS communication were not sent in plaintext.

Test 4: A physical disruption in the network resulted in an IPsec session interruption and no data was transmitted unprotected upon resumption.

2.7.2 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin)

2.7.2.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FTP_TRP.1/Admin) of the ST states all remote administrative communications take place over a secure encrypted SSHv2 session. The SSHv2 session is encrypted using AES encryption. The remote users can initiate SSHv2 communications with the TOE.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Section “Remote Administration Protocols” of the Admin Guide explains how to establish an SSH connection for remote administration. See FCS_SSHS_EXT.1 for specific configuration details.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

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

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

Further assurance activities are associated with the specific protocols.

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.

The TOE offers remote administration via SSHv2 or SSH tunneled through IPsec to provide the trusted path (with protection from disclosure and modification) for all remote administration sessions.

The evaluator found in FCS_SSHS_EXT.1.2-t1 that the SSH connection is not sent in plaintext.

The evaluator found in FTP_ITC.1-t4 that the IPsec connection is not sent in plaintext.

The TOE is not distributed.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

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

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

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

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

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

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

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

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.

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

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

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

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

For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps (per TD0536):

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

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

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

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

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

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

Section “Enabling FIPS Mode” of the Admin Guide states the TOE must be run in the FIPS mode of operation. The use of the cryptographic engine in any other mode was not evaluated nor tested during the CC evaluation of the TOE. Instructions for configuring algorithms are provided with each protocol.

Section “Operational Environment” of the Admin Guide provides information for the Operational Environment to be CC compliant as well as information on what functionality is excluded from the evaluation.

Section “Product Updates” of the Admin Guide describes the verification and update process. It references Section 2.5, steps 7 and 9, which provides step by step instructions for installing and verifying an image.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

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

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

In addition, the evaluator performs the EAs specified below.

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

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

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.

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

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

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

The preparative procedures must

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

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

The evaluator had the Admin Guide to use when configuring the TOE. The completeness of the Admin Guide is addressed by its use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a 'hard label' (e.g., stamped into the metal, paper label) or a 'soft label' (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

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

The evaluator verified that the ST, TOE, and Admin Guide are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

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

See section 3.3.1 for an explanation of how all CM items are addressed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes 'evaluated configuration' instructions). The focus of the testing is to confirm that the requirements specified in Section 5.1.7 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

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

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

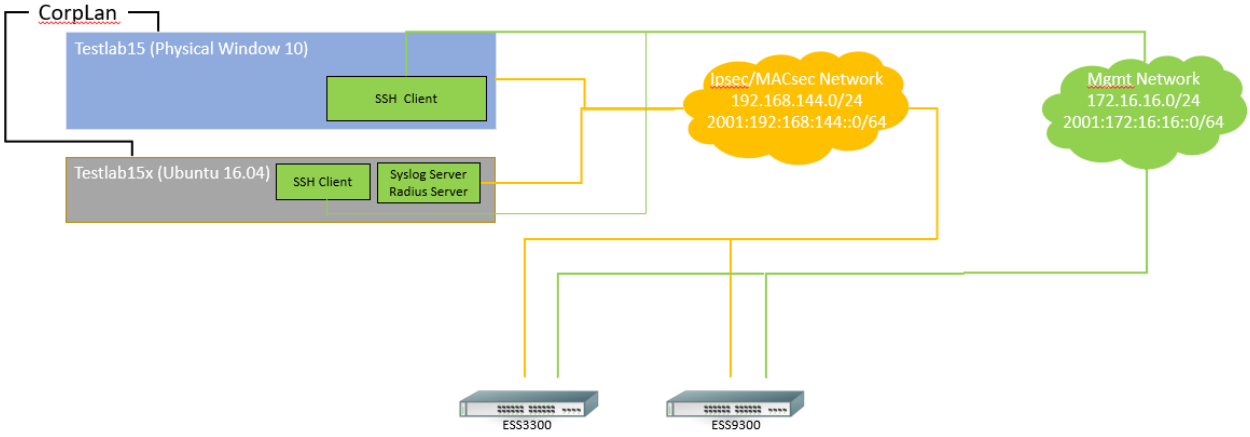
The evaluator should consult Appendix B when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

