www.GossamerSec.com

Assurance Activity Report for Hewlett Packard Enterprise's CX-5420, CX-6200M, CX-6300, CX-6400, CX-8360, and CX-9300S Switch Series MACsec running AOS-CX Version 10.16

Version 0.2
08/17/26

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

Revision History

Revision	Date	Authors	Summary
Version 0.1	07/28/26	Gossamer	Initial draft
Version 0.2	08/17/26	Gossamer	Addressed ECR comments

T The TOE Evaluation was Sponsored by:

Aruba, a Hewlett Packard Enterprise Company
8000 Foothills Blvd.
Roseville, CA 95747

Evaluation Personnel:

- William Micknick
- Patrick Dones

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 Hewlett Packard Enterprise's CX-5420, CX-6200M, CX-6300, CX-6400, CX-8360, and CX-9300s Switch Series MACsec running AOS-CX version 10.16 NDcPP30e/MACSEC10 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

This section provides equivalence arguments for the Common Criteria testing as well as Cryptographic CAVP testing.

1.1.1 Evaluated Platform Equivalence

This section explains why the test subset was adequate to address all product installations. As shown in [ST] the evaluated configuration includes 7 models. The evaluation team performed functional NDcPP30e & MACSEC10 testing against 5 of those models. The two models not tested utilized the same software image, processor, network cards, and embedded MACsec hardware as one of the models that was tested. Therefore, the models are considered equivalent to a device that was tested.

The Green cells in the following table show the device which were tested during the evaluation.

Model	Software Image ID	Processor ID	Microarchitecture	Network Interfaces	Embedded MACsec Hardware
1. CX-5420	BL	AMD Ryzen V1500B	AMD Zen	Broadcom	BCM 82756 BCM 82399
2. CX-6200M	ML	NXP 1046A	ARM Cortex-A72	Broadcom	BCM 54192 BCM 54998SM BCM 82759
3. CX-6300M 4. CX-6300F	FL	NXP 1046A	ARM Cortex-A72	Broadcom	BCM 84898M BCM 82399 BCM 54998SM BCM 82756 BCM 82759 BCM 81343
5. CX-6400	FL	NXP 1046A	ARM Cortex-A72	Broadcom	BCM 81343
6. CX-8360	LL	NXP 1046A	ARM Cortex-A72	Broadcom	BCM 82399 BCM 82398

7. CX-9300S	NL	AMD Ryzen V3C48	AMD Zen 3+	Broadcom	BCM 81343

1.1.2 CAVP CERTIFICATES

The TOE includes the Hewlett Packard Enterprise OpenSSL 3 Provider version 3.1.4a to perform cryptographic operations. The network interface chipsets perform the AES-GCM data encryption for MACsec. The following functions have been CAVP certified.

Functions	Requirement	Standard	Certificate #
Encryption/Decryption			
AES CBC (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A	A4803
AES-CTR (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A	A4803
AES GCM (128 and 256 bits)	FCS_COP.1/DataEncryption	ISO 19772 FIPS Pub 197 NIST SP 800-38D	A4803
Cryptographic hashing			
SHA-256, SHA-384, SHA-512 (digest sizes 256, 384 and 512 bits)	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	A4803
Keyed-hash message authentication			
HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (key and digest sizes of 256, 384 and 512 bits)	FCS_COP.1/KeyedHash	FIPS Pub 198-1 FIPS Pub 180-4 ISO/IEC 9797-2:2011	A4803
Cryptographic signature services			
RSA Digital Signature (rDSA) (2048, 3072 bits)	FCS_COP.1/SigGen	FIPS Pub 186-4	A4803
ECDSA Digital Signature (P-256, P-384, P-521)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 14888-3	A4803

Functions	Requirement	Standard	Certificate #
Random bit generation			
CTR_DRBG(AES) with sw based noise sources with a minimum of 256 bits of entropy	FCS_RBG_EXT.1	NIST SP 800-90A ISO/IEC 18031:2011	A4803
Key generation			
RSA Key Generation (2048-bit)	FCS_CKM.1	FIPS Pub 186-4 ISO/IEC 9796-2	A4803
ECC Key Generation (P-256, P-384, P-521)	FCS_CKM.1	FIPS PUB 186-4	A4803
Key establishment			
RSA	FCS_CKM.2	RSAES-PKCS1-v1_5	Tested with known good implementation
KAS ECC P-256, P-384, P-521	FCS_CKM.2	NIST SP 800-56A Rev 3	A4803
MACsec			
AES-CMAC 128 & 256 bits	FCS_COP.1/CMAC	SP 800-38B	A4803
AES Key Wrap 128 & 256 bits	FCS_COP.1/MACSEC	SP 800-38F (wrap)	A4803
AES GCM 128 & 256 bits	FCS_COP.1/MACSEC	ISO 19772 NIST SP 800-38D	C1877, C1869, AES 4550, AES 4545, AES 4544

1.2 References

The following evidence was used to complete the Assurance Activities:

[ST]	Hewlett Packard Enterprise's CX-5420, CX-6200M, CX-6300, CX-6400, CX-8360, and CX-9300s Switch Series MACsec running AOS-CX version 10.16 Security Target
[AGD]	Common Criteria Administrator Guidance, Target of Evaluation: 4100, 5000, 6000, 8000, 9000, and 10000 Switch Series

2. Protection Profile SFR Assurance Activities

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP30e:FAU_GEN.1)

2.1.1.1 NDcPP30e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP30e: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 shall 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 of the ST states for the administrative tasks of generating/importing/deleting cryptographic keys, the TOE logs the type of key and its SHA256 hash as the key identifier. The TOE is not distributed.

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.

The section entitled "List of Auditable Events (as Mandated by the NDcPPe)" in the [AGD] provides an example of each auditable event required by FAU_GEN.1. During testing, the evaluator mapped the entries in the tables in this section to the TOE generated events, showing that the section provides examples/descriptions of all required audit events.

The evaluator verified the administrative commands when performing all other guidance AAs. Specific references to commands can be found throughout 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 identify and authentication (I&A) mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

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 event when running the other security functional tests described by the protection profiles. For example, the required event for FPT_STM_EXT.1 is Changes to Time. The evaluator collected these audit records when modifying the clock using administrative commands. The evaluator then recorded these 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 Audit Data Generation (MACsec) (MACSEC10:FAU_GEN.1/MACSEC)

2.1.2.1 MACSEC10:FAU_GEN.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 MACSEC10:FAU_GEN.1.2/MACSEC

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: None Defined

Component Testing Assurance Activities: None Defined

2.1.3 User identity association (NDcPP30e:FAU_GEN.2)

2.1.3.1 NDcPP30e: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 NDcPP30e: 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 NDcPP30e: 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 FAU_GEN.1.

2.1.4 Protected Audit Event Storage (NDcPP30e:FAU_STG_EXT.1)

2.1.4.1 NDcPP30e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 NDcPP30e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.3 NDcPP30e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.4 NDcPP30e:FAU_STG_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.5 NDcPP30e:FAU_STG_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.6 NDcPP30e:FAU_STG_EXT.1.6

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

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).

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

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

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

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

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 of [ST] states that the TOE can be configured to export audit records to an external SYSLOG server. This communication is protected with TLS.

Section 6.1 of [ST] also states that the TOE is a standalone device that is able to generate and store audit records of security relevant events as they occur.

Section 6.1 of [ST] explains that the TOE supports storage of local audit records, visible through two CLI commands. The event log is visible using the command "show logging". The TOE is capable of rotating through a set of compressed log files, and will check periodically to determine whether or not to rotate its logs based upon log size. The TOE fills one file (“full” storage is defined as consumption of the total configured storage space within a log file), it will rotate the contents of the current log into a compressed file, while rotating previously compressed files until finally deleting the oldest compressed file.

Section 6.1 of [ST] states that TOE audit records are protected against unauthorized access by only allowing authorized administrators to have access to local audit logs and to perform deletion of logs. Once configured to export audit records, the TOE attempts to transmit all logs in real-time, will temporarily maintain unsent records in the event of a disrupted syslog connection, and sends those records when the remote audit server successfully reestablishes the connection. The TOE uses the TLS protocol to protect audit records transmitted to the external syslog server.

The TOE is not distributed.

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

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

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

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

The section entitled "Secure Remote Logging" in [AGD] explain how to configure and establish the trusted channel to the audit server. It states that all audit events that are generated are logged locally and also sent to all configured syslog servers.

This section indicates that the TOE will attempt to transmit the log to the syslog server at the same time it is generated locally. It also indicates that the syslog server is expected to be compliant with RFC 5425 (TLS Transport Mapping for Syslog).

The section entitled "Audit log rotation" states the possible configuration options for log file size threshold and rotation frequency, and the resulting behavior of these possible configurations.

As the locally stored audit data cannot be modified or deleted by a non-administrative account, there is no configuration necessary to protect it from unauthorized access.

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

The evaluator shall perform the following additional test for this requirement:

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

b. Test 2: For distributed TOEs, Test 1 defined above shall be applicable to all TOE components that forward audit data to an external audit server.

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

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

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

d. Test 4: The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.5. Depending on the configuration this means that the evaluator shall check the content of the audit data when the audit data is just filled to the maximum and then verifies that

i. The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option 'drop new audit data' in FAU_STG_EXT.1.5).

ii. The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option 'overwrite previous audit records' in FAU_STG_EXT.1.5)

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

e. Test 5: For distributed TOEs, for the local storage according to FAU_STG_EXT.1.4 ,Test 1 specified above shall be applied to all TOE components that store audit data locally. For all TOE components that store audit data locally and comply with FAU_STG_EXT.2, Test 2 specified above shall be applied. The evaluator shall verify that the transfer of audit data to an external audit server is implemented.

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

Note: The intent of the test is to ensure that the local audit TSF (as specified by FAU_STG_EXT.1.3) operates independently from the ability to transmit the generated audit data to an external audit server (as specified in FAU_STG_EXT.1.1). There are no specific requirements on the interruption of the connection between the TOE and the external audit server (as for FTP_ITC.1). (TD0886 applied)

Test 1: The evaluator configured the system (per guidance) to securely transfer audit data and performed some auditable administrative actions. The evaluator then captured network traffic between the TOE and the external audit server. The evaluator verified that the packet capture showed the audit data was not clear text on the network.

Test 2: Not Applicable. The TOE is not distributed

Test 3: The evaluator showed the TOE's populated local log file. The evaluator then initiated a power cycle on the TOE. After the power cycle, the evaluator showed the log file again, verifying that the log data persisted.

Test 4: Audit overflow is provided by the TOE through a log file rotation mechanism. When the event log file becomes "full", then the oldest file is deleted and new records are written into a new file. The evaluator generated audit data for each type of the log and verified that the log files were filled and rotated.

Test 5: Not Applicable. The TOE is not distributed.

Test 6: The evaluator presented the local and syslog log files. The evaluator then issued a command to disable connection between the syslog server and TOE. After the interruption, the evaluator issued a command to generate audit data. The evaluator then presented the local and syslog log files confirming that the audit data was recorded locally while confirming syslog server connection remained interrupted and failed to audit the event.

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation - per TD0921 (NDcPP30e:FCS_CKM.1)

2.2.1.1 NDcPP30e: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 6.2 of [ST] indicates that the TOE can generate ECDSA SSH host keys of size P-256, P-384 and P-521. It can also generate RSA, DH and ECDHE asymmetric keys as part of TLS key establishment during TLS negotiations.

For asymmetric key pairs used for key exchange, the TOE supports generating ephemeral ECDH keys and DH keys for the SSHv2 key exchange methods selected in FCS_SSHS_EXT.1.7. This implies that the TOE generates ephemeral 256/384-bit ECDH keys using ECC schemes for P-256/384 curves.

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.

The section entitled "Disabling Unsupported Algorithms" in [AGD] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. Section 6.2 of [ST] identifies the TLS ciphersuites supported by the TOE and indicates they are not configurable.

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 or FIPS PUB 186-5 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. This test must be repeated for each supported RSA modulo and generation method.

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

a) Random Provable Primes (p and q shall be provable primes).

b) Random Probable primes (p and q shall be probable primes).

c) Provable Primes with Conditions (p1, p2, q1, q2, p and q shall all be provable primes)

d) Provable/probable primes with Conditions (p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes)

e) Probable primes with Conditions (p1, p2, q1, q2, p and q shall all be probable primes)

The Random Provable primes, and all the Primes with Conditions can be tested in the same manner because each of these begin with a starting random number and calculate the p and q values from this value. The test instructs the TSF to generate intermediate values and the p, q, n, and d values. The evaluator then validates the correctness of the values generated by the TSF.

To test the key generation method for the Random Provable primes method or 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. If the TSF provides the input to the key generation function, such input must be recorded and verified. For each RSA key length (modulo) claimed, the evaluator shall generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing Key Pair values generated by the TSF with those generated using the same set of input values using a known good implementation.

The Random Probable primes must be tested in a different way because this test generates different random numbers, not related to each other, until the number satisfies the 'probably prime' requirements. This validation method requires two tests for Random Probable primes. These include the Known Answer Test and the Miller-Rabin probabilistic primality test.

To test the key generation method for the Random Probable primes, the known answer test must be used. The evaluator shall compare the results of a known good implementation with the TSF results for a set (of a corresponding size depending on modulus) of supplied values containing private prime factor p, private prime factor q and confirm all public keys, e, match. Then the evaluator shall use the TSF to generate prime p, q pairs for each modulus size and perform the Miller-Rabin tests.

(TD0921 applied)

Key Generation for Elliptic Curve Cryptography (ECC)

Key Generation for ECDSA Schemes

Key pairs for the ECDSA consist of pairs (d, Q), where the private key, d, is an integer, and the public key, Q, is an elliptic curve point.

The evaluator shall verify the implementation of ECC Key Generation by the TOE using the following components tests:

- ECC Key Pair Generation

- ECC Public Key Validation (PKV)

ECC Key Pair Generation Test

There are four methods by which these pairs may be generated:

-FIPS 186-4 Section B.4.1 Key Pair Generation Using Extra Random Bits

Using this method, 64 more bits are requested from the RBG than are needed for d so that bias produced by the mod function is negligible.

-FIPS 186-4 Section B.4.2 Key Pair Generation by Testing Candidates

Using this method, a random number is obtained and tested to determine that it will produce a value of d in the correct range. If d is out-of-range, another random number is obtained (i.e., the process is iterated until an acceptable value of d is obtained.

-FIPS 186-5 Section A.2.1 ECDSA Key Pair Generation using Extra Random Bits

Using this method, more bits are requested from the DRBG than are needed for d so that the bias produced by the mod function is negligible.

-FIPS 186-5 Section A2.2 ECDSA Key Pair Generation by Rejection Sampling

Using this method, a random number is obtained and tested to determine that it will produce a value of d in the correct range. If d is out of range, an ERROR is returned.

The evaluator shall test the ECC Key Pair Generation by having the TSF produce 10 key pairs (d, Q) for each implemented key generation method using each supported curve (i.e., P-256, P-384, and P-521). The steps are the same for each key generation method and curve. The private key, d, shall be generated using the output of an approved DRBG converted to an integer via modular reduction or the discard method. The known private key is then used by the TSF to compute the public key, Q'. To evaluator then validates the correctness by comparing the value Q' computed by the TSF to the public key, Q generated by a known good implementation.

(TD0921 applied)

ECC Public Key Verification (PKV) Test

The evaluator shall generate 12 key pairs (d, Q) for each selected curve, with 6 valid public keys, Q, using a known good implementation and 6 modified, Q', and determine whether the TSF can accurately detect these modifications. Q' should be otherwise valid but include at least one of the following errors: a) point X or Y not on the curve, b) point X or Y is too large for the field for the given curve. The evaluator encouraged to make sure that the modification does not inadvertently result in another valid public key (e.g., modifying Y and accidentally hitting a different point on the curve).

(TD0921 applied)

Key Generation for FIPS PUB 186-5

FIPS 186-5 Key Generation Test

For the Ed25519 curve, 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 186-5 Key Verification Test

For the Ed25519 curve, 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.

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

2.2.2 Cryptographic Key Establishment (NDcPP30e:FCS_CKM.2)

2.2.2.1 NDcPP30e: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 as shown in the table below. 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.

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

Table 6 within section 6.2 of [ST] indicates that the TOE supports RSA key generation using 2048-bit keys, and ECC key generation using curves P-256, P-384 and P-521. These can be used to generate keys for use with a Certificate Signing Request, as well as in support of key establishment methods identified by Table 7.

Table 7 indicates that the TOE supports key establishment using RSA, ECDHE, FFC Schemes using 'safe-primes' and FFC schemes, which is consistent with the NDcPP30e:FCS_CKM.2 selections.

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).

The section entitled "Disabling Unsupported Algorithms" in [AGD] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes key exchange algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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 for ECC and FIPS186-type. 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 shall also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function 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.

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

The FFC Schemes using safe-primes was tested against the public implementation of these schemes refer to FTP_TRP.1/Admin and FTP_ITC.1 for this testing.

2.2.3 Cryptographic Key Destruction (NDcPP30e:FCS_CKM.4)

2.2.3.1 NDcPP30e: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 shall examine the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator shall confirm that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys

for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for2). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator shall check that this is consistent with the operation of the TOE.

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 shall examine the TSS to ensure that it describes how that pattern is obtained and used, and that this justifies the claim that the pattern does not contain any CSPs.

Section 6.2, Table 8, in the ST identifies the Key or CSP, where it is stored, when it is zeroized and how the zeroization is performed.

Section 6.2 of the ST states keys are zeroized when they are no longer needed by the TOE, and additionally, the TOE saves keys to persistent storage. Whether saving or destroying keys, the TOE delays the operation at the physical layer until the administrator issues the “write memory” command, which saves the running configuration to the startup configuration.

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).

Key destruction for SSH host keys and X509 certificate private keys is performed using the '`erase all zeroize`' command which is described in the section entitled "SSH host key generation" and "Certificate Installation and Validation" of [AGD].

Component Testing Assurance Activities: None Defined

2.2.4 Cryptographic Operation (AES-CMAC Keyed Hash Algorithm) (MACSEC10:FCS_COP.1/CMAC)

2.2.4.1 MACSEC10:FCS_COP.1.1/CMAC

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.

Section 6.2 of [ST] states that the TOE supports keyed-hash message authentication in accordance with AES-CMAC algorithm with key sizes 128 bits and 256 bits, the message digest size (output size) of 128 bits and block size of 128-bits. The algorithm conforms to NIST SP 800-38B.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 1: 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 eight arbitrary key-plaintext tuples that will result in the generation of a known MAC value when encrypted. The evaluator shall then verify that the correct MAC was generated in each case.

Test2: 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 shall 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

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

2.2.5 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP30e:FCS_COP.1/DataEncryption)

2.2.5.1 NDcPP30e: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, Table 6 in [ST] identifies the support of CBC and CTR modes of AES as available ciphers for SSH and GCM mode for TLS ciphersuites (all with both 128 and 256-bit keys) on the TOE.

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.

The section entitled "Disabling Unsupported Algorithms" in [AGD] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH cipher algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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 if the TSF is validated against the requirements of the Functional Package for Secure Shell referenced in Section 2.2 of the cPP. 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 section entitled, "CAVP Certificates" earlier in this document.

2.2.6 Cryptographic Operation (Hash Algorithm) (NDcPP30e:FCS_COP.1/Hash)

2.2.6.1 NDcPP30e: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 6.2 of the ST states the TOE uses the SHA-256, 384, and 512 hashing algorithms as part of SSHv2 integrity algorithms (see FCS_SSHS_EXT.1.6) and TLS ciphersuites. The TOE also uses SHA-256 during verification of a new image (trusted updates).

