www.GossamerSec.com

Assurance Activity Report for Infinera GX G42 Optical Network Platform Running Converged OS (COS) 6.2.10

Version 0.3
01/15/2025

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

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

Document: AAR-VID11502

Revision History

Revision	Date	Authors	Summary
Version 0.1	12/23/2024	Sykes	Initial draft
Version 0.2	01/13/2025	Sykes	Response to ECR comments
Version 0.3	01/15/2025	Sykes	Response to ECR comments

The TOE Evaluation was Sponsored by:

Infinera Corporation
9005 Junction Dr, Suite C
Annapolis Junction, MD 20701

Evaluation Personnel:

- Kevin Cummins
- Katie Sykes

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 Infinera GX G42 Optical Network Platform running Converged OS (COS) 6.2.10 NDcPP22e evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

This section explains why the test subset was adequate to address all product installations.

1.1.1 Evaluated Platform Equivalence

The Target of Evaluation (TOE) is Infinera GX G42 Optical Network Platform running Converged OS (COS) version 6.2.10. The TOE is comprised of both software and hardware. The hardware is comprised of the GX G42 chassis. The software is comprised of the Converged OS (COS) version 6.2.10.

The Infinera GX G42 TOE is a 3 RU chassis that accommodates the pluggable modules identified in the table below:

Hardware Model	Specifications
GX G42 3 RU chassis; includes 1 GX-FAN-CTRL, 2 GX-FAN-XMM4s, and 5 GX-FANMODULE-Ds: -Power Entry Module (PEM) -G42 FIPS Input/Output (I/O) Module -Fan Controller Card (FCC) Module -Fan Modules (XMM4 and sled fan) -G42 Management Control Module (XMM4) -Coherent Module ICE6 (CHM6) Transponder Module -Blank Circuit Packs -Tributary Optical Module (TOM); 400G and 100G variants (a field-replaceable, pluggable module that converts the client optical signals to and from a serial electrical signal)[1]	**Software:** Converged OS (COS) version 6.2.10 **Multi-Processor System:** XMM4 - Intel Atom C3558 (Denverton) CHM6 – NXPLS1012ASE7KKB (ARM Cortex A53) **ASIC**: Atlantic ASIC **Power Supply:** Power Entry Modules (PEMs)—reside in the PEM module slots (labeled as PEM 1, PEM 2, PEM 3, and PEM 4) located at the rear of the chassis (in the upper 1 RU section); up to four AC or four DC PEMs are installed to provide 12V DC power supply throughout the GX G42 **Interfaces:** 10Gbps data communication network (DCN) Ethernet RJ-45 interface. Supports 100M, 1G, and 10G Ethernet port speeds and provides a port for debug and remote management. One 10Gbps auxiliary (AUX) Ethernet interface. Supports 100M, 1G, and 10G port speeds. Two 10Gbps nodal control and timing (NCT) Ethernet interfaces supporting 100M, 1G and 10G port speeds

[1] Optical network communication is not in the scope of this evaluation

Hardware Model	Specifications
	(used for multiple chassis and thus not supported or tested in the evaluated configuration) One 1000Mbps auxiliary Ethernet interface supporting 10M, 100M and 1G port speeds. One 1000MBps craft Ethernet interface supporting 10M, 100M and 1G port speeds One console RS-232 serial port interface One Universal Serial Bus (USB) port interface[2]

The evaluator performed full testing on the one TOE platform and microarchitecture.

- GX G42 3 RU chassis (XMM4 - Intel Atom C3558 (Denverton)

1.1.2 CAVP Equivalence

The TOE is a multi-processor system that includes multiple cryptographic libraries for cryptographic services associated with the following protocols: IPsec, IKEv2, SSH and TLSv1.2.

- The XMM4 module contains the following cryptographic libraries which provide the cryptographic services in the evaluated configuration:
  - Intel OpenSSL, Strongswan and SNMP Crypto Libraries (Openssl with libSNMP and libIKE) Version 6.2 firmware. This OpenSSL library is used for IKEv2 key generation and negotiation as well as SSH and TLS cryptographic services including secret negotiation, authentication, key exchange, encryption/decryption, message authentication, hashing and DRBG.
  - Intel Kernel IPsec Crypto Library (Kernel IPsec, LUKS, secureboot Signature verification) Version 4.19.274 firmware. This Kernel IPsec library is used for IPsec encryption/decryption, message authentication and hashing.