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

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The test configuration consisted of the following TOE platforms along with supporting products:

- ESS-3300-NCP
- ESS-3300-CON (used in testing)
- ESS-3300-24T-NCP
- ESS-3300-24T-CON
- ESS-9300-10X-E (used in testing)

Figure 1 of the DTR depicts the test network used during evaluation testing. The figure shows two Devices Under Test (DUTs), which represents the location of the TOEs in the network.



The Gossamer Test servers utilized both a windows and Ubuntu environment. The Windows supporting software included the following.

- Windows 10.0
- Wireshark version 3.4.8
- Windows SSH Client – Putty version 0.73 (used to connect to device console and SSH)
- Tftpd64.exe v4.60

The Gossamer Test servers with an Ubuntu environment included the following tools.

- Openssl version 1.0.2g

- Openssh client version 7.2p2
- Rsyslog version 8.16.0
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Nmap version 7.01
- Stunnel 5.30
- Strongswan v5.3.5
- FreeRADIUS Version 3.0.15
- Scapy version 2.4.5
- Wpa_supplicant v2.10

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

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

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. 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.

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

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

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.

The developer shall provide documentation identifying the list of software and hardware components7 that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This

additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

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

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

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

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

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

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuss testing. None of the public search for vulnerabilities or the fuzz testing uncovered any residual vulnerability.

The evaluator searched the following sources for vulnerabilities on 09/20/23

- National Vulnerability Database (https://web.nvd.nist.gov/vuln/search),
- Vulnerability Notes Database (http://www.kb.cert.org/vuls/),
- Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities),
- Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories),
- Exploit / Vulnerability Search Engin (http://www.exploitsearch.net),
- SecurITeam Exploit Search (http://www.securiteam.com),
- Offensive Security Exploit Database (https://www.exploit-db.com/)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search),

Each site was searched using the following terms:

- “Cisco IOS XE”
- “Cisco Embedded Services”
- “ESS-3300-NCP”
- “ESS-3300-CON”
- “ESS-3300-24T-NCP”
- “ESS-3300-24T-CON”
- “ESS-9300-10X-E”

- “Xilinx ZU3EG”
- “Broadcom BCM54194”
- “CrayCore”
- “MSC MACsec”
- “IOS Common Cryptographic Module”
- “IC2M”

3.5.2 Additional Flaw Hypothesis (AVA_VAN.1)

Assurance Activities: The following additional tests shall be performed:1.) [Conditional]: If the TOE is a TLS server and supports ciphersuites that use RSA transport (e.g. supporting TLS_RSA_WITH_* ciphers) the following test shall be performed. Where RSA Key Establishment schemes are claimed and especially when PKCS#1 v1.5* padding is used, the evaluators shall test for implementation flaws allowing Bleichenbacher and Klima et al. style attacks, including Bock et al's ROBOT attacks of 2017 in the flaw analysis. Even though Bleichenbacher's original paper is two decades old, Bock et al. found these attacks to still be effective in weakening the security of RSA key establishment in current products. Bleichenbacher and Klima et al. style attacks are complex and may be difficult to detect, but a number of software testing tools have been created to assist in that process. The iTC strongly recommends that at least one of the tools mentioned in Bock et al's ROBOT attacks of 2017 webpage or paper, as effective to detect padding oracle attacks, be used to test TOE communications channels using RSA based Key Establishment (related sources: http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf, https://eprint.iacr.org/2003/052, https://robotattack.org/). Network Device Equivalency Consideration.

The TOE does not support a TLS server implementation and therefore is not subject to Bleichenbacher attacks.