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.

The section entitled "Disabling Unsupported Algorithms" in [AGD] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH HMAC algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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 evaluator shall 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 evaluator shall 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 evaluator shall 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 evaluator shall 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 evaluator shall 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 evaluator shall 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 section entitled, "CAVP Certificates" earlier in this document.

2.2.7 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP30e:FCS_COP.1/KeyedHash)

2.2.7.1 NDcPP30e: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.

Table 6 in Section 6.2 of the ST states the TOE uses HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA512 for SSHv2 and TLS and provides the hash algorithm, key size, block size, and output MAC length used for each.

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.

The section entitled "Disabling Unsupported Algorithms" in [AGD] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH HMAC algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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 section entitled, "CAVP Certificates" earlier in this document.

2.2.8 Cryptographic Operation (MACsec AES Data Encryption and Decryption) - per TD0728 (MACSEC10:FCS_COP.1/MACSEC)

2.2.8.1 MACSEC10:FCS_COP.1.1/MACSEC

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 PP-Module in addition to the ones already required by the NDcPP in FCS_COP.1/DataEncryption.

Section 6.2 of [ST] states that the TOE performs AES key wrap with AES-GCM. AES is specified in ISO 18033-3, AES Key Wrap is specified in NIST SP 800-38F, GCM is specified in ISO 19772.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform testing for AES-GCM as required by the NDcPP in FCS_COP.1/DataEncryption.

In addition to the tests specified in the NDcPP for other iterations of FCS_COP.1, the evaluator shall perform the following tests:

Test 3: KW-AE Test: To test the authenticated encryption capability of AES key warp (KW), the evaluator shall provide five sets of 100 messages and keys to the TOE for each key length supported by the TSF. 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.

Test 4: KW-AD Test: To test the authenticated decryption capability of AES KW, the evaluator shall provide five sets of 100 ciphertext values and keys to the TOE for each key length supported by the TSF. 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

(TD0728 applied)

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

2.2.9 Cryptographic Operation (Signature Generation and Verification) - per TD0921 (NDcPP30e:FCS_COP.1/SigGen)

2.2.9.1 NDcPP30e: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 6.2 of the ST explains the TOE supports both RSA and ECDSA signing and verification. The TOE verifies RSA signatures on firmware updates and supports ECDSA authentication during SSH.

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.

The section entitled "Disabling Unsupported Algorithms" in [AGD] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH host-key and public-key algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

Component Testing Assurance Activities: ECDSA Signature Algorithm Tests

The evaluator shall verify the implementation of ECDSA Signature generation by the TOE using the Signature Generation Test. This test validates the TSF generation of the (r, s) pair that represent the digital signature. The digital signature shall be verified using the same domain parameters and hash function that were used during signature generation. An approved hash function or an XOF shall be used for this purpose.

Signature Generation Test

The purpose of this test is to verify the ability of the TSF to produce correct signatures.

To test signature generation, the evaluator supplies 10 pseudorandom messages to the TSF and a key pair, (d, Q), generated by a known good implementation. Exercising each applicable curve (i.e., P-256, P 384, or P-521) and hash algorithm or extendable-output function combination, the TSF generates signatures for each supplied

message and returns the corresponding signatures. Using a known-good implementation, the evaluator validates the signatures by using the associated public key, Q, to verify the signature.

Signature Verification Test

The purpose of this test is to verify the ability of the TSF to accept valid signatures and reject invalid signatures.

For each curve/hash algorithm or extendable-output function combination supported by the TSF, the evaluator shall use a known good implementation to generate a key pair, (d, Q), and use known good implementation with the private key, d, to sign 15 pseudorandom messages of 1024 bits. The evaluator shall alter some of the messages or signatures so that signature verification should fail.

To test signature verification, the evaluator supplies the public key, Q, and 15 pseudorandom messages, including altered messaged, to the TSF for verification of signatures. The evaluator shall then verify that the TSF validates correct signatures on the original messages and flags or rejects the altered messages.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature generation by the TOE using the Signature Generation Test. This test verifies the ability of the TSF to produce correct signatures.

There are 2 different RSA Signature algorithms that can be implemented. These include:

a. RSASSA-PKCS1-v1.5

b. RSASSA-PSS

To test signature generation, the evaluator generates or obtains 10 messages for each modulus size/hash or extendable-output function combination supported by the TOE. Using a key generated by a known good implementation, the TSF generates and returns the corresponding signatures to a known good implementation that validates the signatures by using the associated public key to verify the signature.

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

The evaluator shall verify the implementation of RSA Signature verification by the TOE using the Signature Verification Test. This test verifies the ability of the TSF to recognize valid and invalid signatures.

There are 2 different RSA Signature algorithms that can be implemented. These include:

a. RSASSA-PKCS1-v1.5

b. RSASSA-PSS

For each modulus size/hash or extendable-output function combination supported by the TOE, the evaluator shall use a known good implementation to generate a modulus and three associated key pairs, (d, e). Each private key d is used to sign six pseudorandom messages each of 1024 bits. Some of the public keys, e, messages, IR format, or signatures must be altered so that signature verification should fail. The modifications must cover distinct 'key modified', 'message modified', 'signature modified', 'IR moved', and 'trailer moved' tests. The modulus, hash or extendable-output algorithm, public key e values, messages, and signatures are forwarded to the TSF. The TSF then attempts to verify the signatures and returns the results. The evaluator then compares the received results with the stored results from a known good implementation.

The evaluator shall verify that the TSF validates correct signatures on the original messages and flags or rejects the altered messages.

(TD0921 applied)

The TOE has been CAVP tested. Refer to the ACVP certificates identified in Section 1.1, “CAVP Certificates.”

2.2.10 HTTPS Protocol (NDcPP30e:FCS_HTTPS_EXT.1)

2.2.10.1 NDcPP30e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.2 NDcPP30e:FCS_HTTPS_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 and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Section 6.2 of [ST] states that an HTTPS/TLS connection is available which presents Web GUI and Rest API administrative interfaces. The TOE implements HTTPS per RFC 2818. A connection can be established only if the peer initiates the connection.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to configure TOE for use as an HTTPS client or HTTPS server.

The TOE automatically uses HTTPS on its TLS interfaces and requires no additional configuration beyond that needed to establish the trusted channels.

Component Testing Assurance Activities: This test is now performed as part of FIA_X509_EXT.1/Rev testing.

Tests are performed in conjunction with the TLS evaluation activities.

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

See FIA_X509_EXT.1/Rev.

2.2.11 MACsec - per TD0884 (MACSEC10:FCS_MACSEC_EXT.1)

2.2.11.1 MACSEC10:FCS_MACSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 MACSEC10:FCS_MACSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.3 MACSEC10:FCS_MACSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.4 MACSEC10:FCS_MACSEC_EXT.1.4

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 ability of the TSF to implement MACsec in accordance with IEEE 802.1AE-2018. 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 the claimed EtherTypes are accepted by the MACsec interface. Where the SFR selection has provided additional EtherType support, the TSS must describe the usage of each of those frame types within the context of MKA, MACsec, and MAC control frame support. (TD0884 applied)

Section 6.2 of [ST] states the following on the TOE's MACsec implementation: The TOE implements MACsec in accordance with IEEE 802.1AE-2018. The TOE derives a Secure Channel Identifier (SCI) from a peer's MAC address and port data to uniquely identify the originator of a MACsec Protocol Data Unit (MPDU) and rejects any MPDUs that do not contain the identifier. Once configured on an interface, only EAPOL (PAE EtherType 88-8E), MACsec Ethernet frames (EtherType 88-E5) and MAC control frames are permitted and others are rejected.

Component Guidance Assurance Activities: If the TOE requires enabling or disabling specific EtherTypes to operate within the evaluated configuration, the guidance documentation must provide this information to the administrator. (Per TD0884)

The TOE does not require enabling or disabling specific EtherTypes to operate within the evaluated configuration.

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

Test 5: 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 6: The evaluator shall send Ethernet traffic to the TOE's MAC address that iterates through the full range of supported EtherType values (refer to List of Documented EtherTypes) and observes that traffic for all EtherType

values is discarded by the TOE except for the traffic which has an EtherType value described in the SFR. 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. (TD0884 applied)

Test 5: 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 6: 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 88-E5. MACsec and EAPOL EtherTypes can be seen throughout other MACsec tests.

2.2.12 MACsec Integrity and Confidentiality - per TD1030 (MACSEC10:FCS_MACSEC_EXT.2)

2.2.12.1 MACSEC10:FCS_MACSEC_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 MACSEC10:FCS_MACSEC_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.3 MACSEC10: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. This should include any confidentiality offsets used, the use of an ICV (including the supported length), and ICV generation with the SAK, using the SCI (or SSCI, when XPN is used) as the most significant bits of the initialization vector (IV) and an IV appropriate to the supported ciphersuites (least significant 32 bits as the IV in a non-XPN context, and a 32-bit SSCI and 64-bit packet number as the IV when XPN is used). (TD1030 applied)

Section 6.2 of [ST] states that the TOE implements MACsec with support for integrity protection with a confidentiality offset of 0, 30, 50. The TSF provides assurance of the integrity of protocol data units (MPDUs) using an Integrity Check Value (ICV) of 16 bytes derived with the Secure Association Key (SAK). The TOE provides the ability to derive an Integrity Check Value Key (ICK) from a CAK using a 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. The ICV is derived from the SCI and PN. This forms the 96-bit IV used by GCM.

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

Section "Configuration of Confidentiality" in the [AGD] describes the steps taken by the administrator to configure, apply, and remove the confidentiality offset to a macsec policy on the TOE:

switch(config-macsec-policy)# confidentiality [offset {0|30|50}]

To remove the confidentiality setting, the administrator can enter ‘no confidentiality’. In the evaluated configuration, this should only be done when changing a configuration.

The sub-section “confidentiality” under the MACsec commands section in the Security Guide contains additional information on this subject.

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

Test 7: 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, audit logs, or both that the frame bytes after the MACsec Tag values in the received traffic is not obviously predictable.

Test 8: 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 7: A valid test MACsec connection was performed in MACsec10: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 8: 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 - per TD0825 (MACSEC10:FCS_MACSEC_EXT.3)

2.2.13.1 MACSEC10:FCS_MACSEC_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.2 MACSEC10: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 6.2 of [ST] states the TOE generates unique Secure Association Keys (SAKs) using key derivation from Connectivity Association Key (CAK) per section 9.8.1 of IEEE 802.1X-2010, and the TOE's random bit generator as specified by FCS_RBG_EXT.1 such that the likelihood of a repeating SAK is no less than 1 in 2 to the power of the size of the generated key. The TOE generates unique nonce for the derivation of SAKs using the TOE's random bit generator as specified by FCS_RBG_EXT.1.

Component Guidance Assurance Activities: None Defined

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

Refer to other MACsec test cases, which show the TOE's MACsec capabilities.

2.2.14 MACsec Key Usage - per TD0803 & TD0825 (MACSEC10:FCS_MACSEC_EXT.4)

2.2.14.1 MACSEC10:FCS_MACSEC_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.2 MACSEC10:FCS_MACSEC_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.3 MACSEC10:FCS_MACSEC_EXT.4.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.4 MACSEC10:FCS_MACSEC_EXT.4.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.5 MACSEC10: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 PP-Module.

Section 6.2 of [ST] states the TOE supports peer authentication using only pre-shared keys. The TOE distributes SAKs between MACsec peers using AES key wrap as specified in FCS_COP.1/MACSEC. The TOE supports specifying a lifetime for CAKs. The TOE associates Connectivity Association Key Names (CKNs) with CAKs that are defined by the key derivation function using the CAK as input data (per 802.1X, section 9.8.1). The TOE associates Connectivity Association Key Names (CKNs) with CAKs. The length of the CKN is an integer number of octets, between 1 and 32 (inclusive).

Component Guidance Assurance Activities: If the method 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.

(conditional, the length of the CKN is configurable) The evaluator shall verify that the guidance describes how to set the CKN length to 1-32 octets. (TD0803 applied)

Section "MACsec Policy Configuration" in the [AGD] describes the configuration steps for enabling or disabling pre-shared keys, which is the only supported method of peer authentication under the evaluated configuration. The TOE does not support 802.2X EAP TLS. The CKN supports a range of 1 to 64 hexadecimal characters in the evaluated configuration; this is non-configurable.

Section "Configuration of CAK Lifetime" in the [AGD] gives the administrative commands to specify the lifetime of CAKs.

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

Test 9: 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 one minute and then disconnect the TOE from the peer and stop the sniffer. The evaluator

shall use the packet captures to verify that theSC was established via the selected mechanism and that the non-VLAN EtherType of the first data frame sent between the TOE and the peer is 88-E5.

Test 10: 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.

Test 9: 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 10: This test was performed in MACSEC10: FCS_MACSEC_EXT.4-t1, where the evaluator verified that the key is wrapped in the captured MKPDUs.

2.2.15 MACsec Key Agreement - per TD0825, TD0882 & TD0889 (MACSEC10:FCS_MKA_EXT.1)

2.2.15.1 MACSEC10:FCS_MKA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.2 MACSEC10:FCS_MKA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.3 MACSEC10:FCS_MKA_EXT.1.3

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 6.2 of [ST] states the TOE implements Key Agreement Protocol (MKA) in accordance with IEEE 802.1X-2010 and 802.1Xbx-2014. The TOE enables data delay protection for MKA. The TOE provides assurance of the integrity of MKA protocol data units (MKPDUs) using an Integrity Check Value (ICV) derived from an Integrity Check Value Key (ICK). The TOE provides the ability to derive an Integrity Check Value Key (ICK) from a CAK using a KDF. The TOE enforces an MKA Lifetime Timeout limit of 6.0 seconds and Hello Timeout limit of 2.0 seconds.

Guidance Assurance Activities: None Defined

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

Test 11: The evaluator shall transmit MKA traffic (MKPDUs) to the TOE from a MKA-capable peer in the operational environment. The evaluator shall verify via packet captures, audit logs, or both that the last 16 octets of the MKPDUs in the received traffic do not appear to be predictable.

Test 12: The evaluator shall transmit valid MKA traffic to the TOE from a 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 11: A successful MACsec connection was established in MACsec10:FCS_MACSEC_EXT.1-t1 above. The evaluator observed the ICV in the MKPDU packets and determined they were not obviously predictable.

Test 12: 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.

2.2.15.4 MACSEC10:FCS_MKA_EXT.1.4

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 13a: The evaluator shall configure the TOE to establish a MKA session with a new peer. The evaluator shall verify that the TOE sends a fresh SAK to the peer and sends other MKPDUs required for a new session. The evaluator shall verify from packet captures that MKPDUs are sent at least once every two seconds or every half-second, in accordance with the SFR selection.

Test 13b: (Conditional - If 'EAPTLS with DevIDs' is selected in FCS_MACSEC_EXT.4.1) The evaluator shall use EAP-TLS to derive a CAK and configure the TOE's peer to send '0' in the MKA parameter field for MACsec Capability (Table 11-6 in 802.1X-2020). The evaluator shall observe that the peer is deleted from the connection after MKA Life Time has passed.

Test 14a: (Conditional - if any 'group CAK' selection is made in FCS_MKA_EXT.1.5) The evaluator shall configure the TOE to send a fresh SAK with two peers as active participants. The evaluator shallverify that the TOE sends a fresh SAK to the peers and sends other MKPDUs required for a new session. The evaluator shall verify from packet captures that MKPDUs are sent at least once every two seconds or every half second in accordance with the SFR Selection.

Test 14b: (Conditional - if any 'group CAK' selection is made in FCS_MKA_EXT.1.5) Disconnect one of the peers. Arbitrarily introduce an artificial delay in sending a fresh SAK following the change in the Live Peer List. For this delayed fresh SAK, use a man-in-the-middle device to observe that the MKA Life Time of 6.0 seconds is enforced by the TSF.

(TD0882 applied)

Test 13a: The evaluator set up a MACsec peer 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 MACsec peers. The evaluator then started an MKA session between the TOE and the active participant peer, also taking a packet capture of the session. The evaluator analyzed the packet capture and noted that the TOE sends MKPDUs at least once two seconds.

Test 13b: Not applicable. The TOE does not support EAPTLS with DevIDs.

Test 14a: Not applicable. The TOE does not support Group CAKs.

Test 14b: Not applicable. The TOE does not support Group CAKs.

2.2.15.5 MACSEC10:FCS_MKA_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.6 MACSEC10:FCS_MKA_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.7 MACSEC10:FCS_MKA_EXT.1.7

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 6.2 of the [ST] explains TOE implements Key Agreement Protocol (MKA) in accordance with IEEE 802.1X-2010 and 802.1Xbx-2014. The TOE enables data delay protection for MKA. The TOE provides assurance of the integrity of MKA protocol data units (MKPDUs) using an Integrity Check Value (ICV) derived from an Integrity Check Value Key (ICK). The TOE provides the ability to derive an Integrity Check Value Key (ICK) from a CAK using a KDF. The TOE enforces an MKA Lifetime Timeout limit of 6.0 seconds and Hello Timeout limit of 2.0 seconds. The TOE behaves as a Key Server. The Key Server refreshes a SAK when it expires. The Key Server distributes a SAK by using a pairwise CAK. The pairwise CAK is derived from MKA or a pre-shared key. The Key Server refreshes a CAK when it expires. The Key Server distributes a fresh SAK whenever a member is added to or removed from the live membership of the CA.

The TOE validates MKPDUs according to 802.1X-2010, Section 11.11.2. In particular, the TOE discards without further processing any MKPDUs to which any of the following conditions apply:

a) The destination address of the MKPDU was an individual address.

b) The MKPDU is less than 32 octets long.

c) The MKPDU is not a multiple of 4 octets long.

d) The MKPDU comprises fewer octets than indicated by the Basic Parameter Set body length, as encoded in bits 4 through 1 of octet 3 and bits 8 through 1 of octet 4, plus 16 octets of ICV.

e) The CAK Name is not recognized.

If an MKPDU passes these tests, then the TOE begins processing it as follows:

a) If the Algorithm Agility parameter identifies an algorithm that has been implemented by the receiver, the ICV shall be verified as specified in IEEE 802.1X-2010 Section 9.4.1.

b) If the Algorithm Agility parameter is unrecognized or not implemented by the receiver, its value can be recorded for diagnosis but the received MKPDU shall be discarded without further processing.

Each received MKPDU that is validated as specified in this clause and verified as specified in 802.1X-2010, section 9.4.1 shall be decoded as specified in 802.1X-2010, section 11.11.4.

When Data Delay Protection (DDP) is enabled, MKA PDUs are exchanged every 0.5 seconds instead of every 2.0 seconds. The PN advertised by the peer is updated as the LPN in the Rx channel of the TOE.

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 Policy Configuration" in the [AGD] details the steps for MACsec policy configuration, including settings when the TOE acts as a key server.

Section "Key Server Priority" in the [AGD] gives instructions for configuring the TOE's MKA key server priority.

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:

Test 15: (Conditional - if any 'group CAK' selection is made in FCS_MKA_EXT.1.5) 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 three 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 an 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 16: The evaluator shall start an MKA session between the TOE and an environmental MACsec peer 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.

(TD0889 applied)

Test 15: Not applicable. The TOE does not support Group CAKs

Test 16: The evaluator set up a MACsec peer 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 MACsec peer. The evaluator then started an MKA session between the TOE and the active participant peer, 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 - per TD0990 (NDcPP30e:FCS_RBG_EXT.1)

2.2.16.1 NDcPP30e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.2 NDcPP30e: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.