The evaluated configuration requires that the TOE be configured in FIPS mode to ensure that the CAVP tested algorithms are used. The following functions have been CAVP certified, with the exception of FFC schemes using safe prime groups which are verified using a known good implementation.

SFR	Algorithm	NIST/ISO Standard	Certificate #	
			Intel Kernel IPsec Crypto LIbrary firmware 4.19.274	Intel Openssl with libSNMP, libIKE firmware version 6.2
FCS_CKM.1	RSA KeyGen (2048, 3072, 4096 bits)	FIPS PUB 186-4		A4958
	ECDSA KeyGen/KeyVer	FIPS PUB 186-4		A4958

[2] USB ports are disabled in FIPS mode

SFR	Algorithm	NIST/ISO Standard	Certificate #	
			Intel Kernel IPsec Crypto LIbrary firmware 4.19.274	Intel Openssl with libSNMP, libIKE firmware version 6.2
	P-256, P-384 and P-521			
	FFC Safe Prime Groups (DH-14/15/16/17/18/ 19/20/21 ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192)	NIST SP 800-56A RFC 3526 RFC 7919	Tested with known good implementation as part of the test assurance activities for FCS_SSHS_EXT.1.7 test 2, FCS_TLSS_EXT.1.3 test 1 and test 2, and FCS_IPSEC_EXT.1.11.	
FCS_CKM.2	KAS ECC	NIST SP 800-56A		A4958
	FFC Safe Prime Groups (DH-14/15/16/17/18/ 19/20/21 ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192)	NIST SP 800-56A RFC 3526 RFC 7919	Tested with known good implementation as part of the test assurance activities for FCS_SSHS_EXT.1.7 test 2, FCS_TLSS_EXT.1.3 test 1 and test 2, and FCS_IPSEC_EXT.1.11.	
FCS_COP.1/ DataEncryption	AES CBC (128, 192 and 256 bits)	CBC as specified in ISO 10116, GCM as specified in ISO 19772		A4958
	AES GCM (128, 192 and 256 bits)	GCM as specified in ISO 19772	A4956	A4958
	AES CTR (128 and 256 bits)	CTR as specified in ISO 10116		A4958
FCS_COP.1/SigGen	RSA SigGen/SigVer (2048 bits, 3072, 4096 bits)	FIPS PUB 186-4		A4958
	ECDSA SigGen/SigVer (P-256, P-384, P-521)	FIPS PUB 186-4		A4958
FCS_COP.1/Hash	SHA-1/256/384/512 (digest sizes 160, 256, 384, 512 bits)	ISO/IEC 10118-3:2004	A4956	A4958
FCS_COP.1/ KeyedHash	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (digest sizes 160, 256, 384, 512 bits, respectively)	ISO/IEC 9797-2:2011, Section 7 'MAC Algorithm 2'	A4956	A4958
FCS_RBG_EXT.1	CTR_DRBG (AES) with platform based noise source (256 bits)	ISO/ICE 18031:2011		A4958

The evaluation team performed full testing on the single TOE platform and microarchitecture (GX G42 3 RU chassis (XMM4 - Intel Atom C3558 (Denverton)). All microarchitectures have been addressed as part of evaluation testing and CAVP testing, therefore, no equivalency argument is needed.

1.2 References

The following evidence was used to complete the Assurance Activities:

- Infinera GX G42 Optical Network Platform running Converged OS (COS) 6.2.10 Security Target, Version 1.1, 01/15/2025 (**ST**)
- Infinera GX-G42 Optical Network Platform Release 6.2.10 Hardening Guide, Revision V002, January 2025 (**Admin Guide**)

2. Protection Profile SFR Assurance Activities

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

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 (Security Audit) of the ST states that all cryptographic keys are assigned an administrator specified name upon generation or import. This unique key name is included in the audit logs for the administrator actions of generating/import of, changing, or deleting of cryptographic keys.

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

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

Section “Annex: Audit Generation” in the **Admin Guide** provides an embedded spreadsheet identifying all of the required audit events consistent with the ST along with a sample of each record for each SFR where applicable.