Section 6.2 of [ST] indicates that the TOE instantiates it's AES-256 CTR_DRBG with a 384-bit seed (containing a minimum of 256 bits of entropy) from one software-based noise source.

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 TOE does not provide the ability to configure the 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 shall verify that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator shall

verify that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

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 Protocol - per TD0967 & TD1023 (SSH10:FCS_SSH_EXT.1)

2.2.17.1 SSH10:FCS_SSH_EXT.1.1

TSS Assurance Activities: The evaluator shall ensure that the selections indicated in the ST are consistent with selections in this and subsequent components. Otherwise, this SFR is evaluated by activities for other SFRs

N/A; this SFR is evaluated by activities for other SFRs.

Guidance Assurance Activities: There are no guidance evaluation activities for this component. This SFR is evaluated by activities for other SFRs.

N/A; There are no guidance evaluation activities for this component.

Testing Assurance Activities: There are no test evaluation activities for this component. This SFR is evaluated by activities for other SFRs

There are no test evaluation activities for this component.

2.2.17.2 SSH10:FCS_SSH_EXT.1.2

TSS Assurance Activities: The evaluator shall check to ensure that the authentication methods listed in the TSS are identical to those listed in this SFR component; and, ensure if password-based authentication methods have been selected in the ST then these are also described; and, ensure that if keyboard-interactive is selected, it describes the multifactor authentication mechanisms provided by the TOE.

Section 6.2 of [ST] states that the TOE implements the SSHv2 protocol, compliant to the following RFCs: 4251, 4252, 4253, 4254, 5656, 6668. The TOE supports public key-based and password-based authentication. The TOE allows use of the ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 algorithms for both host and user public key authentication. The TOE establishes a user identity when an SSH client presents a public key or correct password.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure the configuration options, if any, for authentication mechanisms provided by the TOE are described.

Section "Management Interfaces" in the [AGD] details the necessary configuration steps for setting up the console and SSH management interfaces. The section "SSH authorized keys" specifically describes the actions by an administrator to configure public key authentication.

Testing Assurance Activities: Test 1: [conditional] If the TOE is acting as SSH Server:

a. The evaluator shall use a suitable SSH Client to connect to the TOE, enable debug messages in the SSH Client, and examine the debug messages to determine that only the configured authentication methods for the TOE were offered by the server.

b. [conditional] If the SSH server supports X509 based Client authentication options:

a. The evaluator shall initiate an SSH session from a client where the username is associated with the X509 certificate. The evaluator shall verify the session is successfully established.

b. Next the evaluator shall use the same X509 certificate as above but include a username not associated with the certificate. The evaluator shall verify that the session does not establish.

c. Finally, the evaluator shall use the correct username (from step a above) but use a different X509 certificate which is not associated with the username. The evaluator shall verify that the session does not establish.

Test 2: [conditional] If the TOE is acting as SSH Client, the evaluator shall test for a successful configuration setting of each authentication method as follows:

a. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established.

b. Next, the evaluator shall use bad authentication data (e.g. incorrectly generated certificate or incorrect password) and ensure that the connection is rejected.

Steps a-b shall be repeated for each independently configurable authentication method supported by the server.

Test 3: [conditional] If the TOE is acting as SSH Client, the evaluator shall verify that the connection fails upon configuration mismatch as follows:

a. The evaluator shall configure the Client with an authentication method not supported by the Server.

b. The evaluator shall verify that the connection fails.

If the Client supports only one authentication method, the evaluator can test this failure of connection by configuring the Server with an authentication method not supported by the Client. In order to facilitate this test, it is acceptable for the evaluator to configure an authentication method that is outside of the selections in the SFR.

Test 1a.: The evaluator referred to FCS_SSH_EXT.1.2 and verified that only the configured authentication methods (password, publickey) were offered.

Test 1b.: The TOE does not support X509 based client authentication.

Test 2: The TOE does not act as an SSH Client, Not Applicable.

Test 3: The TOE does not act as an SSH Client, Not Applicable.

2.2.17.3 SSH10:FCS_SSH_EXT.1.3

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

Section 6.2 of the [ST] states the TOE's SSHv2 implementation limits SSH packets to a size of 262127 kilobytes. Anything larger will be dropped by the TOE.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test 1: The evaluator shall demonstrate that the TOE accepts the maximum allowed packet size.

Test 2: This test is performed to verify that the TOE drops packets that are larger than size specified in the component.

a. The evaluator shall establish a successful SSH connection with the peer.

b. Next the evaluator shall craft a packet that is slightly larger than the maximum size specified in this component and send it through the established SSH connection to the TOE. The packet should not be greater than the maximum packet size + 16 bytes. If the packet is larger, the evaluator shall justify the need to send a larger packet.

c. The evaluator shall verify that the packet was dropped by the TOE. The method of verification will vary by the TOE. Examples_include reviewing the TOE audit log for a dropped packet audit or observing the TOE terminates the connection.

(TD0732 applied)

Test 1: The evaluator established an ssh session with the TOE, then created and sent a packet to the TOE that was exactly the maximum packet size of 262127 bytes verifying the TOE accepted the packet by confirming the connection was not terminated when transmitted by the ssh client.

Test 2: The evaluator established an ssh session with the TOE, then created and sent a packet to the TOE that exceeded the maximum packet size by 1 (262128). The evaluator then confirmed the TOE dropped the packet by terminating the connection with the ssh client.

2.2.17.4 SSH10:FCS_SSH_EXT.1.4

TSS Assurance Activities: The evaluator will check the description of the implementation of SSH in the TSS to ensure the encryption algorithms supported are specified. The evaluator will check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Section 6.2 of the [ST] asserts the TOE supports AES-CBC and AES-CTR (both 128 and 256 keyed variants) ciphers for data encryption and hmac-sha2-256/sha2-512 for data integrity (and does not allow the “none” MAC algorithm).

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Section "Disabling unsupported algorithms" in the [AGD] gives the administrator commands for restricting the SSH interface to only use certified algorithms.

```
switch(config)# ssh ciphers aes128-ctr, aes256-ctr, aes128-cbc, aes256-cbc
```

Individual algorithms are ordered and advertised to the peer SSH device as configured.

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

If the TOE can be both a client and a server, these tests must be performed for both roles.

Test 1: The evaluator must ensure that only claimed algorithms and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall establish an SSH connection with a remote endpoint. 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 only the algorithms defined in the ST for the TOE for SSH connections. The evaluator shall perform one successful negotiation of an SSH connection and verify that the negotiated algorithms were included in the advertised set. If the evaluator detects that not all algorithms defined in the ST for SSH are advertised by the TOE or the TOE advertises additional algorithms not defined in the ST for SSH, the test shall be regarded as failed.

The data collected from the connection above shall be used for verification of the advertised hashing and shared secret establishment algorithms in FCS_SSH_EXT.1.5 and FCS_SSH_EXT.1.6 respectively.

Test 2: For the connection established in Test 1, the evaluator shall terminate the connection and observe that the TOE terminates the connection.

Test 3: The evaluator shall configure the remote endpoint to only allow a mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the attempt fails.

Test 1: The evaluator attempted to establish an SSH connection with one of the claimed SSH algorithms in the SFR to encrypt the session. The evaluator then observed the advertised algorithms in the packet capture from the session and observed that the TOE supports the following:

aes128-ctr

aes256-ctr

aes128-cbc

aes256-cbc

Test 2: After establishing a connection with each SSH algorithm. The evaluator entered the command to terminate the connection, and the TOE terminated the connection.

Test 3: After configuring the SSH remote endpoint to use a cipher prohibited by the ST, the evaluator attempted to connect to the TOE. The subsequent connection request was rejected.

2.2.17.5 SSH10:FCS_SSH_EXT.1.5

TSS Assurance Activities: The evaluator will check the description of the implementation of SSH in the TSS to ensure the hashing algorithms supported are specified. The evaluator will check the TSS to ensure that the hashing algorithms specified are identical to those listed for this component.

Section 6.2 of the [ST] asserts the TOE supports AES-CBC and AES-CTR (both 128 and 256 keyed variants) ciphers for data encryption and hmac-sha2-256/sha2-512 for data integrity (and does not allow the “none” MAC algorithm).

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Section "Disabling unsupported algorithms" in the [AGD] gives the administrator commands for restricting the SSH interface to only use certified algorithms.

```
switch(config)# ssh macs hmac-sha2-256, hmac-sha2-512
```

Individual algorithms are ordered and advertised to the peer SSH device as configured.

Testing Assurance Activities: Test 1: The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.

Test 2: [conditional] If aes256-gcm@openssh.com is not being negotiated, the evaluator shall configure an SSH peer to allow only a hashing algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected. (TD1023 applied)

Test 1: The evaluator referred to FCS_SSH_EXT.1.4 and verified that hmac-sha2-256, hmac-sha2-512, and implicit were offered by the TOE .

Test 2: The evaluator configured the SSH peer to use a hashing algorithm not included in the ST selection (hmac-md5). The subsequent connection attempt failed.

2.2.17.6 SSH10:FCS_SSH_EXT.1.6

TSS Assurance Activities: The evaluator will check the description of the implementation of SSH in the TSS to ensure the shared secret establishment algorithms supported are specified. The evaluator will check the TSS to ensure that the shared secret establishment algorithms specified are identical to those listed for this component.

Section 6.2 of the [ST] states that the TOE uses ecdh-sha2-nistp256/384 for SSHv2 key exchange.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Section "Disabling unsupported algorithms" in the [AGD] gives the administrator commands for restricting the SSH interface to only use certified algorithms.

```
switch(config)# ssh key-exchange-algorithms ecdh-sha2-nistp256, ecdh-sha2-nistp384
```

Individual algorithms are ordered and advertised to the peer SSH device as configured.

Testing Assurance Activities: Test 1: The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.

Note that it is permissible for the TOE to advertise the 'kex-strict-c-v00@openssh.com' or 'kex-strict-s-v00@openssh.com' value alongside the other key exchange mechanisms. These values are associated with client and server, respectively.

Test 2: The evaluator shall configure an SSH peer to allow only a key exchange method that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected.

Test 3: The evaluator shall configure the SSH peer to omit the kex-strict value from the kex_algorithms field. The evaluator shall verify that the TOE is able to successfully establish a connection with the SSH peer.

Test 4: The evaluator shall configure the SSH peer to include the kex-strict value in the kex_algorithms field. The evaluator shall verify that the TOE is able to successfully establish a connection with the SSH peer.

(TD0967 applied)

Test 1: The evaluator referred to test FCS_SSH_EXT.1.4 and verified the TOE advertised ecdh-sha2-nistp256 (RFC 5656) and ecdh-sha2-nistp384 (RFC 5656).

Test 2: The evaluator attempted to connect to the TOE with a SSH client using an unallowable key exchange algorithm (diffie-hellman-group1-sha1). The connection attempt was rejected by the TOE.

Test 3: The evaluator attempted to connect to the TOE with a SSH client that omits the kex-strict algorithm (kex-strict-c-v00@openssh.com) from its SSH client key exchange message. The TOE accepted the connection.

Test4: The evaluator attempted to connect to the TOE with a SSH client that includes the kex-strict algorithm (kex-strict-c-v00@openssh.com) in its SSH client key exchange message. The TOE accepted the connection.

2.2.17.7 SSH10:FCS_SSH_EXT.1.7

TSS Assurance Activities: The evaluator will check the description of the implementation of SSH in the TSS to ensure the KDFs supported are specified. The evaluator will check the TSS to ensure that the KDFs specified are identical to those listed for this component

Section 6.2 in [ST] states that the TOE ssh implementation is in strict compliance with RFC 4253.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.17.8 SSH10:FCS_SSH_EXT.1.8

TSS Assurance Activities: The evaluator shall check the TSS to ensure that if the TOE enforces connection rekey or termination limits lower than the maximum values that these lower limits are identified.

In cases where hardware limitation will prevent reaching data transfer threshold in less than one hour, the evaluator shall check the TSS to ensure it contains:

a. An argument describing this hardware-based limitation and

b. Identification of the hardware components that form the basis of such argument.

For example, if specific Ethernet Controller or Wi-Fi radio chip is the root cause of such limitation, these subsystems shall be identified.

Section 6.2 in [ST] states the TOE initiates a rekey before 1 hour has passes or before 1GB of data transfer occurs, whichever comes first.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that if the connection rekey or termination limits are configurable, it contains instructions to the administrator on how to configure the relevant connection rekey or termination limits for the TOE.

Section "SSH Rekey" in the [AGD] states that the SSH server will perform a rekey operation for all open SSH sessions at every hour or before 1 GB of data transferred, whichever occurs first. These limits are non-configurable.

Testing Assurance Activities: The test harness needs to be configured so that its connection rekey or termination limits are greater than the limits supported by the TOE -- it is expected that the test harness should not be initiating the connection rekey or termination.

Test 1: Establish an SSH connection. Wait until the identified connection rekey limit is met. Observed that a connection rekey or termination is initiated. This may require traffic to periodically be sent, or connection keep alive to be set, to ensure that the connection is not closed due to an idle timeout.

Test 2: Establish an SSH connection. Transmit data from the TOE until the identified connection rekey or termination limit is met. Observe that a connection rekey or termination is initiated.

Test 3: Establish an SSH connection. Send data to the TOE until the identified connection rekey limit or termination is met. Observe that a connection rekey or termination is initiated.

Test 1: The evaluator attempted to connect to the TOE using a SSH client waiting an hour to ensure that a rekey happened at the 1 hour threshold. At the 1 hour threshold, the connection was rekeyed.

Test 2: The evaluator attempted to connect to the TOE using a SSH client while downloading at least 1gb of files from the TOE. Before 1gb of data was transmitted, the connection was rekeyed.

Test 3: The evaluator attempted to connect to the TOE using a SSH client while issuing a TOE CLI command intended to print a large amount of text, thus sending text data to the test server. Before 1gb of data was transmitted, the connection was rekeyed.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.18 SSH Protocol - Server - per TD0682 (SSH10:FCS_SSHS_EXT.1)

2.2.18.1 SSH10:FCS_SSHS_EXT.1.1

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 check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Section 6.2 of [ST] states that the TOE implements the SSHv2 protocol, compliant to the following RFCs: 4251, 4252, 4253, 4254, 5656, 6668. The TOE supports public key-based and password-based authentication. The TOE allows use of the ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 algorithms for both host and user public key authentication.

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

Test 1: The evaluator shall use a suitable SSH Client to connect to the TOE and examine the list of server host key algorithms in the SSH_MSG_KEXINIT packet sent from the server to the client to determine that only the configured server authentication methods for the TOE were offered by the server.

Test 2: The evaluator shall test for a successful configuration setting of each server authentication method as follows. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established. Repeat this process for each independently configurable server authentication method supported by the server.

Test 3: The evaluator shall configure the peer to only allow an authentication mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the TOE sends a disconnect message.

(TD0682 applied)

Test 1: The evaluator referred to test FCS_SSH_EXT.1.4-t1 where the evaluator verified that the TOE only advertised ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, and no other algorithms.

Test 2: The evaluator configured the test machine to attempt a connection using only a single advertised algorithm and confirmed a successful connection was initiated. This was then repeated for each advertised algorithm in test 1.

Test 3: The evaluator configured the test machine to only use ssh-dss, an algorithm not supported by the TOE. The evaluator then attempted to connect to the TOE, the connection was rejected.

2.2.19 TLS Client Protocol - per TD0899 (NDcPP30e:FCS_TLSC_EXT.1)

2.2.19.1 NDcPP30e:FCS_TLSC_EXT.1.1

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

Section 6.2 in [ST] lists the supported ciphersuites in the TOE's TLS client implementation:

- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256

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

Section 6.2 of [ST] gives the complete list of cipher suites permitted for use on the TOE when it acts as a TLS client in the evaluated configuration. These ciphersuites are non-configurable, and no other configuration is necessary for their use.

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

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

The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.

b. Test 2: The evaluator shall establish the connection with a server presenting a certificate that contains the serverAuth (OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage extension and verify that the connection successfully negotiated. The evaluator shall then verify that when the same server presents an otherwise valid server certificate that contains the extendedKeyUsage extension without serverAuth the client rejects the connection. Ideally, the two certificates should be identical except for the OID values.

c. Test 3 [conditional]: Perform this test only if support of TLS 1.2 is claimed. The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

d. Test 4: The evaluator shall perform the following 'negative tests':

i. [conditional]: Perform this test only if support of TLS 1.2 is claimed. The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the TOE TLS client denies the connection.

ii. Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite (compatible with the server-selected version of TLS) not presented in the Client Hello handshake message. The evaluator shall verify that the TOE TLS client rejects the connection after receiving the Server Hello.

iii. The evaluator shall attempt to establish a TLS connection using each valid TLS/SSL version (i.e. TLS 1.3, TLS 1.2, TLS 1.1, TLS 1.0, SSL 3.0, SSL 2.0). The evaluator shall verify that the version(s) specified in FCS_TLSC_EXT.1.1 are successfully established and all other versions are rejected by the TOE TLS client. If a supported_versions extension is not sent by the TOE in the ClientHello, then the evaluator must ensure the test server responds with a ServerHello that is valid for the TLS version being negotiated. If the TOE includes the Supported Versions extension in its ClientHello, the evaluator shall also ensure the version(s) specified in the extension match the version(s) in FCS_TLSC_EXT.1.1. NOTE: For TLS 1.3 aware test servers, it is appropriate for the test server to issue a TLS Alert. The TOE client must not attempt to continue the connection.

e. Test 5: The evaluator shall perform the following modifications to the traffic (i.e. Man-in-the-middle modifications that result in invalid signatures and MACs):

i. [conditional]: Perform this test only if support of TLS 1.2 is claimed. If using DHE or ECDH ciphersuites, modify the signature block in the Server's Key Exchange handshake message, and verify that the client denies the connection and no application data flows. The handshake shall be valid (e.g. the Finished message is calculated using the modified signature), with the exception of the invalid signature. This test does not apply to ciphersuites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

ii. [conditional]: Perform this test only if support of TLS 1.3 is claimed. Modify the signature block in the Server's Certificate Verify handshake message, and verify that the client denies the connection and no application data flows. The handshake shall be valid (e.g. the Finished message is calculated using the modified signature), with the exception of the invalid signature.

f. Test 6: The evaluator shall perform the following 'scrambled message tests':

i. Modify a byte in the Server Finished handshake message and verify that the handshake does not finish successfully and no application data flows. (Note: This modification must be performed prior to the contents of the Finished message being encrypted.)

ii. [conditional]: Perform this test only if support of TLS 1.2 is claimed. Send a garbled message from the server after the server has issued the ChangeCipherSpec message and verify that the handshake is not finished successfully and no application data flows. (Note: TLS 1.3 provides for a dummy ChangeCipherSpec message to aid in middlebox compatibility if such an option is enabled in the specific implementation [see Section D.4 in RFC 8446]. If TLS 1.3 middlebox compatibility mode is enabled a ChangeCipherSpec message may appear in packet traces, but it does not influence the protocol. To be clear: for TLS 1.3, this test does not need to be performed.)

iii. [conditional]: Perform this test only if support of TLS 1.3 is claimed. Send a plaintext EncryptedExtensions message from the server and verify that the handshake is not finished successfully and no application data flows.