From a review of the ST, the Guidance and through testing the evaluator also determined that the guidance contains all of the administrative actions and their associated audit events that are relevant to the PP and to use of the TOE. These administrative actions are consistent with the security requirements implemented in the TOE and were found to have appropriate management capabilities identified in the guidance documentation.

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

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

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

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

2.1.2 User identity association (NDcPP22e:FAU_GEN.2)

2.1.2.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See NDcPP22e:FAU_GEN.1.

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

See NDcPP22e:FAU_GEN.1.

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

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

This activity is accomplished in conjunction with the testing of FAU_GEN.1.1.

The TOE is not distributed.

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

2.1.3.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

their generated audit data to other components it contains a mapping between the transmitting and storing TOE components.

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

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

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

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

Section 6.1 (Security Audit) of the ST states the TOE is a standalone TOE that is able to transmit audit logs to an external syslog server over a secure IPSec channel. The TOE transfers audit logs to a syslog server in real-time. The TOE simultaneously stores audit logs locally and transmits the same logs remotely. If the IPsec connection fails or the remote syslog server is otherwise unavailable, the administrator may recover messages from local storage. When the connection is re-established, only new audits will be sent.

When the local audit storage is full, the TOE will overwrite the oldest audit records first using a FIFO (first in, first out) replacement mode performed via rotation of log files. The TOE retains 10 log files which are rotated when they exceed 30 MB. The log files are stored in a filesystem of 8.4GB with typically 3.0GB of available space. The TOE will issue a warning message when there is only 10% storage space remaining. The TOE will not allow the configuration of a smaller rotation size or lower number of log files to be retained.

An authorized administrator must log into the TOE in order to view the local audit records. There are no interfaces via which the administrator can clear/delete or otherwise modify the contents of the local audit logs.

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

The evaluator shall also examine the guidance documentation to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit

event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and 'cleared' periodically by sending the data to the audit server.

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

Section “FAU_STG_EXT.1” in the **Admin Guide** states that the TOE maintains an internal log as well as connects to an external system for log reporting. The interface to external systems is described in section **5.13** of the in this document. Log messages are generated by system functions and sent to all log recording points (local as well as remote) simultaneously. Each log message has timestamp and a message ID. If a connection to a remote log server is not available (e.g. due to lack of configuration, or a network issue), only an authorized administrator can view the local audit records and there are no interfaces via which the administrator can clear/delete or otherwise modify the contents of the local audit logs.

When the local audit storage is full, the TOE will overwrite the oldest audit records first using a FIFO (First in, First out) replacement mode performed via rotation of log files. The TOE retains 10 log files which are rotated when they exceed 30 MB. The log files are stored in a filesystem of 8.4GB with typically 3.0GB of available space. The TOE will issue a warning message when there is only 10% storage space remaining. The TOE will not allow configuration of a smaller rotation size or lower the number of log files to be retained.

Section “Configure Remote Audit Logging” in the **Admin Guide** provides the commands for configuring a remote syslog server and configuring an SPD to protect the audit log transmission over an IPsec SA.

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

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

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

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

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

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

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

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

Test 1: The evaluator initiated a packect capture and then established a successful IPsec connection between the TOE and the syslog server utilizing Rsyslogd 8.16.0. The evaluator then issued several commands followed by a retrieval of the local audit log. The evaluator confirmed that all of the events that appeared in the local audit log were also transferred to the syslog server and the packet capture showed that the transmission of audit events to the syslog server was encrypted using IPsec.

Test 2: The option “overwrite previous audit records” is selected in FAU_STG_EXT.1.3. The evaluator issued commands to fill up the audit log buffer and then viewed the log and saved the output. The evaluator then issued more commands. The evaluator observed that the new audit records were generated and verified that the oldest audit records had been overwritten.

Test 3: Not applicable. The TOE does not claim support for FAU_STG_EXT.2/LocSpace.