(Note: Under TLS 1.3, the EncryptedExtensions message is the first message to be encrypted with the handshake traffic secret.)

iv. [conditional]: Perform this test only if support of TLS 1.2 is claimed. Modify at least one byte in the server's nonce in the Server Hello handshake message and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

Test 1: The evaluator established a TLS session from the TOE to a test server with the test server configured to accept connections with only one of the claimed cipher suites. The evaluator used a network sniffer to capture the TLS session negotiation. The evaluator examined each traffic capture and observed that the expected TLS cipher was negotiated.

Test 2: The evaluator configured the test server to send a certificate with the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is accepted by the TOE. The evaluator reconfigured the test server to retry the TLS session using a cert that is missing the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is rejected by the TOE.

Test 3: The evaluator established a TLS session from the TOE. A modified test server negotiates TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, but returns an RSA Certificate. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is not negotiated successfully.

Test 4i: The evaluator configured a test server to offer only the TLS_NULL_WITH_NUL_NULL ciphersuite. The evaluator then attempted to establish a TLS session from the TOE to that test server. Using a network sniffer, the evaluator captured the TLS session negotiation and observed that the TLS session is rejected by the TOE.

Test 4ii: The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to choose a ciphersuite that the TOE did not offer in its Client Hello handshake message. Using a network sniffer, the evaluator captured the TLS session negotiation and observed that the TLS session is rejected by the TOE.

Test 4iii: For each version of TLS/SSL, the evaluator attempted to establish a TLS connection with the TOE. The evaluator confirmed that only versions claimed in the ST accepted a connection while all other versions were rejected.

Test 5i: The evaluator configured the TOE to connect to a GSS test server using TLS. During the connection the evaluator caused the server to modify the signature block in the Server's Key Exchange handshake message. The connection attempt after modifying the signature block was rejected.

Test 5ii: The evaluator configured the TOE to connect to a GSS test server using TLS. During the connection the evaluator caused the server to modify the signature block in the Server's Certificate Verify handshake message. The evaluator modified the Client’s random value that is used by the server to calculate the signature. The

evaluator confirmed no other modifications beyond the previously tested control case, and the TOE exhibited the expected behavior of rejecting the connection.

Test 6i and 6ii: The evaluator obtained a packet capture of the TLS session negotiation between the TOE (client) and a test server. The server implementation of the TLS protocol was modified as stated in the assurance activity 'i' and 'ii'. The evaluator verified that the client did not finish the negotiation and no application data was transferred.

Test 6iii: The evaluator forced the TLS Server to send its Encrypted Extensions message after the initial TLS handshake in plaintext (rather than encrypted with the secret derived during the handshake). The plaintext Encrypted Extensions message is visible as the first Application Data packet sent by the TLS Server. Although the message is decoded as Application Data, the encrypted extensions message with bytes "...:08:00:00:02:00:00:..." is visible in plaintext within that Application Data packet, and is not actually encrypted.

2.2.19.2 NDcPP30e:FCS_TLSC_EXT.1.2

TSS Assurance Activities: The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported.

Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the administrator are relaxed and the identifier may also be established through a 'Gatekeeper' discovery process. The TSS shall describe the discovery process and highlight how the reference identifier is supplied to the 'joining' component. Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1 and the ST author selected attributes from RFC 5280, the evaluator shall ensure the TSS describes which attribute type, or combination of attributes types, are used by the client to match the presented identifier with the configured identifier. The evaluator shall ensure the TSS presents an argument how the attribute type, or combination of attribute types, uniquely identify the remote TOE component; and the evaluator shall verify the attribute type, or combination of attribute types, is sufficient to support unique identification of the maximum supported number of TOE components.

If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE's conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC5952 for IPv6, RFC 3986 for IPv4) is enforced.

Section 6.2 of [ST] states the TOE supports the use of FQDN and IPv4 addresses as reference identifiers within a certificate’s CommonName (CN) or Subject Alternate Name (SAN) extension. The TOE checks the SAN/CN when performing certificate validation as described in NDcPP30e:FIA_X509_EXT.1/Rev. Wildcards are allowed in

certificates. IP addresses are converted to binary by parsing decimal delimited by periods. The conversion happens before any comparisons are made. Canonical format is enforced.

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 includes support for IP addresses, 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.

Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1, the SFR selects attributes from RFC 5280, and FCO_CPC_EXT.1.2 selects 'no channel'; the evaluator shall verify the guidance provides instructions for establishing unique reference identifiers based on RFC5280 attributes.

The section titled "Certificate Installation and Validation" in [AGD] discusses X.509 digital certificates that are used for remote logging protection. This section explains how certificates are validated.

The section entitled "Secure Remote Logging" further states that the syslog client shall compare the syslog server's FQDN or IPv4 address against the syslog server's certificate Common Name or Subject Alternative Name. This indicates that the SAN field is not required. This section provides the command for enabling certificate name checking and provides a note that says "While IP address is supported for identity verification, it is recommended that FQDN is used for higher assurance."

The TOE is not distributed.

Testing Assurance Activities: Note that the following tests are marked conditional and are applicable under the following conditions:

a. For TLS-based trusted channel communications according to FTP_ITC.1 where RFC 6125 is selected, tests 1-6 are applicable.

or

b. For TLS-based trusted path communications according to FTP_TRP where RFC 6125 is selected, tests 1-6 are applicable

or

c. For TLS-based trusted path communications according to FPT_ITT.1 where RFC 6125 is selected, tests 1-6 are applicable. Where RFC 5280 is selected, only test 7 is applicable.

Note that for some tests additional conditions apply.

IP addresses are binary values that must be converted to a textual representation when presented in the CN of a certificate. When testing IP addresses in the CN, the evaluator shall follow the following formatting rules:

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.

- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

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

a. Test 1 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the CN.

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

b. Test 2 [conditional]: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, URI). When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the SAN.

c. Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

d. Test 4 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, SRV).

e. Test 5 [conditional]: The evaluator shall perform the following wildcard tests with each supported type of reference identifier that includes a DNS name (i.e. CN-ID with DNS, DNS-ID, SRV-ID, URI-ID):

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

ii. [conditional]: The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds if wildcards are supported or fails if wildcards are not supported. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

f. Test 6: Objective: The objective of this test is to ensure the TOE is able to differentiate between IP address identifiers that are not allowed to contain wildcards and other types of identifiers that may contain wildcards.

[conditional]: If IP addresses identifiers are supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.0.1, CN=2001:0DB8:0000:0000:0008:0800:200C:*when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

Test 7 [conditional]: If the secure channel is used for FPT_ITT, and RFC 5280 is selected, the evaluator shall perform the following tests. Note, when multiple attribute types are selected in the SFR (e.g. when multiple attribute types are combined to form the unique identifier), the evaluator shall modify each attribute type in accordance with the matching criteria described in the TSS (e.g. creating a mismatch of one attribute type at a time while other attribute types contain values that will match a portion of the reference identifier):

i. The evaluator shall present a server certificate that does not contain an identifier in the Subject (DN) attribute type(s) that matches the reference identifier. The evaluator shall verify that the connection fails.

ii. The evaluator shall present a server certificate that contains a valid identifier as an attribute type other than the expected attribute type (e.g. if the TOE is configured to expect id-atserialNumber=correct_identifier, the certificate could instead include id-at-name=correct_identifier), and does not contain the SAN extension. The evaluator shall verify that the connection fails. Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass this test.

iii. The evaluator shall present a server certificate that contains a Subject attribute type that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.

iv. The evaluator shall confirm that all use of wildcards results in connection failure regardless of whether the wildcards are used in the left or right side of the presented identifier. (Remark: Use of wildcards is not addressed within RFC 5280.)

Test 1: The evaluator configured the TOE to expect a CN-ID or DN-ID. The evaluator then established a TLS session from the TOE targeting a server using a valid certificate with a CN matching the domain name used by the client. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed a successful connection. The evaluator then established a TLS session from the TOE targeting a server using a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 2: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 3: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 4: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 5: The evaluator tested hostname wildcards by configuring an expected DNS on the TOE with the test server's certificates configured with wildcard DNS names. The TOE successfully checked the hostname wildcards and behaved as expected. The evaluator used a network sniffer to capture the TLS session negotiation and observed that the TLS session was negotiated as shown in column 3 of the following table.

Certificate Contents	Host ID	Expected Result
CN=bar.*.example.com	bar.foo.example.com	No Connection
SAN=bar.*.example.com	bar.foo.example.com	No Connection
CN=*.example.com	foo.example.com	Successful Connection
SAN=*.example.com	foo.example.com	Successful Connection

Test 6: The evaluator configured the TOE to connect with the GSS test server using TLS with the test server alternately configured with a certificate identifier as indicated in each test case below. The evaluator observed that the TOE connected when the identifier fulfilled the required rules, and the connection was rejected when the rules were not followed.

Test 7: The TOE does not utilize TLS for FTP_ITT communication and therefore this test is not applicable.

2.2.19.3 NDcPP30e:FCS_TLSC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

b. Test 2 [conditional]: If 'except with the following administrator override' is selected, the evaluator shall change the presented certificate(s) or modify the operational environment, so that certificate validation fails due to the TSF's inability to determine revocation status. The evaluator shall verify that the certificate is not accepted by the TSF until the Security Administrator authorizes the TSF to establish the connection and this action results in the Trusted Channel being successfully established.

c. Test 3: While performing testing of invalid TLS Client Reference Identifiers, expired X.509 certificates, and invalid X.509 trust chains; the evaluator shall ensure the TSF does not present an administrator override option, with the exception of failure to determine revocation status (if selected). Note: This should be a review of behavior observed while performing other tests.

Test 1: This test has been performed in NDcPP30e:FIA_X509_EXT.1 Test 1.

Test 2: Not applicable as the TOE does not provide an administrator override.

Test 3: The TOE does not offer an administrator override, including for failure to determine revocation status.

2.2.19.4 NDcPP30e:FCS_TLSC_EXT.1.4

TSS Assurance Activities: If 'present the Supported Groups Extension' is selected, the evaluator shall verify that TSS describes the Supported Groups Extension and whether the required behaviour is performed by default or may be configured. If TLS 1.2 is claimed and DHE ciphers are claimed, then the TSS must also specify whether the TOE is capable of negotiating DHE ciphers and whether the TOE client will terminate if an unsupported DHE parameter set is returned in the Server Key Exchange or whether all valid server-generated DHE parameters are accepted.

Section 6.2 of [ST] states that Elliptical curves P-256, P-384, and P-521 are supported. These are not configurable.

DHE ciphers are not claimed in the evaluated configuration.

Guidance Assurance Activities: If the TSS indicates that the Supported Groups Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Groups Extension.

No configuration is needed to meet the Supported Elliptic Curves/Supported Groups Extension claims.

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

a. Test 1 [conditional]: If 'not present the Supported Groups Extension' is selected, the evaluator shall examine the Client Hello message and verify it does not contain the Supported Groups extension.

b. Test 2 [conditional]: If 'present the Supported Groups Extension' is selected, the evaluator shall configure the server to perform ECDHE or DHE (as applicable) key exchange using each of the TOE's supported groups. The evaluator shall verify that the connection succeeds. This test shall be repeated for each type of key exchange message/extension supported (i.e. Key Share extension for TLS 1.3 and Server Key Exchange Message for TLS 1.2).

c. Test 3 [conditional]: If secp curves are selected, the evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve and shall verify that the connection fails and no application data flows. The non-supported curve shall be as similar to the selected curve(s) as possible (i.e. a non-selected curve when not all curves are selected or P-224). This test shall be repeated for each type of key exchange message/extension supported (i.e. Key Share extension for TLS 1.3 and Server Key Exchange Message for TLS 1.2).

d. Test 4a [conditional, for TLS 1.3 only]: If ffdhe curves are selected, the evaluator shall configure the server to perform a DHE key exchange in the TLS connection using a non-supported group and shall verify that the connection fails and no application data flows. The non-supported group shall be as similar to the selected group(s) as possible (i.e. a non-selected group when not all groups are selected or undefined Codepoint 0x0105 (ffdhe8192 + 1)).

e. Test 4b [conditional, for TLS 1.2 only]: If ffdhe curves are selected, the evaluator shall configure the server to return DHE parameters in the Server Key Exchange in the TLS connection that do not meet the construction for any claimed ffdhe group. The evaluator shall verify that the connection fails and no application data flows. If the TOE client supports any server-returned DHE parameter set, then this test is not applicable.

Test 1: Not applicable, 'present the Supported Groups Extension' is selected.

Test 2: The evaluator for each supported group in the ST, configured the GSS test server to support a key exchange and with the supported group, established a connection between the server and TOE.

Test 3: The evaluator configured the GSS test server to perform a key exchange using an unsupported ECDHE curve (P-224). Upon attempting a connection with the TOE, the connection was rejected.

Test 4a: The evaluator configured the TOE to connect to a GSS test server using TLS with a TOE supported FFDHE key exchange method. The evaluator also configured the GSS test server to accept that same FFDHE key exchange

method, but to require a curve that was not supported by the TOE (i.e., ffdhe8192 + 1). The evaluator then caused the TOE to attempt the connection, and observed a rejection.

Test 4b: Not applicable, ffdhe curves are not selected.

2.2.19.5 NDcPP30e:FCS_TLSC_EXT.1.5

TSS Assurance Activities: [Conditional]: The evaluator shall verify that TSS describes the signature_algorithms extension and whether the required behavior is performed by default or may be configured.

[Conditional]: The evaluator shall verify that TSS describes the signature_algorithms_cert extension and whether the required behavior is performed by default or may be configured.

Section 6.2 of [ST] states that the signature_algorithms extension presented supports the rsa_pkcs1 with sha256(0x0401), rsa_pkcs1with sha384(0x0501), rsa_pkcs1 with sha512(0x0601), ecdsa_secp256r1 with sha256(0x0403), ecdsa_secp384r1 with sha384(0x0503), ecdsa_secp521r1 with sha512(0x0603), rsa_pss_rsae with sha256(0x0804), rsa_pss_rsae with sha384(0x0805), rsa_pss_rsae with sha512(0x0806), rsa_pss_pss with sha256(0x0809), rsa_pss_pss with sha384(0x080a), rsa_pss_pss with sha512(0x080b) algorithms. These algorithms are not configurable.

Guidance Assurance Activities: If the TSS indicates that the signature_algorithms extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithms extension.

The signature_algorithms extension is not configurable on the TOE; this requirement is not applicable.

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

a. Test 1 [conditional]: The evaluator shall perform the following tests if 'present the signature_algorithms extension' is selected:

i. The evaluator shall examine the Client Hello message and verify it contains the signature_algorithms extension and the SignatureSchemes match the SignatureSchemes specified in the requirement.

ii. The evaluator shall establish a TLS connection using each of the SignatureSchemes specified by the requirement and observes the session is successfully completed. The evaluator shall ensure the test server sends a leaf Certificate that has a public key algorithm that is consistent with the SignatureScheme being tested. For TLS 1.2 and if the ciphersuite is DHE or ECDHE, the evaluator shall ensure that the server sends Server Key Exchange messages consistent with the SignatureScheme being tested. For TLS 1.3, the evaluator shall ensure that the server sends Certificate Verify messages consistent with the SignatureScheme being tested.

b. Test 2 [conditional]: The evaluator shall perform the following tests if 'present the signature_algorithms_cert extension' is selected:

i. The evaluator shall examine the Client Hello message and verify it contains the signature_algorithms_cert extension and the SignatureSchemes match the SignatureSchemes specified in the requirement.

ii. The evaluator shall establish a TLS connection using a certificate chain using each of the SignatureSchemes specified by the requirement. The evaluator shall ensure the signatures used in the certificate chain are consistent with the SignatureScheme being tested.

Test 1i: The evaluator configured the TOE to connect to the test server using TLS in a normal scenario. The evaluator used a packet sniffer to capture and view the Client Hello message verifying that the signature_algorithms extension was present and contained the SignatureSchemes 0x0401, 0x0501, and 0x0601 as stated in the requirement by the ST.

Test 1ii: The evaluator configured the test server to attempt a connection using each SignatureScheme in the requirement, the connection was accepted by the TOE.

Test 2: Not applicable as 'present the signature_algorithms_cert extension' is not selected.

2.2.19.6 NDcPP30e:FCS_TLSC_EXT.1.6

TSS Assurance Activities: The evaluator shall verify that TSS describes whether the list of supported ciphersuites can be configured or not.

Section 6.2 of [ST] states that the supported ciphersuites cannot be configured when in the evaluated configuration.

Guidance Assurance Activities: If the TSF provides the ability of configuring the list of supported ciphersuites, the evaluator shall verify that AGD guidance includes configuration of the list of supported ciphersuites.

The TOE does not provide the ability to configure the ciphersuites when in the evaluated configuration.

Testing Assurance Activities: [conditional]: If the TSF provides the ability of configuring the list of supported ciphersuites, the evaluator shall establish a TLS connection using one of the possible configurations of the list of supported ciphersuites. The evaluator shall then change the configuration and repeat the test. The evaluator shall verify that the behavior of the TOE has changed according to the modification of the list of ciphers. This test shall be repeated for all supported TLS versions. If the TSF does not provide the ability of configuring the list of supported ciphersuites, this test shall be omitted.

Not Applicable as the TSF does not provide the ability to configure the list of supported ciphersuites.

2.2.19.7 NDcPP30e:FCS_TLSC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall establish a TLS connection with a server and observe that the early data extension and the post-handshake client authentication extension according to RFC 8446 Section 4.2 are not advertised in the Client Hello Message. This test shall be executed for all TLS versions supported by the TOE.

The evaluator configured the TOE to connect with the GSS test server using TLS. The evaluator ensured the GSS test server and TOE were able to connect with compatible connection parameters (i.e. a Control Test). After the connection attempt, the evaluator examined the packet captures to determine that the Early Data extension, and the Post Handshake Auth extension were not offered.

2.2.19.8 NDcPP30e:FCS_TLSC_EXT.1.8

TSS Assurance Activities: The evaluator shall verify in the TSS that, for TLS 1.3, the TOE shall not permit out-of-band provisioning of pre-shared keys (PSKs) in the evaluated configuration.

Section 6.2 of [ST] states that for TLS 1.3, the TOE shall not permit out-of-band provisioning of pre-shared keys (PSKs) in the evaluated configuration.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

No TOE configuration is necessary to meet the requirement in this SFR.

Testing Assurance Activities: None Defined

2.2.19.9 NDcPP30e:FCS_TLSC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

a. Test 1 [conditional]: If 'support TLS 1.2 secure renegotiationâ€¦â€' is selected, the evaluator shall use a network packet analyzer/sniffer to capture a TLS 1.2 handshake between the two TLS endpoints. The evaluator shall verify that either the 'renegotiation_info' field or the SCSV ciphersuite is included in the ClientHello message during the initial handshake.

b. Test 2 [conditional]: If 'support TLS 1.2 secure renegotiationâ€¦â€' is selected, the evaluator shall perform a TLS 1.2 handshake and verify the TOE TLS Client's handling of ServerHello messages received during the initial handshake that include the 'renegotiation_info' extension. The evaluator shall modify the length portion of this

field in the ServerHello message to be non-zero and verify that the TOE TLS client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

c. Test 3 [conditional]: If 'support TLS 1.2 secure renegotiationâ€¦â€' is selected, the evaluator shall perform a TLS 1.2 handshake and verify that ServerHello messages received during secure renegotiation contain the 'renegotiation_info' extension. The evaluator shall modify either the 'client_verify_data' or 'server_verify_data' value and verify that the TOE TLS client terminates the connection.

d. Test 4a [conditional, if the TOE supports TLS 1.3]: The evaluator shall initiate a TLS session between the TSF and a test TLS 1.3 server that completes a compliant TLS 1.3 handshake, followed by a hello request message. The evaluator shall observe that the TSF completes the initial TLS 1.3 handshake successfully, but terminates the session on receiving the hello request message.

It is preferred that the TSF sends a fatal error alert message (e.g., unexpected message) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).

Test 4b [conditional, if the TOE supports TLS 1.2 and rejects TLS 1.2 renegotiation attempts]: The evaluator shall initiate a TLS session between the so-configured TSF and a test TLS 1.2 server that is configured to perform a compliant handshake, followed by a hello request. The evaluator shall confirm that the TSF completes the initial handshake successfully but does not initiate renegotiation after receiving the hello request.

(TD0899 applied)

Test 1: Not Applicable, "support TLS 1.2 secure renegotiation" is not selected.

Test 2: Not Applicable, "support TLS 1.2 secure renegotiation" is not selected.

Test 3: Not Applicable, "support TLS 1.2 secure renegotiation" is not selected.

Test 4a: The evaluator connected to the TOE using TLSv1.3, then caused the server to attempt a connection with the TOE using renegotiation. The TOE rejected the renegotiation attempt.

Test 4b: The evaluator connected to the TOE using TLSv1.2, then caused the server to attempt a connection with the TOE using renegotiation. The TOE rejected the renegotiation attempt.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For all tests in this chapter the TLS server used for testing of the TOE shall be configured not to require mutual authentication.

Section 6.2 of the ST states that the TOE supports TLS v1.2 and v1.3 with the ciphersuites for its syslog connections the following ciphersuites are supported for communications with syslog servers:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256

Section 6.2 of the ST also states that the TOE supports TLS v1.2 with the ciphersuites listed above for its syslog connections.

The list of ciphersuites in the TSS is consistent with those listed in the requirement.

2.2.20 TLS Client Support for Mutual Authentication (NDcPP30e:FCS_TLSC_EXT.2)

2.2.20.1 NDcPP30e:FCS_TLSC_EXT.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 TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Section 6.2 of [ST] describes the use of client-side certificates on the TOE; the TOE can be configured with an X509 certificate which it will send to a TLS server in response to a certificate request message sent by the TLS server.

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

Section entitled "Certificate Installation and Validation" in [AGD] explains how to generate a CSR and import the X.509 digital certificate that is used for remote logging protection.

Component Testing Assurance Activities: For all tests in this section the TLS server used for testing of the TOE shall be configured to require mutual authentication.

Test 1: The evaluator shall establish a connection to a peer server that is configured for mutual authentication (i.e. sends a server Certificate Request (type 13) message). The evaluator shall observe that the TOE TLS client sends both client Certificate (type 11) and client Certificate Verify (type 15) messages during its negotiation of a TLS channel and that Application Data is sent.

In addition, all other testing in FCS_TLSC_EXT.1 and FIA_X509_EXT.* must be performed as per the requirements.

The evaluator performed all TLS client tests such that the test server used for testing of FCS_TLSC_EXT.1.1 Test 1 required mutual authentication. In all of the following test cases the TOE responded in accordance with the AA and Security Target claims.

2.2.21 TLS Server Protocol - per TD0899 & TD0973 (NDcPP30e:FCS_TLSS_EXT.1)

2.2.21.1 NDcPP30e:FCS_TLSS_EXT.1.1

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

The evaluator shall verify that the TSS contains a description of how the TOE technically prevents the use of unsupported and undefined SSL and TLS versions.

Section 6.2 of [ST] states that the TOE supports the following ciphersuites:

- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, and
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256

The evaluator confirmed that the set of ciphersuites selected in FCS_TLSS_EXT.1.1 and the set identified in the TSS was identical.

The TOE acts as a TLS server supporting TLSv1.2 and TLSv1.3 only. No older versions of TLS, and no version of SSL are supported.

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

Section "Web Interface" in [AGD] contains a subsection entitled "Cipher Suites" which lists the cipher suites are permitted in the evaluated configuration. The 10 cipher suites listed here for the Aruba CX switch acting as a TLS server match those claimed by the FCS_TLSS_EXT.1.1 selection in the [ST]. No configuration is needed to support these cipher suites.

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

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

b. Test 2: The evaluator shall perform the following tests:

i. The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection.

ii. [conditional]: Perform this test only if support of TLS1.2 is claimed. The evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

c. Test 3: The evaluator shall perform the following modifications to the traffic:

i. [conditional]: Perform this test only if support of TLS1.2 is claimed. Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

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

[conditional]: Perform this test only if support of TLS1.2 is claimed.The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

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

iii.[conditional]: Perform this test only if support of TLS 1.3 is claimed. The evaluator shall use a client to send a Client Hello message containing a single curve in the Supported Groups extension. The curve that is selected to be presented in this extension should not be supported by the TOE. The evaluator shall verify that the TOE disconnects after receiving the Client Hello message.

iv. [conditional]: Perform this test only if support of TLS 1.3 is claimed. The evaluator shall use a client to send a Client Hello message containing multiple curves in the Supported Groups extension. These curves should be chosen such that only one of these curves is supported by the TOE. The evaluator shall verify that the TOE responds with a Hello Retry Request message selecting the supported curve. This shall be reflected in the Key Share extension of the Hello Retry Request message.

d. Test 4: The evaluator shall attempt to establish a TLS/SSL connection using each of the supported TLS/SSL versions (i.e., TLS 1.3, TLS 1.2, TLS 1.1, TLS 1.0, SSL 3.0, SSL 2.0). The client shall be configured so it only supports the version being tested. The evaluator shall verify that the versions specified in FCS_TLSS_EXT.1.1 are successfully established and all other versions not successfully established. If the TOE attempts to downgrade the version, it is acceptable for the test client to terminate the connection; however, the version selected by the TOE shall always be a version specified in FCS_TLSS_EXT.1.1.

Test 1: The evaluator attempted to connect to the TOE using each of the ciphersuites claimed. A packet capture was obtained for each connection attempt. The following are the cipher suites for which successful connections could be established:

RSA_WITH_AES_256_GCM_SHA384

RSA_WITH_AES_128_GCM_SHA256

ECDHE-RSA-AES256-GCM-SHA384

ECDHE-ECDSA-AES256-GCM-SHA384

ECDHE_RSA_WITH_AES_128_GCM_SHA256

ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_AES_128_GCM_SHA256

TLS_AES_256_GCM_SHA384

TLS_AES_128_CCM_SHA256

TLS_AES_128_CCM_8_SHA256

Test 2:

i. The evaluator attempted to connect to the TOE using all ciphers except those defined in the PP and observed the connection was rejected.

ii. The evaluator attempted to connect to the TOE with a client that sents a Client Hello to the TOE containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and observed the connection was rejected.

Test 3:

i. The evaluator attmpeted to establish a TLS session with the TOE when the evaluator's client modified a byte in the Client Finished handshake message. The evaluator observed the connection was rejected.

ii. The evaluator established a TLS session with the TOE and examined the captured traffic. The evaluator located the ChangeCipherSpec message (Content type hexadecimal 14) in the packet capture and found it to be followed by an EncryptedHandshakeMessage (Content type hexadecimal 16). The evaluator examined the payload of the EncryptedHandshakeMessage to determine if it contained a cleartext or encrypted version of the required Finished message. A Cleartext version would begin with a Content type hexidecimal value of 14, while an encrypted version would (for at least one of three test messages) not contain a Content type hexidecimal value of 14.

iii. The evaluator attempted to establish a TLS session with the TOE when the evaluator's client specified only one key exchange method in the Client Hello (X25519). The evaluator observed the connection being dropped after the Client Hello was recieved.

iv. The evaluator attempted to establish a TLS session with the TOE when the evaluator's client specified claimed key exchange methods in the Client Hello (P-256, P-384, P-521). The evaluator observed successful connection attempts using all three key exchange methods.

Test 4: The evaluator alternately attempted to connect to the TOE using a control case (TLS not specified), SSL2.0, SSL3.0, TLSv1.0, TLSv1.1, TLSv1.2 and TLSv1.3. Only the control case, TLS1.2, and TLS1.3 connections were successful.

2.2.21.2 NDcPP30e:FCS_TLSS_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS describes the algorithms and key sizes the TSF supports for authenticating itself to TLS clients. The evaluator shall ensure these algorithms are consistent with the selected ciphersuites.

Section 6.2 of [ST] states that ECDSA key exchanges using secp256r1, secp384r1, and secp521r1 are supported, in addition to RSA key sizes 2048, 3072, and 4096.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Section 6.2 of [ST] states that no configuration is necessary to meet the claimed algorithms.

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

a. Test 1 [conditional]: If ECDHE ciphersuites/groups are supported:

i. The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite (TLS 1.2) or group (TLS 1.3) and a single supported elliptic curve specified in the supported groups extension. The Evaluator shall verify (through a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange (TLS 1.2) or Server Hello (key_share, for TLS 1.3) message and successfully establishes the connection.

For TLS 1.2, the evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single unsupported elliptic curve (e.g., secp192r1 (0x13)) specified in RFC 4492, Section 5.1.1. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

For TLS 1.3, the evaluator shall attempt a connection using a supported ciphersuite and a single unsupported group. Both the key_share and supported_groups extensions must be set to the same unsupported group. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

b. Test 2a: [conditional] If the TOE supports TLS 1.2 and DHE ciphersuites, the evaluator shall repeat the following test for each supported parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use each supported Diffie-Hellman parameter size.

The evaluator shall attempt a connection using a supported DHE ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a Server Key Exchange Message where p Length is consistent with the ones configured Diffie-Hellman parameter size(s).

Test 2b: [conditional] If the TOE supports TLS 1.3 and 'ffdhe' parameters are selected, the evaluator shall repeat the following test for each supported FFDHE parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use each supported Finite Field Diffie-Hellman parameter size.

The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a Server Key Share Extension Message where the KeyShareServerHello structure contains a KeyShareEntry structure with an opaque key_exchange value whose Length is consistent with the configured Finite Field Diffie-Hellman parameter size(s). (TD0973 applied)

c. Test 3: [conditional] If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size..

Refer to FCS_TLSS_EXT.1.3 below as the tests overlap.

2.2.21.3 NDcPP30e:FCS_TLSS_EXT.1.3

TSS Assurance Activities: The evaluator shall verify that the TSS lists all EC Diffie-Hellman curves and/or Diffie-Hellman groups used in the key establishment by the TOE when acting as a TLS Server. The evaluator shall ensure these algorithms are consistent with the selected ciphersuites.

Section 6.2 of [ST] states that key exchanges using secp256r1, secp384r1, and secp521r1 are supported by the TOE.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Section 6.2 of [ST] states that no configuration is necessary to meet the claimed algorithms.

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

a. Test 1 [conditional]: If ECDHE ciphersuites/groups are supported:

i. The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite (TLS 1.2) or group (TLS 1.3) and a single supported elliptic curve specified in the supported groups extension. The Evaluator shall verify (through a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange (TLS 1.2) or Server Hello (key_share, for TLS 1.3) message and successfully establishes the connection.

For TLS 1.2, the evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single unsupported elliptic curve (e.g., secp192r1 (0x13)) specified in RFC 4492, Section 5.1.1. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

For TLS 1.3, the evaluator shall attempt a connection using a supported ciphersuite and a single unsupported group. Both the key_share and supported_groups extensions must be set to the same unsupported group. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

b. Test 2a: [conditional] If the TOE supports TLS 1.2 and DHE ciphersuites, the evaluator shall repeat the following test for each supported parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use each supported Diffie-Hellman parameter size.

The evaluator shall attempt a connection using a supported DHE ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a Server Key Exchange Message where p Length is consistent with the ones configured Diffie-Hellman parameter size(s).

Test 2b: [conditional] If the TOE supports TLS 1.3 and 'ffdhe' parameters are selected, the evaluator shall repeat the following test for each supported FFDHE parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use each supported Finite Field Diffie-Hellman parameter size.

The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a Server Key Share Extension Message where the KeyShareServerHello structure contains a KeyShareEntry structure with an opaque key_exchange value whose Length is consistent with the configured Finite Field Diffie-Hellman parameter size(s). (TD0973 applied)

c. Test 3: [conditional] If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size.

The evaluator performed these tests on all TLS server interfaces.

Test 1: The evaluator attempted to establish a TLS session with the TOE when the evaluator's TLS client specified only one key exchange method in the Client Hello. The evaluator observed that the TOE selected the same curve as offered by the client for the following key exchange methods.

- ECDHE w/ P-256 curve
- ECDHE w/ P-384 curve
- ECDHE w/ P-521 curve

The evaluator also attempted a connection using ECDHE w/ P-224 curve and observed the TOE rejected the connection attempt.

Test 2: Not applicable. The TOE does not support ciphersuites that use DHE key exchanges.

Test 3: For TLS 1.2, the evaluator attempted to establish a TLS session with the TOE when the evaluator's TLS client specified only one key exchange method in the Client Hello. The evaluator observed that the TOE selected the same key size as offered by the client for the following key exchange methods, RSA 2048, RSA 3072, and RSA 4096.

2.2.21.4 NDcPP30e:FCS_TLSS_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes if session resumption based on session IDs is supported (RFC 4346 and/or RFC 5246), if session resumption based on session tickets is supported (RFC 5077) and/or if session resumption according to RFC 8446 is supported.

If session tickets are supported, the evaluator shall verify that the TSS describes that the session tickets are encrypted using symmetric algorithms consistent with FCS_COP.1/DataEncryption. The evaluator shall verify that the TSS identifies the key lengths and algorithms used to protect session tickets.

If session tickets are supported, the evaluator shall verify that the TSS describes that session tickets adhere to the structural format provided in Section 4 of RFC 5077 and if not, a justification shall be given of the actual session ticket format.

If the TOE claims a TLS server capable of session resumption (as a single context, or across multiple contexts), the evaluator shall verify that the TSS describes how session resumption operates (i.e. what would trigger a full handshake, e.g. checking session status, checking Session ID, etc.). If multiple contexts are used, the TSS describes how session resumption is coordinated across those contexts. In case session establishment and session resumption are always using a separate context, the TSS shall describe how the contexts interact with respect to session resumption (in particular regarding the session ID). It is acceptable for sessions established in one context to be resumable in another context.

Section 6.2 of [ST] states that the TOE does supports session resumption using session IDs.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Section 6.2 of [ST] states that no configuration is necessary to meet the claimed behavior.

Testing Assurance Activities: Test Objective: To demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption).

The evaluator shall perform the following tests:

a. Test 1 [conditional]: If the TOE does not support session resumption based on session IDs according to RFC 5246 (TLS 1.2) or session tickets according to RFC 5077 (TLS 1.2) or session resumption according to RFC 8446 (TLS 1.3), the evaluator shall perform the following test:

i. For all supported TLS versions the client shall send a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket. A non-zero length session identifier for TLS 1.3 would result in testing compatibility mode which is not the objective of this test. For TLS 1.3, the evaluator shall ensure that a 'psk_key_exchange_modes' extension is included in the Client Hello.

ii. The client verifies the server does not send a NewSessionTicket handshake message (at any point in the handshake).

iii. The client verifies the Server Hello message contains a zero-length session identifier. For TLS 1.2 the client could alternatively pass the following steps (not applicable for TLS 1.3):

Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

iv. The client completes the TLS handshake and captures the SessionID from the ServerHello.

v. The client sends a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).

vi. The client verifies the TOE (1) implicitly rejects the SessionID by sending a ServerHello containing a different SessionID and by performing a full handshake (as shown in Figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Remark: If multiple contexts are supported for session resumption, the session ID or session ticket may be obtained in one context for resumption in another context. It is possible that one or more contexts may only permit the construction of sessions to be reused in other contexts but not actually permit resumption themselves. For contexts which do not permit resumption, the evaluator is required to verify this behaviour subject to the description provided in the TSS. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

b. Test 2 [conditional]: If the TOE supports session resumption using session IDs according to RFC 5246 (TLS 1.2), the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

i. The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246). When the session is resumed, the evaluator shall verify on the TLS Client used for performing this test, that the TOE (TLS Server) has not advertised support for the early data extension.

ii. The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and performing a full handshake (as shown in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ID may be obtained in one context for resumption in another context. There is no requirement that the session ID be obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ID constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

c. Test 3 [conditional]: If the TOE supports session tickets according to RFC 5077 (supported only by TLS 1.2), the evaluator shall carry out the following steps:

i. The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with an abbreviated handshake described in Section 3.1 of RFC 5077 and illustrated with an example in figure 2. Of particular note: if the server successfully verifies the client's ticket, then it may renew the ticket by including a NewSessionTicket handshake message after the ServerHello in the abbreviated handshake (which is shown in figure 2). This is not required, however as further clarified in Section 3.3 of RFC 5077. When the session is resumed, the evaluator shall verify on the TLS Client used for performing this test, that the TOE (TLS Server) has not advertised support for the early data extension.

ii. The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then modify the session ticket and send it as part of a new Client Hello message. The evaluator shall confirm that the TOE either (1) implicitly rejects the session ticket by performing a full

handshake (as shown in figure 3 or 4 of RFC 5077), or (2) terminates the connection in some way that prevents the flow of application data.

Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ticket may be obtained in one context for resumption in another context. There is no requirement that the session ticket be obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ticket constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

d. Test 4 [conditional]: If the TOE supports session resumption according to RFC 8446 (supported only by TLS 1.3), the evaluator shall carry out the following steps:

i. The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the pre-shared key in the ClientHello. The evaluator shall confirm that the TOE responds similarly to figure 3 of RFC 8446 after successfully reusing the pre-shared-key to resume the session. Specifically, the server must not send back a Certificate message if the session is correctly resumed. When the session is resumed, the evaluator shall verify on the TLS Client used for performing this test, that the TOE (TLS Server) has not advertised support for the early data extension.

ii. The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then modify the pre-shared key and send it as part of a new Client Hello message. The evaluator shall confirm that the TOE either (1) implicitly rejects the session ticket by performing a full handshake, or (2) terminates the connection in some way that prevents the flow of application data.

iii. The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then force the non-TOE client to attempt to establish a new connection using the previous session ticket material as a pre-shared key, but set psk_key_exchange_modes with a value of psk_ke in the Client Hello message and omit the psk_ke_dhe. The evaluator shall confirm that the TOE either (1) implicitly rejects the session ticket by performing a full handshake, or (2) terminates the connection in some way that prevents the flow of application data.

Test 1: Not applicable. The TOE claims to support session resumption using session ID values.

Test 2a: The evaluator first attempted a successful TLS negotiation capturing the TOE-generated session ID value from the server hello message. The evaluator initiated a new TLS connection sending the session ID value obtained during the successful negotiation. The evaluator observed that the TOE resumed the first TLS session because the TOE responded with a ServerHello message containing the original Session ID value followed by ChangeCipherSpec and Finished messages.

Test 2b: The evaluator attempted to open a TLS connection to the TOE where the evaluator's client recorded the session_id from a failed TLS negotiation. The evaluator observed the TOE reject the connection.

Test 3: Not applicable. The TOE claims to support session resumption using session ID values.

2.2.21.5 NDcPP30e:FCS_TLSS_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that TSS describes whether the list of supported ciphersuites can be configured or not.

Section 6.2 of [ST] states that the supported ciphersuites for HTTPS connections are not configurable.

Guidance Assurance Activities: If the TSF provides the ability of configuring the list of supported ciphersuites, the evaluator shall verify that AGD guidance includes configuration of the list of supported ciphersuites.

Section 6.2 of [ST] states that no configuration is necessary to meet the claimed algorithms.

Testing Assurance Activities: Test 1[conditional]: If the TSF provides the ability of configuring the list of supported ciphersuites, the evaluator shall establish a TLS connection using one of the possible configurations of the list of supported ciphersuites. The evaluator shall then change the configuration and repeat the test. The evaluator shall verify that the behavior of the TOE has changed according to the modification of the list of ciphers. This test shall be repeated for all supported TLS versions. If the TSF does not provide the ability of configuring the list of supported ciphersuites, this test shall be omitted.

Not applicable as the TOE does not provide the ability to configure the ciphersuites.

2.2.21.6 NDcPP30e:FCS_TLSS_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: According to RFC 8446 Section 4.2.10, a PSK is required to use the early data extension. As NDcPP only allows the use of PSK in conjunction with session resumption, a NDcPP conformant TOE which acts as TLS Server cannot use the early data extension if session resumption is not supported. For TOEs that do not support session resumption, execution of test FCS_TLSS_EXT.1.4 Test 1 is regarded as sufficient that the TOE does not support the early data extension. For TOEs that support session resumption, FCS_TLSS_EXT.1.4 Test 2(i), 3(i) or 4(i) (depending on the supported TLS versions and the way session resumption is implemented) ensure that the TOE does not support the early data extension.

Refer to FCS_TLSS_EXT.1.4 Test 1.

2.2.21.7 NDcPP30e:FCS_TLSS_EXT.1.7

TSS Assurance Activities: The evaluator shall verify in the TSS that, for TLS 1.3, the TOE shall not permit out-of-band provisioning of pre-shared keys (PSKs) in the evaluated configuration.

Section 6.2 of [ST] states that the TOE shall not permit out-of-band provisioning of pre-shared keys (PSKs) in the evaluated configuration.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Section 6.2 of [ST] states that there is no configuration ability for permitting OOB provisioning of pre-shared keys.

Testing Assurance Activities: None Defined

2.2.21.8 NDcPP30e:FCS_TLSS_EXT.1.8

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

a. Test 1 [conditional]: If 'support secure renegotiationâ€¦â€' is selected, the evaluator shall use a network packet analyzer/sniffer to capture a TLS 1.2 handshake between the two TLS endpoints. The evaluator shall verify that the 'renegotiation_info' extension is included in the ServerHello message.

b. Test 2 [conditional]: If 'support secure renegotiationâ€¦â€' is selected, the evaluator shall perform a TLS 1.2 handshake and modify the length portion of the field in the ClientHello message in the initial handshake to be non-zero. The evaluator shall verify that the TOE TLS server sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

c. Test 3 [conditional]: If 'support secure renegotiationâ€¦â€' is selected, the evaluator shall perform a TLS 1.2 handshake and modify the 'client_verify_data' or 'server_verify_data' value in the ClientHello message received during secure renegotiation. The evaluator shall verify that the TOE TLS server terminates the connection.

d. Test 4a [conditional, if the TOE supports TLS 1.3]: The evaluator shall follow the operational guidance as necessary to configure the TSF to negotiate TLS 1.3. The evaluator shall initiate a valid initial TLS 1.3 session, send a valid client hello on the non-renegotiable TLS channel, and observe that the TSF terminates the session.

Note: It is preferred that the TSF sends a fatal error alert message (e.g., unexpected message) in response to this, but it is acceptable that the TSF terminates the connection silently (i.e., without sending a fatal error alert).

Test 4b [conditional, if the TOE supports TLS 1.2 and rejects TLS 1.2 â€‹renegotiation attempts]: The evaluator shall follow the operational guidance as necessary to configure the TSF to negotiateTLS 1.2 and reject renegotiation. The evaluator shall initiate a valid initial TLS 1.2 session, send a valid ClientHello on the non-renegotiable TLS channel, and observe that the TSF does not perform renegotiation of the TLS channel.

(TD0899 applied)

Test 1: Performed as part of test 2, where an attempt is made with a properly formatted renegotiation_info extension.

Test 2: The evaluator attempted to establish a TLS session using openssl s_client with the TOE in two distinct cases: 1) the openssl library is modified to change the initial renegotiation_info extension to a non-zero extension length; and 2) an attempt is made with a properly formatted renegotiation_info extension. The evaluator observed the former caused a failure while the latter yielded a successful connection.

Test 3: The evaluator attempted to establish a TLS session using openssl s_client with the TOE where the openssl library is modified to increment the last byte of the renegotiation data by 2. The evaluator observed the TOE TLS server terminating the connection upon receiving the modified data.

Test 4a: Using TLSv1.3, the evaluator configured the TOE to connect to a GSS test server. During the connection the evaluator caused the server to connect normally with renegotiation. The evaluator verified that the TOE terminates the connection after receiving the Hello Request during a request for secure renegotiation.

Test 4b: Using TLSv1.2, The evaluator configured a test client to send a Hello Reset Request after a normal handshake. The evaluator observed that the TOE terminates the connection after receiving the second Hello Request.

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 Management (NDcPP30e:FIA_AFL.1)

2.3.1.1 NDcPP30e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 NDcPP30E: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 6.3 of [ST] states that the administrator can set a lockout failure count for login attempts as the TOE's default configuration does not enforce a failed login limit. If the count is exceeded, the targeted account is locked (preventing remote administrators from logging in through SSH under the locked account/username) for an administrator-configurable time limit. The TOE does not lock administrative access through local console, only remote/SSHv2or WebUI administrator access.

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.

The section entitled "Login Management" in [AGD] provides instruction on configuring the number of login attempts and the lockout time, and states that the functionality does not apply for local administrators on a console session, thus guaranteeing administrator access.

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

The evaluator performed 3 failed login attempts using a known bad password, all of which were rejected by the TOE. The evaluator immediately attempted to login using good credentials and the TOE rejected the login attempt, showing the account had been locked. The evaluator then waited a short time less than the configured lockout duration and showed that the login attempt (using good credentials) was unsuccessful. The evaluator then waited more than the configured lockout duration and showed that the login attempt (also using good credentials) was successful.

2.3.2 Password Management (NDcPP30e:FIA_PMG_EXT.1)

2.3.2.1 NDcPP30e: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 check that the TSS:

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

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

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

Section 6.3 of [ST] states that passwords can be composed of any alphabetic, numeric, and a wide range of special characters identified in the requirement. Password length can be configured by an administrator to be between 1-32 characters. The evaluator observed that the claimed range of length values does include the value 15.

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.

The section entitled "General password rules" in [AGD] identifies possible characters that can be used in a password, guidance on how to construct a strong password, and how to set the minimum password length. It states that the minimum password length ranges from 1 to 32 characters.

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

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

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

The evaluator attempted to set/change a password for a user's account using several attempts. Throughout those attempts, every upper case letter, lower case letter, digit, and special character (as specified by the SFR in [ST]) were used in a password. The evaluator also confirmed that a minimum password length was enforced by attempting to set passwords with 7 characters while a minimum length of 8 was configured (and observing the TOE reject the password).

2.3.3 Pre-Shared Key Composition - per TD0825 (MACSEC10:FIA_PSK_EXT.1)

2.3.3.1 MACSEC10:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.3.2 MACSEC10: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.

Section 6.3 of [ST] states that the TOE supports the use of pre-shared keys for MKA as defined by IEEE 802.1X-2010. The pre-shared keys are not generated by the TOE but rather the TOE will accept a PSK as a string of hexadecimal characters.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on the composition of strong PSKs, 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-basedPSKs for each protocol identified in the requirement, or generating a bit-based pre-shared key, or both.

The section entitled "MACsec Configuration (using pre-shared keys)" in [AGD] indicates that CKN and CAK are not auto-generated, are hexadecimal character values between 1 and 64 characters in length, and longer values are stronger. This section provided instructions to enter these hexadecimal keys.

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 17: (conditional, the TOE supports PSKs 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 18: (conditional, the TOE does not generate bit-based PSKs) The evaluator shall obtain a bit-based PSK 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 19: (conditional, the TOE can generate bit-based PSKs) The evaluator shall generate a bit-based PSK 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.

Test 17: The evaluator configured the MACsec functionality by specifying a CAK that is used by the PSKs for MKA. The TOE accepted the correct length of the values for the CAK which is demonstrated by the Part 1a (2 hexadecimal), 1b (64 hexadecimal), 1c (32 hexadecimal) for the key sizes of 128 bits. Similarly, the TOE accepted the correct length of the values for the CAK which is demonstrated by the Part 2a (2 hexadecimal), 2b (64 hexadecimal) and 2c (32 hexadecimal) for the key sizes of 256 bits.

Test 18: This test has been performed in MACsec10:FCS_MACSEC_EXT.1-t1. The evaluator reviewed that a successful protocol negotiation can be performed with the CKN value of 1000 and CAK value of 12345678901234567890123456789012.

Test 19: The TOE does not generate bit-based PSKs, therefore this test is not applicable.

2.3.4 Protected Authentication Feedback (NDcPP30e:FIA_UAU.7)

2.3.4.1 NDcPP30e: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 to ensure authentication data is not revealed while entering for each local login allowed.

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: The evaluator observed during testing that passwords are obscured on the console login.

2.3.5 User Identification and Authentication - per TD0900 (NDcPP30e:FIA_UIA_EXT.1)

2.3.5.1 NDcPP30e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.2 NDcPP30e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.3 NDcPP30e:FIA_UIA_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.4 NDcPP30e:FIA_UIA_EXT.1.4

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 remote authentication mechanism (e.g. SSH public key, Web GUI password, etc.) and optional local authentication mechanisms supported by the TOE. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a 'successful logon'.

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

For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not, all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no 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 administrator identification and authentication. The description shall cover authentication and identification for remote TOE administration and optionally for local TOE administration if claimed by the ST author. For each TOE component that does not support authentication of Security Administrators according to FIA_UIA_EXT.1 the TSS shall describe any unauthenticated services/services that are supported by the component.

Section 6.3 of [ST] states that in the evaluated configuration, users can connect to the TOE via a local console or remotely using SSHv2, WebUI or RestAPI. It explains that passwords can be composed of any alphabetic, numeric, and a wide range of special characters (identified in FIA_PMG_EXT.1). Passwords can be between 1-32 characters.

Section 6.2 of [ST] states that remote SSHv2 connections can be established via both public key-based and password-based authentication, and that the TOE allows use of the ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 algorithms for public key authentication.

Section 6.3 of [ST] explains that the TOE requires users to be identified and authenticated before they can access any of the TOE functions except to display a warning banner and to permit network switching services without identification or authentication.

The TOE is not distributed.

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 sections entitled "User, Password, and Session Management" and "Management Interfaces" in [AGD] describe how to configure and log in via both the local console and remote SSH and WebUI.

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 all combinations of supported credentials and login methods, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

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

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

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

The TOE offers the several user interfaces where authentication is provided and the evaluator tested each interface (local and remote) as specified by the Security Target.

Test 1 - Using each interface the evaluator performed an unsuccessful and successful logon 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.

Test 3 - Using each interface the evaluator found that no functions were available to the administrator accessing the console with the exception of acknowledging the banner.

Test 4 - The TOE is not distributed, thus tests 1 through 3 above test the only TOE component.

2.3.6 X.509 Certificate Validation (NDcPP30e:FIA_X509_EXT.1/Rev)

2.3.6.1 NDcPP30e: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 expected that either OCSP or CRL revocation checking is performed when a certificate is presented to the TOE (e.g. during authentication). 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).

b. 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.

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

d. 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 shall then attempt 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.

e. Test 4a: [conditional] If OCSP is selected, the evaluator shall configure an authorized responder or use a man-in-the-middle tool to use a delegated OCSP signing authority to respond to the TOE's OCSP request. The resulting positive OCSP response (certStatus: good (0)) shall be signed by an otherwise valid and trusted certificate with the extendedKeyUsage extension that does not contain the OCSPSigning (OID 1.3.6.1.5.5.7.3.9). The evaluator shall verify that the TSF does not successfully complete the revocation check.

Note: Per RFC 6960 Section 4.2.2.2, the OCSP signature authority is delegated when the CA who issued the certificate in question is NOT used to sign OCSP responses.

f. Test 4b: [conditional] If CRL is selected, the evaluator shall present an otherwise valid CRL signed by a trusted certificate that does not have the cRLsign key usage bit set and verify that validation of the CRL fails.

g. 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.)

h. Test 6: The evaluator shall modify any byte in the certificate signatureValue field (see RFC 5280 Section 4.1.1.3), which is normally the last field in the certificate, and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

i. 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.)

j. Test 8 (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The following tests are run when a minimum certificate path length of three certificates is implemented:

k. 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.

l. 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.

m. 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.

The TOE validates certificates as part of the TLS Session establishment with a syslog server. The evaluator performed each of the following tests on this interface.

Test 1 -- The evaluator configured the TOE and a peer with valid certificates. The evaluator then attempted to make a connection between the peer devices using TLS protected syslog. A successful connection was made. The evaluator then configured a syslog server that presented a certificate chain with an invalid certification path by deleting an intermediate CA so that the certificate chain was invalid because of a missing certificate. The connection between the TOE and the syslog server was refused by the TOE.

Test 2 -- The evaluator used the TOE's TLS client (syslog) to attempt connections to a test server. The test server then presented a certificate during the TLS negotiation where the certificate was expired. The TOE rejected the connection.

Test 3 -- The evaluator used a test server to accept connection attempts from the TOE TLS client (syslog). The test server then presented a certificate during the TLS negotiation where the certificate was valid. A packet capture was obtained of this TLS negotiation which shows that the connection was successful. The evaluator revoked certificates in the chain (individually) and attempted the same connection from the syslog client. The attempt after revoking the certificate was not successful.

Test 4 -- The evaluator configured an OCSP responder to present a certificate that does not have the OCSP signing purpose. The evaluator established a TLS session from the TOE TLS client such that the TOE receives OCSP response signed by the invalid certificate and ensured that the TLS session was not negotiated successfully.

Test 5 -- The evaluator configured a test server to present a certificate that had a byte in the first eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 6 -- The evaluator configured a test server to present a certificate that had a byte in the last eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 7 -- The evaluator configured a test server to present a certificate that had a byte in the public key of the certificate modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 8a - The evaluator configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where the ECDSA certificates have a supported named curve. The TOE accepted the certificate as valid.

Test 8b - The evaluator then configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where an intermediate CA's ECDSA certificate has an explicit curve. The TOE rejected the certificate as invalid.

Test 8c - The evaluator first attempted to upload a valid subordinate CA certificate with the elliptic curve parameters specified as a named curve. Next, the evaluator attempted to upload a certificate to the TOEs trust store that uses an explicit format version of the elliptic curve parameters. The evaluator observed that the valid certificate with named-curves was successfully imported, while the certificate using an explicitly stated curve was not imported.

2.3.6.2 NDcPP30e: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 shall confirm 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 shall confirm 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: The evaluator configured a test syslog server and a PKIX SSH client to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test server and the PKIX SSH client to connect to the TOE. In each case the evaluator observed that the TOE rejected the connections.

Test 2: The evaluator configured a test server and PKIX SSH client to present a certificate chain containing a CA certificate having the basicConstraints section but with the cA flag not set (i.e., FALSE). The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test server and the PKIX SSH client to connect to the TOE. In each case the evaluator observed that the TOE rejected the connection.

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 6.3 of [ST] states that OCSP is supported for X509v3 certificate validation. Certificates are validated as part of the authentication process when they are presented to the TOE and when they are loaded into the TOE. The following fields are verified:

- Chain length
- Certificate revocation check with OCSP
- Certificate Validity
- CA validity check
- keyUsage verification
- Signature verification
- SAN/CN check with wild card support

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.

The section entitled "Certificate Installation and Validation" in [AGD] discusses X.509 digital certificates that are used for remote logging protection. The section explains how certificates are validated and provides a list of checks made. This section indicates that the TOE verifies the validity dates of all certificates within a certificate chain, uses OCSP to verify that certificate have not been revoked, and verifies the Server Authentication, Client Authentication and OCSP signing EKU bits within certificates provided by a peer.

Component Testing Assurance Activities: None Defined

2.3.7 X.509 Certificate Authentication (NDcPP30e:FIA_X509_EXT.2)

2.3.7.1 NDcPP30e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.7.2 NDcPP30e: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 6.3 of [ST] explain that certificates are checked and if found not valid are not accepted or if the OCSP server cannot be contacted for validity checks, then the connection is rejected.

This behavior is consistent with the selection of "not accept the certificate" in FIA_X509_EXT.2.2.

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.

The section entitled "Certificate Installation and Validation" in [AGD] states that X.509 digital certificates are used by the Secure Remote Logging feature. This section then provides instructions to identify the certificate used for Secure Remote Logging.

This section also explains that the TOE treats all OCSP-related failures as a failure to authenticate the peer device's certificate.

This section also states that the TOE must be configured to have the proper network access to reach the OSCP responder, and configured with an accurate time in order to ensure the OCSP responses are valid.

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

a. Test 1: 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 established a trusted channel to a syslog server. For this channel, the TOE was an initiator of the connection from the TOE the syslog server protected by TLS. The evaluator demonstrated that when the revocation server for the certificate presented by the remote test server was available, the TOE was successful in establishing the TLS session.

The evaluator then made the revocation server inaccessible and observed that the TOE was able to successfully establish connections with the syslog server. Since this was the behavior claimed in the SFR selection for a TLS connection, this test passed.

2.3.8 X.509 Certificate Requests (NDcPP30e:FIA_X509_EXT.3)

2.3.8.1 NDcPP30e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 NDcPP30e: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.

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

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.

The section entitled "Certificate Installation and Validation" in [AGD] provides instructions for generating and installing a certificate on the TOE. This section include instructions for specifying fields for Common Name, Organization, Organizational Unit, or Country within the CSR. This section also indicates the instructions apply for

the TOE syslog client certificate, but the section entitled "Web Server Certificate" explains that these same instructions apply for a certificate used by the TOE https server.

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 generated a certificate signing request by following the instructions in the guidance documentation for generating the request. The request was then exported to an external CA. While the CSR was within the CA, the evaluator examined the CSR and found that it included the fields identified in the Security Target. The CSR was also used to generate certificates on the CA which was returned to the TOE (also through a trusted path).

Test 2 - The evaluator tested that a certificate chaining to a CA that is not trusted cannot be imported into the TOE manually. The evaluator also tested that a certificate that chained to a trusted CA and be imported into the TOE.

2.4 Security management (FMT)

2.4.1 Management of Security Functions Behaviour (NDcPP30e:FMT_MOF.1/Functions)

2.4.1.1 NDcPP30e:FMT_MOF.1.1/Functions

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

For non-distributed TOEs, the evaluator shall ensure the TSS for each administrative function identified the TSS details how the Security Administrator determines or modifies the behaviour of (whichever is supported by the

TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE).

Section 6.2 of [ST] states that audit records are protected against unauthorized access by only allowing authorized administrators to have access to local audit logs, perform an authorized deletion of those logs, or modify the behavior of audit data transmission.

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

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

Section "Secure Remote Logging" describes the configuration steps that must be taken to modify the behavior of audit data and transmit that data to an external IT entity. The section also explicitly states that only an operator with administrative privileges can modify the behavior of audit data.

Component Testing Assurance Activities: If 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection

The evaluator shall perform the following tests:

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

b. Test 2: The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as Security Administrator. The effects of the modifications should be confirmed.

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

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

The evaluator shall perform the following tests:

a. Test 1: The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.4, FAU_STG_EXT.1.5 and FAU_STG_EXT.2.

b. Test 2: The evaluator shall try to modify all security related parameters for configuration of the handling of audit data with prior authentication as Security Administrator. The effects of the modifications should be confirmed. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.4, FAU_STG_EXT.1.5 and FAU_STG_EXT.2.

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

If 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection

The evaluator shall perform the following tests:

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

b. Test 2: The evaluator shall try to modify the behaviour when Local Audit Storage Space is full with prior authentication as Security Administrator. This attempt should be successful. The effect of the change shall be verified.

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

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

The evaluator shall perform the following tests:

a. Test 1: The evaluator shall try to determine the behaviour of all options chosen from the second selection without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions without administrator authentication shall fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

b. Test 2: The evaluator shall try to determine the behaviour of all options chosen from the second selection with prior authentication as Security Administrator. This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions with Security Administrator authentication shall be successful.

Test 1: The evaluator referred to NDcPP30e:FMT_MTD/CryptoKeys-t1 where any attempts to modify the TOE's behavior prior to login as an administrator were interpreted as login credentials and failed.

Test 2: The evaluator logged into the TOE as a Security Administrator and successfully determined and modified the behavior of the transmission protocols for audit data to external entities, as well as the handling of audit data based on the selections in the [ST]. To determine the behaviour of the selected functionality and to verify the effects of modifications applied to the TOE by the administrator, the evaluator used commands to display the functionality of the TOE before and after modifications were applied.

The audit functionality when Local Audit Storage Space is Full cannot be modified.

2.4.2 MANAGEMENT OF SECURITY FUNCTIONS BEHAVIOUR (NDCPP30E:FMT_MOF.1/MANUALUPDATE)

2.4.2.1 NDCPP30E: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 see Section 2.4.1.1. There are no specific requirements for non-distributed TOEs.

The TOE is not distributed so no description required.

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).

The section entitled "Software Signing and Verification" in [AGD] explains how to verify the version of currently active software through the “show version” command, and how to query the loaded but inactive version of the software through the “show images” command in the case of delayed activation. The section entitled "Copying the software and rebooting the switch" describes how firmware validation is performed through signature verification.

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

a. Test 1: The evaluator shall try to perform the update using a legitimate update image without prior authentication as Security Administrator (either by authentication as a user with no administrator privileges or without user authentication at all - depending on the configuration of the TOE). The attempt to update the TOE should fail.

b. Test 2: The evaluator shall try to perform the update with prior authentication as Security Administrator using a legitimate update image. This attempt should be successful. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

The evaluator tested to determine that two commands used during a TOE update are not accepted by the TOE prior to a successful login. Any user that can login, is considered an administrator and can perform TOE updates.

The TOE is not distributed.

2.4.3 Management of TSF Data (NDcPP30e:FMT_MTD.1/CoreData)

2.4.3.1 NDcPP30e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For each administrative function identified in the guidance documentation that is accessible through an interface prior to administrator log-in are identified, the evaluator shall confirm that

the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

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 6.4 of [ST] explains that the TOE offers command line functions which are accessible via the CLI. The CLI is a text-based interface which can be accessed from a directly connected terminal or via a remote terminal using SSHv2. These command line functions can be used to manage every security policy, as well as the non-security relevant aspects of the TOE. The TOE also permits administrators to perform administrative tasks using an HTTPS/TLS protected communication channel offering a Web-based GUI and upload certificates through a RESTAPI interface.

This section provides a list of security functions (which include handling of X.509 certificates and managing the TOE's trust store) that are available to administrator once authenticated. It also states that none of these functions are available to any user before being identified and authenticated.

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.

The evaluator reviewed the guidance documentation while performing the guidance assurance activities in this AAR. The evaluator identified the TSF data manipulating functions and referenced the guidance documentation to demonstrate that it contained the corresponding configuration information. The evaluator documented these Guidance Assurance Activities throughout this AAR with the requirements to which they apply.

The section entitled "User, Password, and Session Management" in [AGD] states that once in secure mode, the switch does not offer any management services or access to its management functions, except for displaying a warning banner, without requiring a user to be identified and authenticated. The evaluator observed that this explains how the TOE ensure that only administrators have access to TSF-data manipulating functions.

The section entitled "Certificate Installation and Validation" in [AGD] provides instructions for managing the TOE trust store. This section explains how to configure the CA certificate of the remote syslog server's PKI, how to

generate a certificate signing request (CSR) for the TOE syslog client and how to import a certificate created externally using that CSR. These steps explain how to designate a CA certificate as a trust anchor.

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 separate testing for FMT_MTD.1/CoreData is required.

2.4.4 Management of TSF Data (NDcPP30e:FMT_MTD.1/CryptoKeys)

2.4.4.1 NDcPP30e: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 Section 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 names the operations that are performed.

Section 6.4 of the [ST] states that only administrators can perform management operations including the command to generate, import and delete cryptographic keys as defined by Table 8 Key Zeroization.

These key manipulation operations are performed using commands available through the Command Line Interface.

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

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

The section entitled "Certificate Installation and Validation" in [AGD] provides instructions to generate keys used as part of a certificate signing request (CSR). This section also explains how to load trusted CA and TOE certificates. This sections also describes how the TOE certificate and trusted CA certificates can be listed and deleted.

The section entitled "Web Server Certificate" explains that the same procedures to load a trusted CA, generate a CSR and import a certificate are used for the Web Server Certificate as are describe in the section entitled "Certificate Installation and Validation". The administrator is simply required to issue a command to associate the new certificate with the TOE https-server.

The section entitled "SSH host key generation" describes how to generate SSH host keys. This section also indicates that when a host key is generated, it overwrites (deletes) the current key of the same type.

The section entitled “SSH authorized keys” describes how to import public keys which are used for user SSH public key authentication.

All of the above key manipulation is performed using commands available through the Command Line Interface.

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

a. Test 1: The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. 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.

b. Test 2: The evaluator shall try to perform at least one of the related actions with prior authentication as Security Administrator. This attempt should be successful.

The evaluator attempted to import a cryptographic key (SSH public key) for a user before being authenticated as an administrator. This attempt was unsuccessful. The evaluator logged into the TOE as an authorized administrator and attempted to import a cryptographic key (SSH public key) and observed that the import was successful.

2.4.5 Specification of Management Functions - per TD0880 (NDcPP30e:FMT_SMF.1)

2.4.5.1 NDcPP30e: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.

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

Section 6.4 of [ST] identifies all the management functions:

- Ability to administer the TOE locally and remotely;
- Ability to configure the access banner;
- Ability to configure the session inactivity time before session termination or locking;
- Ability to update the TOE, and to verify the updates using [digital signature] capability prior to installing those updates;
- Ability to configure the authentication failure parameters for FIA_AFL.1;
- Ability to modify the behavior of the transmission of audit data to an external IT entity;
- Ability to manage the cryptographic keys,
- Ability to configure the cryptographic functionality,
- Ability to set the time which is used for time-stamps;
- Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors,
- Ability to import X.509v3 certificates to the TOE's trust store

- Ability to manage the trusted public keys database

It explains that console, WebUI, and SSH CLI access provide full administrative capabilities.

Section 6.3 of [ST] explains that the local console must be co-located with the TOE and protected with equivalent physical security measures. The section entitled "Connecting to the console port" in [AGD] provides a warning that the local console must be physically protected.

Component Guidance Assurance Activities: See TSS Assurance Activities

See the TSS assurance activity. The [AGD] was used throughout the evaluation and the evaluator was able to perform all required functions as part of testing.

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

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

2.4.6 Specification of Management Functions (MACsec) - per TD0803, TD0825, TD0840, & TD0889 (MACSEC10:FMT_SMF.1/MACSEC)

2.4.6.1 MACSEC10:FMT_SMF.1.1/MACSEC

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.

Section 6.4 of [ST] indicates that the TOE support the following management functions which correspond to those identified by the requirement.

- Manage a PSK-based CAK and install it in the device;
- Manage the Key Server to create, delete, and activate MKA participants as specified in 802.1X-2020, sections 9.13 and 9.16 (cf. MIB object ieee8021XKayMkaParticipantEntry) and section.12.2 (cf. function createMKA());

- Specify a lifetime of a CAK; and
- Enable, disable, or delete a PSK-based CAK using the MIB object ieee8021XKayMkaPartActivateControl.

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.

The section entitled "MACsec Configuration (using pre-shared keys)" in [AGD] describes the ability of the security administrator to manage a PSK-based CAK and install it in the device. The section "Key Server Priority" in [AGD] provides instructions to manage key server to create, delete, and activate MKA participants. The section "Configuration of CAK Lifetime" in [AGD] specifies instructions to configure the lifetime of a CAK.

The section entitled "MACsec Configuration (using pre-shared keys)" in [AGD] describes the ability of the security administrator to enable, disable or delete a pre-shared key.

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 PSKs 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.

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 20: 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.

(conditional, the length of the CKN is not configurable) Repeat this test again for CKN of invalid lengths zero and 33, and observe failure.

(conditional, the length of the CKN is configurable) The evaluator shall observe that the admin guidance instructs the admin to configure the TOE to only allow CKN lengths of 1-32 bytes.

(TD0803 applied)

Test 21: (conditional, 'Cause key server to generate a new group CAK...' is selected) The evaluator shall test the ability of the TOE to enable and disable MKA participants using the management function specified in the ST. The evaluator shall install PSKs 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 cannot communicate with device C. The evaluator shall re-enable the MKA participant of device C and observe that the TOE is now able to communicate with devices B and C. (TD0889 applied)

Test 22: 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:

Test 22.1 (Conditional: The TOE supports MKA keychains with multiple CKN/CAKs): 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. (TD0840 applied)

Test 22.2: 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 23: (conditional, '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 ieee8021XKayCreateNewGroup to true), and observe that the TOE distributes a new group CAK.

Test 20: The evaluator configured the MACsec functionality by specifying a CKN and that the PSK is to be used as a CAK. The TOE accepted the correct length of the octet values for the CKN, which is demonstrated by the Part 1a (2 hexadecimal) and 1b (64 hexadecimal) for the key sizes of 128 bits. Similarly, the TOE accepted the correct length of the values for the CKN, which is demonstrated by the Part 2a (2 hexadecimal) and 2b (64 hexadecimal) for the key sizes of 256 bits.

Test 21: Not applicable as 'Cause key server to generate a new group CAK...' is not selected.

Test 22: 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 23: The TOE does not generate a new group CAK, therefore this test is not applicable.

2.4.7 Restrictions on Security Roles (NDcPP30e:FMT_SMR.2)

2.4.7.1 NDcPP30e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.7.2 NDcPP30e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.7.3 NDcPP30e: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 (e.g. if local administrators and remote administrators have different privileges or if several types of administrators with different privileges are supported by the TOE).

Section 6.4 of [ST] states the TOE includes a manager account that corresponds to the required 'Authorized Administrator' also referred to as 'Security Administrator' in some requirements or text.

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.

The sections entitled "User, Password, and Session Management" and "Management Interfaces" in [AGD] describe how to configure and log in via both the local console and remote SSH and WebUI.

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, if the TSF shall be validated against the Functional Package for Secure Shell referenced in Section 2.2 of the cPP; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

Testing of TOE security protocols (e.g., SSH and TLS) along with the manipulation of X509 certificates was conducted using primarily the TOE CLI that is available via SSHv2. Refer to protocol testing results.

Testing of timeout values, authentication, TOE updates, self-tests, and changes to time were tested using CLI over SSH.

The TOE is not distributed.

2.5 Protection of the TSF (FPT)

2.5.1 Protection of Administrator Passwords (NDcPP30e:FPT_APW_EXT.1)

2.5.1.1 NDcPP30e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 NDcPP30e: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 6.5 of [ST] explains the TOE maintains and protects passwords for administrative user accounts as authentication data. Locally defined passwords are not stored in plaintext form, instead the TOE stores the

password as salted SHA-512 hashes. The TOE does not offer any functions that will disclose to any user a plain text password.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.2 Protection of CAK Data (MACSEC10:FPT_CAK_EXT.1)

2.5.2.1 MACSEC10: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 6.5 of [ST] states that CAK is stored in an encrypted form and the TOE does not offer any functions that will disclose to any user a CAK.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.3 Failure with Preservation of Secure State - per TD0816 (MACSEC10:FPT_FLS.1)

2.5.3.1 MACSEC10:FPT_FLS.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 indicates that the TSF will attain a secure/safe state (e.g., shutdown) if 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 attain a secure/safe state (e.g., shutdown) if a self-test failure is detected. (TD0816 applied)

Section 6.5 of [ST] states that if the TOE encounters a self-test failure, failure of integrity check of the TSF executable image, or failure of noise source health tests it will shut down. The TOE will not restart as long as it has a failure and will need administrator intervention.

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.

The section entitled "MACsec Selftest" in [AGD] provides instructions for the administrator to issue two commands to ensure that POST self-tests failures, FIPS module failures, and entropy failures result in secure failure. It indicates that in the event of such a failure, an EMERGENCY log will be issued and the switch will immediately reboot. The reboot will continue as long as the self-test continues to fail.

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 24: For each failure mode specified in the ST that can be deliberately induced to fail, the evaluator shall ensure that the TOE attains a secure state (e.g., shutdown) after initiating each failure mode type. For TOEs with redundant failover capability, the evaluator shall determine that the failed components attain a secure state 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. (TD0816 applied)

Test 24: The evaluator used 3 special builds provided by the vendor to cause failures. Each build caused a specific type of failure: one caused the integrity check of the image to fail, one caused a failure of each FIPS self-tests, and one caused a failure of noise source health tests. The evaluator uploaded each build (one at a time) into boot flash on the device and then configured the TOE to boot to that failure image. During TOE initialized the failures were detected and the TOE generated an EMERGENCY log message indicating the failure and the TOE rebooted. This cycle continued until the evaluator was able to use console access to cause the TOE to boot to a valid image. The evaluator tested using each of the 3 failure images.

2.5.4 Replay Detection - per TD0746 & TD0881 (MACSEC10:FPT_RPL.1)

2.5.4.1 MACSEC10:FPT_RPL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.4.2 MACSEC10: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 MKPDUs and MPDUs and how replayed MKPDUs and MPDUs are handled by the TSF. (TD0881 applied)

Section 6.5 of [ST] states that the TOE detects and logs all attempts to replay MPDUs and MKA frames. This section goes on to describe the approach used by the TOE to detect replay.

The TOE detects replay by allowing an administrator to enables replay protection within the MACsec policy context with either a default or admin-specified window size. With replay protection enabled, packets are expected to arrive within the replay protection window number of packets. For example, with a window size of 10, any packet arriving out-of-sequence by more than 10 packets will be discarded. A window size of 0 (the default) enforces strict order of packet reception, discarding all packets not received in perfect sequence. The no form of this command disables replay protections and resets the window size to its 0 default.

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 MN and PN values increase for each direction.

Test 25: 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. (TD0746 applied)

Test 26: The evaluator will capture frames during an MKA session and record the lowest MN from the Basic Parameter set observed in a particular time range. The evaluator shall then send a frame with a lower MN, and then verify that this frame is dropped. The evaluator will verify that the device logged this event.

(TD0881 applied)

Test 25: The evaluator enabled replay protection on the TOE in order to run this test. The evaluator 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. The evaluator then attempted to send the same traffic, which contains an old packet number (PN). The TOE successfully detects the invalid PN and drops 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 detects the invalid PN, drops the traffic, and reports the error in an audit log.

Test 26: This test has been performed in MACsec10:FPT_RPL.1 test 25.

2.5.5 Replay Detection for XPN - per TD0728 (MACSEC10:FPT_RPL_EXT.1)

2.5.5.1 MACSEC10:FPT_RPL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.2 MACSEC10:FPT_RPL_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 includes XPN in the description of how replay is detected for MPDUs and how replayed MPDUs are handled by the TSF.

Section 6.5 of [ST] states that the TOE supports extended packet numbering (XPN) per IEE 802.1AE-2018 using a MACsec policy configured GCM cipher suites that are based on 128-bit or 256-bit AES keys. When leveraging an XPN cipher suite, the counter used to detect replayed packets is extended to 64 bits. All other replay detection logic and mechanisms remain the same.

Component Guidance Assurance Activities: If the use of XPN or the XPN cipher suites used by the TOE are configurable, the evaluator shall examine the guidance documentation to determine that it describes how this is configured.

The section entitled "MACsec Configuration (using pre-shared keys)" in [AGD] provides instructions to define a MACsec policy that includes identifying the GCM ciphersuite that is to be used by the TOE. This section indicates the TOE supports GCM cipher suites with both 128-bit and 256-bit AES keys. It also indicates that Extended Packet Numbering (XPN) cipher suites are supported. The example shows how to configure two sample ciphersuites 'gcm-aes-256' and 'gcm-aes-xpn-256'.

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

Test 29: The evaluator shall establish a MACsec connection between the TOE and a test system using the GCM-AES-XPN-128 cipher suite if selected, otherwise use GCM-AES-XPN-256. The evaluator shall write or obtain a script to send a small frame with a known payload (such as five bytes of all zeroes) to the TOE. The evaluator shall activate a packet capture tool on the connection between the TOE and the test system and then use the test system to send this frame to the TOE 4,294,967,267 (2^32 + 1) times. The evaluator shall use the packet capture tool to verify that for the first and last frames sent, the least significant 32 bits are the same. This means the most significant bits should have been incremented during this test. Since the IV is different the two encrypted frames should be different.

Note that if traffic is sent to the TOE at a rate of 10 GB/s, this will take approximately 5 minutes as per IEEE 802.1AE-2018.

Test 30: If both cipher suites were selected, then the evaluator shall reconfigure the TOE using the second cipher suite and rerun Test 29 to demonstrate support for both cipher suites. (TD0728 applied)

Test 29: The evaluator established a MACsec connection w/ the TOE as one peer while collecting a packet capture of the MACsec traffic. This connection used GCM-AES-XPN-256. The evaluator sent a small frame 2^32+1 times to the TOE. The evaluator observed that the first 32 bits were the same in the first frame and in the last frame. The evaluator also observed that the two encrypted frames were different.

Test 30: The evaluator repeated the prior test using GCM-AES-XPN-128 and observed the same correct behavior.

2.5.6 Protection of TSF Data (for reading of all symmetric keys) (NDcPP30e:FPT_SKP_EXT.1)

2.5.6.1 NDcPP30e: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 any interface designed specifically for that purpose, by any enabled role, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

Section 6.5 of [ST] states the TOE stores it's SSH host private keys and TLS server certificate private keys in plaintext form but does not offer any functions to output the cryptographic key value. Similarly, there is no function to view any other encrypted key.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.7 Reliable Time Stamps (NDcPP30e:FPT_STM_EXT.1)

2.5.7.1 NDcPP30e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.2 NDcPP30e: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 6.5 of [ST] explains the TOE is a hardware appliance that includes a reliable real-time clock, for maintaining time. The TOE uses the clock to support several security functions including timestamps for audit records, timing elements of cryptographic functions, and inactivity timeouts. The TOE provides the administrator the ability to manually set the clock.

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

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.

The section entitled "Date and Time Configuration" in [AGD] explains how to set the date and time. The section entitled "Updating Date and Time through NTP" explains that while the protocol is supported, it is not claimed nor tested in the evaluated configuration.

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

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

c. Test 3 [conditional]: If the TOE obtains time from the underlying VS, 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.

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.

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 found that the time was successfully changed.

Test 2: The TOE does not support the use of NTP to set time.

Test 3: The TOE does not obtain time from an underlying VS system, thus this test is not applicable.

2.5.8 TSF TESTING - PER TD0836 (NDCPP30E:FPT_TST_EXT.1)

2.5.8.1 NDCPP30E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.8.2 NDCPP30E:FPT_TST_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 details each of the self-tests that are identified by the SFR; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. If more than one failure response is listed in FPT_TST_EXT.1.2, the evaluator shall examine the TSS to ensure it clarifies which response is associated with which type of failure.

For distributed TOEs the evaluator shall examine the TSS to ensure that it details which TOE component performs which self-tests and when these self-tests are run. The evaluator shall also examine the TSS to ensure it describes how the TOE reacts if one or more TOE components fail self-testing (e.g. halting and displaying an error message; failover behaviour).

(TD0836 applied)

Section 6.5 of [ST] states that the TOE performs a suite of self-tests to verify its integrity. The TOE performs an integrity test of its firmware (by validating the firmware's RSA digital signature) product and also performs a set of power-up self-tests including AES, SHS, HMAC, RSA, ECDSA and DRBG known answer tests. The TOE automatically performs its known answer power on self-tests (POST) on its cryptography library by computing a trial cryptographic operation (e.g., AES encryption) and then comparing the calculated result to the known correct result (already compiled into the library). This ensures that the TOE's implementations work correctly. Should any of the tests fail, the TOE halts the boot process.

The TOE is not distributed.

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.

The section entitled "Self Tests" in [AGD] explains the errors that may result from the self-tests, and recommends rebooting the device or loading new firmware in the event of a self-test error.

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, Section 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

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

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

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.

(TD0836 applied)

The evaluator logged into each TOE device and initiated a restart. While the TOE rebooted, the evaluator observed messages which indicated the TOE performed the TOE self-testing as claimed by the Security Target. The TOE also generated boot messages indicating that it was the image that was about to execute.

2.5.9 TRUSTED UPDATE (NDCPP30E:FPT_TUD_EXT.1)

2.5.9.1 NDCPP30E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.9.2 NDCPP30E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.9.3 NDCPP30E: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 shall 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. 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 of the update, and the actions that take place for both successful and unsuccessful signature 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.

Section 6.5 of [ST] explains that upgrading the ArubaOS firmware is a manual process performed by an authorized administrator. An administrator can use the “show version” and “show images” commands to query the TOE’s loaded and active firmware versions.

This section explains that the TOE firmware is digitally signed with RSA 3072 using SHA-256. The TOE uses one of two embedded (within the TOE’s firmware images) public keys to verify the digital signature. The firmware is readily available on the HPE website. Uploading the firmware to the devices does require successful authentication to the devices in order to issue the CLI commands needed to update. The TOE will validate the firmware validation during the loading process and will reject the firmware if validation fails. HPE signs the firmware images and includes the HPE signing public keys within the running firmware. Once the TOE has successfully verified a new firmware image, it is loaded and becomes active upon the next reboot.

The TOE does not claim to support automatic checking for updates and is not distributed.

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.

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

The section entitled "Software Signing and Verification" in [AGD] explains how to verify the version of currently active software through the “show version” command, and how to query the loaded but inactive version of the software through the “show images” command in the case of delayed activation. The sections entitled "Firmware Validation" and "Copying the software and rebooting the switch" describe how firmware validation is performed through signature verification and explains what happens with a successful and unsuccessful verification. The section entitled "Copying the software and rebooting the switch" describes how to copy the image onto the switch for an update.

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:

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

ii. An image that has not been signed.

iii. An 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)

iv. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

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

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

Test 1: Prior to performing an update, the evaluator verified the TOE version using TOE commands. The evaluator then followed guidance to install a valid update to the TOE. Upon successful installation, the evaluator verified the TOE version once again and confirmed that the version after the successful update has changed as expected.

Test 2: The evaluator attempted to perform a TOE update using a legitimate update that was modified using a hex editor. The TOE rejected the modified update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with the digital signature removed. The TOE rejected the modified update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with a digital signature which was created using the wrong private key (i.e., wrongSignature). The TOE rejected the modified update and the product version did not change.

Test 3: Not applicable. The TOE does not use published hashes for updates.

2.6 TOE ACCESS (FTA)

2.6.1 TSF-INITIATED TERMINATION (NDCPP30E:FTA_SSL.3)

2.6.1.1 NDCPP30E: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 6.6 of [ST] states the TOE can be configured by an administrator to set an interactive session timeout value (any integer value in minutes). The inactivity timeout is 30 minutes by default. This session timeout value is applicable to both local and remote CLI sessions. An SSHv2, Web or RestAPI remote session that is inactive (i.e., no commands issuing from the remote client) for the defined timeout value will be terminated. The TOE RestAPI interface is not interactive, but does enforce the same session timeout as the Web interface.

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.

The section entitled "Session Timeout" in [AGD] explains how to configure the inactivity time period for both local and remote sessions through the “session-time” command. Similarly, the section entitled “Password Policy and Session Configuration for Web Interface” explains how to configure the inactivity time period for Web interface and RestAPI interface.

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

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

The evaluator followed the guidance to configure the session timeout periods for SSH CLI, WebUI and RestAPI remote sessions. The evaluator confirmed that the session was terminated after the configured time period. The inactivity time period was configured for periods of 1 minute, 3 minutes and 5 minutes.

2.6.2 User-initiated Termination (NDcPP30e:FTA_SSL.4)

2.6.2.1 NDcPP30e: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 remote administrative session (and if applicable the local administrative session) are terminated.

Section 6.6 of [ST] states the TOE allows a user to terminate both local and remote sessions (including SSH, Web and RestAPI sessions). The TOE accepts the 'exit' command to terminate local and remote CLI sessions. The TOE offers a logout Web operation and a RestAPI logout URL to terminate Web and RestAPI sessions.

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

The section entitled "Session Timeout" in [AGD] explains how to terminate a local or remote session with the "exit" command.

The instructions to terminate a Web session is provided in the section entitled "Password Policy and Session Configuration for Web Interface", where the administrator is told how to locate the 'logout' operation on the Web interface.

The instructions to terminate a RestAPI session is provided in the section entitled "Connecting to the Rest API Interface through the Management Port". This section indicates that a logout URL is offered to terminate RestAPI sessions.

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

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

b. Test 2: For each method of remote administration, the evaluator shall initiate an interactive remote session with the TOE. The evaluator shall then follow the guidance documentation to exit or log off the session and 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, a Web and a RestAPI connection and observed that the session ended and the connections was terminated. The SSH connection was terminated using the 'exit' command, the Web connection was terminated by the 'logout' operation, and the RestAPI connection was terminated by the 'logout URL'.

2.6.3 TSF-initiated Session Locking (NDcPP30e:FTA_SSL_EXT.1)

2.6.3.1 NDcPP30e: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 6.6 of [ST] states the TOE terminates local sessions that have been inactive for an administrator-configured period of time.

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.

The section entitled "Session Timeout" in [AGD] explains how to set the inactivity timeout to ensure administrators are logged out after a period of inactive usage.

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 followed the guidance to configure the idle timeout periods for the Local Console session and confirmed that the session was terminated after the configured time period. The inactivity time period was configured using the "session-timeout" command for periods of 1 minute, 3 minutes and 5 minutes.

2.6.4 Default TOE Access Banners (NDcPP30e:FTA_TAB.1)

2.6.4.1 NDcPP30e: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/or remote) available to the Security Administrator (e.g., serial port,

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

Section 6.3 of [ST] states that users can connect to the TOE via a local console or remotely using SSHv2, WebUI or RestAPI. These are the only methods the evaluator observed as being available for administration.

Section 6.6 of [ST] states that the TOE can be configured to display a warning banner before administrators successfully establish interactive sessions with the TOE, (i.e., console, SSH CLI and WebUI), allowing users to terminate their session prior to performing any functions.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

The section entitled "Configuring Login Banner" in [CC-Admin] explains how to configure the banner messages.

Component Testing Assurance Activities: The evaluator shall also perform the following test:

a. Test 1: The evaluator shall follow the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of 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. The TOE also provides a banner for HTTPS connections prior to login.

2.7 Trusted path/channels (FTP)

2.7.1 Inter-TSF trusted channel (NDcPP30e:FTP_ITC.1)

2.7.1.1 NDcPP30e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.2 NDcPP30e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 NDcPP30e: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 6.7 of [ST] explains the TOE must be configured to use TLS to ensure that any exported audit records are sent only to the configured server so they are not subject to inappropriate disclosure or modification. The TOE is acting as a client in this instance and receives a certificate from the audit server for identification. See section 6.2 for a description of the TLS protocol implemented by the TOE.

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.

The section entitled "Secure Remote Logging" in [AGD] explains how to configure secure remote logging, and states that the TLS tunnel must be restarted on the audit server if the connection is unintentionally broken.

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 TLS to protect communications with an external audit server (syslog server).

A successful TOE TLS connection supporting communication to an external audit server was established. Examining the packet capture from that test evaluators saw that the connection between the TOE component and the external syslog server was established; the TOE initiated the connection; and Application data that was transferred is encrypted (i.e., not plaintext).

The evaluator began a packet capture of 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 for a period of time before reconnected the wiring. Because the TOE automatically reconnects broken TLS 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.

This disconnection was performed with a short duration where the TOE simply continued use of the existing TLS session, and with a long duration where the TOE fully negotiated a new TLS session (after discarding the old session).

Upon completion of these activities, the resulting transcripts and packet captures were inspected. This data showed the following:

Test 1: The TOE support for TLS protected syslog was demonstrated.

Test 2: The TOE initiated a TLS connection for TLS protected syslog.

Test 3: Syslog communication was not plaintext.

Test 4: A physical disruption in the network resulted in a TLS session interruption and no data was transmitted unprotected.

2.7.2 Inter-TSF Trusted Channel (MACsec Communications) (MACSEC10:FTP_ITC.1/MACSEC)

2.7.2.1 MACSEC10:FTP_ITC.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.2 MACSEC10:FTP_ITC.1.2/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.3 MACSEC10:FTP_ITC.1.3/MACSEC

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: None Defined

Component Testing Assurance Activities: None Defined

2.7.3 Trusted Path (NDcPP30e:FTP_TRP.1/Admin)

2.7.3.1 NDcPP30e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.3.2 NDcPP30e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.3.3 NDcPP30e: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 6.7 of [ST] states the TOE provides SSHv2 secured remote administration. Additionally, an HTTPS/TLS connection is available which presents a Web GUI administrative interface and the RESTAPI interface. The administrator can initiate the remote session. The remote session is secured (disclosure and modification) using CAVP tested cryptographic operations, and all remote security management functions require the use of the SSHv2 protected channel or HTTPS/TLS protected channel.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

The section entitled "User, Password, and Session Management" and "Management Interfaces" sections of the [AGD] describe how to configure and log in via the remote SSH interfaces. The section entitled “Connecting to the Management Port” explains how to establish a session to the switch using the SSH and Web interface by using SSH client software or an HTTPS browser. The section entitled ‘Connecting to the Management Port’ in [AGD] - provides step 3 and 4 that explain how to log into the HTTPS interface. The section entitled “Connecting to the Rest API Interface through the Management Port” in [AGD] provides instructions to login using the REST API interface.

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 to provide the trusted path (with protection from disclosure and modification) for all remote administration sessions.

The evaluator performed the following on the SSH protected CLI and the HTTPS/TLS protected WebUI:

a) The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE.

b) The evaluator connected to the TOE and performed a login using an administrator account

c) The evaluator then terminated the connection by 'Logout' and terminated the packet capture.

The evaluator performed the following on the HTTPS/TLS protected RestAPI interface:

a) The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE.

b) The evaluator initiated a RestAPI operation providing a valid username and password for an administrator.

This data showed the following:

Test 1: The TOE support for SSH protected remote administration was demonstrated.

Test 2: Remote administration sessions protected by SSH did not contain plaintext data.

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 programming 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). Explicitly labeling TSFI as security relevant or non-security relevant is not necessary. A TSFI is implicitly security relevant if it is used to satisfy an evaluation activity, or if it is identified in the ST or guidance documentation as adhering to the security policies (as presented in the SFRs). The intent is that these interfaces will be adequately tested and having an understanding of

how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied. According to the description above 'security relevant' corresponds to the combination of 'SFR-enforcing' and 'SFR-supporting' as defined in CC Part 3, paragraph 224 and 225.

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

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 shall use 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 a TSFI that is explicitly 'mapped' to invoke the desired functionality. For example, generating a random bit string or destroying a cryptographic key that is no longer needed are capabilities that may be specified in SFRs, but are not invoked by an interface.

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

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: It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to Security Administrators and users (as appropriate) as part of the TOE.

Note that additional Evaluation Activities for the guidance documentation in the case of a distributed TOE are defined in Appendix B.4.2.1.

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR. For the related evaluation activities, the evaluation evidence documents Security Target, AGD documentation (user guidance) and functional specification documentation (if provided) shall be used as input documents. Each input document is subject to ALC_CMS.1-2 requirements.

In addition, the evaluator performs the EAs specified below.

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR. For the related evaluation activities, the evaluation evidence documents Security Target, AGD documentation (user guidance) and functional specification documentation (if provided) shall be used as input documents. Each input document is subject to ALC_CMS.1-2 requirements.

In addition, the evaluator performs the EAs specified below.

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

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 implementation associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic implementations was not evaluated nor tested during the CC evaluation of the TOE.

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 implementation associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic implementations was not evaluated nor tested during the CC evaluation of the TOE.

b. The evaluator shall verify that this process includes instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

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

As identified throughout this AAR, the [AGD] provides instructions for configuring the TOE's cryptographic security functions. The [AGD] provides instructions for configuring the cryptographic algorithms and parameters used for the evaluated configuration. The [AGD] is clear that no other cryptographic configuration has been evaluated or tested. There are warnings and notes throughout the [AGD] regarding use of functions that are and are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP30e:FPT_TUD_EXT.1.

3.2.2 PREPARATIVE PROCEDURES (AGD_PRE.1)

Assurance Activities: 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 Security Administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that Security Administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

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 Security Administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

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 on how to manage the TSF as a product and as a component of the larger Operational Environment in a manner that allows to preserve integrity of the TSF.

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

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 mandate that they shall be changed.

The evaluation team had the [AGD] to use when configuring the TOE.

The completeness of the [AGD] 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: 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.

ALC_CMC.1-1 The evaluator shall check that the TOE provided for evaluation is labelled with its reference.

989 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

Alternatively, the unique reference provided for the TOE may be the combination of the unique reference of each component from which the TOE is comprised (e.g. in the case of a composed TOE).

ALC_CMC.1-2 The evaluator shall check that the TOE references used are consistent.

If the TOE is labelled more than once then the labels have to be consistent. For example, it should be possible to relate any labelled guidance documentation supplied as part of the TOE to the evaluated operational TOE. This ensures that consumers can be confident that they have purchased the evaluated version of the TOE, that they have installed this version, and that they have the correct version of the guidance to operate the TOE in accordance with its ST.

The evaluator also verifies that the TOE reference is consistent with the ST.

If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumers will be able to determine whether they have the appropriate items.

The evaluator verified that the ST, TOE and Guidance 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: When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

ALC_CMS.1-1 The evaluator shall check that the configuration list includes the following set of items:

a) the TOE itself;

b) the evaluation evidence required by the SARs in the ST.

ALC_CMS.1-2 The evaluator shall examine the configuration list to determine that it uniquely identifies each configuration item.

1103 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

See section 3.3.1 above 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: 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 shall 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 Appendix 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.

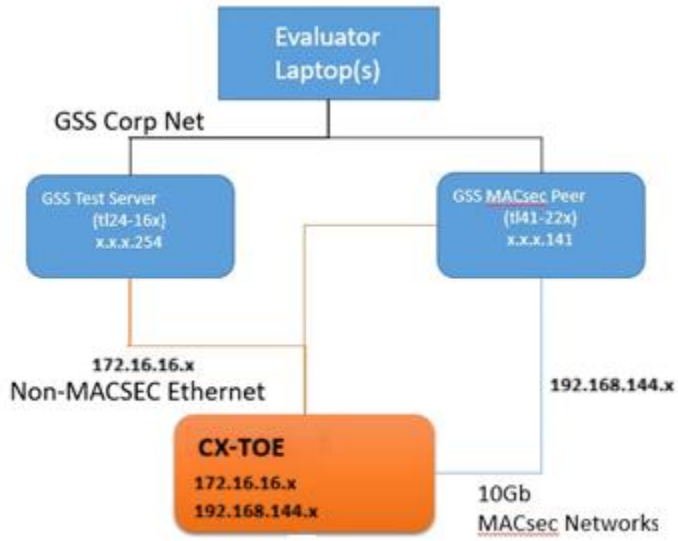

Figure 1 Test Setup

TOE Platforms Tested:

- CX-5420 running AOS 10.16
- CX-6200M running AOS 10.16
- CX-6300M running AOS 10.16
- CX-8360 running AOS 10.16
- CX-9300S running AOS 10.16

Supporting Products:

- None

Supporting Software:

The Gossamer Test servers utilized both a windows and Ubuntu environment. The Windows supporting software included the following.

- Windows 8.0, 10.0 and 11 (Evaluator Laptops)
- Wireshark version 4.0.6, 4.0.7 and 4.0.8
- Windows SSH Client - Putty version 0.74, 0.76 & 0.78 (used to connect to device console and SSH)
- OpenSSL 1.0.2g
- Openssh client version 7.2p2
- Big Packet Putty, Openssh-client version 7.2p2

- Rsyslog version 8.16.0
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Nmap version 7.01 (Linux)
- Stunnel 5.30

The Gossamer Test servers with an Ubuntu environment acted as platforms to initiate testing. The test servers also acted as a syslog server.

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 shall follow a set of well-defined activities and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

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 components[7] that compose the TOE. Hardware components should identify compute-capable hardware components, at a minimum

that must include the processor, and where applicable, discrete crypto ASICs, TPMs, etc. used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic implementations, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis.

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

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

b. a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, Table 2]

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

The evaluator shall formulate hypotheses in accordance with process defined in Appendix A. The evaluator shall document 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 evaluator searched the MITRE CVE Database, National Vulnerability Database, and CVE details (https://www.cve.org/, https://web.nvd.nist.gov/vuln/search, and https://www.cvedetails.com/vulnerability-search.php, ref CVE), Known Vulnerability Exploit Catalog (https://www.cisa.gov/known-exploited-vulnerabilities-catalog, ref KEV), Vulnerability Notes Database (http://www.kb.cert.org/vuls/, ref VND), Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities, ref Rapid7), Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories, ref ZDI), Tenable Network Security (http://nessus.org/plugins/index.php?view=search, ref TEN), and Offensive Security Exploit Database (https://www.exploit-db.com/, ref EDB) on 8/14/2026 (from 10/1/2023) with the following search terms: "ArubaOS", "AOS 10.16", "TLS", "SSH", "Cortex A72", "NPX 1046A", "Xeon D-1518", "Xeon D-1527", "Xeon D-1537", "Xeon D-1637", "Atom C2538", "AOS-CX", "AOS-CX RSA Engine", "AOS-CX Crypto", and "AES ECB".

Testing Assurance Activities: The following additional tests shall be performed: a. [conditional]: If theTOE 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/).

The TOE utilizes TLS ciphersuites that contain the RSA algorithm in the key exchange portion of the TLS handshake. The evaluator used testssl.sh (https://testssl.sh/bleichenbacher/), which is a third-party developed script that tests TLS servers for the ROBOT vulnerability. The evaluator concluded that none of the TOE's TLS servers are vulnerable to the ROBOT attack.