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

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.2.1.1 NDcPP22e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The TOE supports key generation using RSA schemes (2048, 3072, 4096 bits) that meet FIPS PUB 186-4, Appendix B.3 and ECC schemes (P-256, P-384, P-521) that meet FIPS PUB 186-4, Appendix B.4 for the following:

- Generating key pairs for certificate signing requests (IPsec, TLS).
- Generates ECDHE keys for key exchange/establishment (SSH, TLS)
- Signature generation and verification for authentication (IPsec, TLS, SSH)

The TOE implements FFC schemes using 'safe-prime' groups that meet NIST SP 800-56A Revision 3 for the following:

- Generates DH keys for FFC key establishment (TLS)
- Generates keys for Diffie-Hellman groups for key exchange according to both RFC 3526 and RFC 5114 (IPsec, 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 key generation scheme(s) and key size(s) for all cryptographic protocols defined in the Security Target.

Section "Generate host SSH keys" in the **Admin Guide** provides the command for generating RSA 2048, 3072 and 4096 bit keys and ECDSA p-256, p-384 and -p-521 keys for use in SSH. Section "Add/remove SSH key exchange algorithms" provides the commands for configuring the SSH server key exchange algorithms, while Section "Add/remove SSH key-authentication algorithms" provides the commands for specifying the SSH host key and user public key for client authentication.

Section "Generate TOE Asymmetric Key Pair & issue Certificate Signing Request (CSR)" in the **Admin Guide** provides the command for generating TLS and IPsec certificate signing requests using RSA key sizes 2048, 3072 and 4096 bits and ECDSA key sizes eccp256, eccp384 and eccp521.

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

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

Key Generation for FIPS PUB 186-4 RSA Schemes

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

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

a) Random Primes:

- Provable primes

- Probable primes

b) Primes with Conditions:

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

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

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

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

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

- Primes q and p shall both be provable primes

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

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

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

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

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

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

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

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

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

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

(TD0580 applied)

The evaluator performed a CAVP certificate analysis and review which demonstrated that tests identified in this assurance activity for FCS_CKM.1 (with the exception of FFC schemes using safe prime groups) were successfully performed. Refer to the CAVP certificates identified in Section 1.1.2 of this AAR. Testing for FFC schemes using safe prime groups was performed as part of testing for FCS_CKM.2.1 using a known good implementation. Please refer to NDCPP22E:FCS_CKM.2.1 below.

2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.2.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be:

Scheme | SFR | Service

RSA | FCS_TLSS_EXT.1 | Administration

ECDH | FCS_SSHC_EXT.1 | Audit Server

ECDH | FCS_IPSEC_EXT.1 | Authentication Server

The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

(TD0580 applied)

Section 6.2 (Cryptographic Support) of the ST contains a table mapping the key establishment schemes to each service and the corresponding SFR. The supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.

Scheme	SFR	Services
ECC (P-256/384/521) DH-19/20/21	FCS_IPSEC_EXT.1	Syslog over IPsec RADIUS over IPsec TACACS+ over IPsec NTP over IPsec
ECC (P-256/384/521)	FCS_SSHS_EXT.1	SSH Remote Administration
	FCS_TLSS_EXT.1	HTTPS/TLS Remote Administration
FFC schemes using safe prime groups (DH-14/15/16/17/18)	FCS_IPSEC_EXT.1	IKEv2 key exchange (Syslog, RADIUS, TACACS+ and NTP over IPsec)
FFC schemes using safe prime groups (ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192)	FCS_TLSS_EXT.1	HTTPS/TLS Remote Administration
FFC schemes using safe prime groups (DH group 14)	FCS_SSHS_EXT.1	SSH Remote Administration

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

See FCS_CKM.1 where this activity has been performed.

Component Testing Assurance Activities: Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

RSA-based key establishment

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

FFC Schemes using 'safe-prime' groups

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

(TD0580 applied)

The evaluator performed a CAVP certificate analysis and review which demonstrated that tests identified in this assurance activity for FCS_CKM.2 (with the exception of FFC schemes using safe prime groups) were successfully performed. Refer to the KAS ECC certificate identified in Section 1.1.2.

For FFC schemes using safe prime groups (DH group 14/15/16/17/18/19/20/21, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192) key generation and key exchange, Gossamer tested using known good implementations: OpenSSH, Strongswan leveraging OpenSSL, and OpenSSL s_client to ensure correctness of the TOE's use of the safe prime groups in SSH, IPsec and TLS for key exchange. This testing was performed as part of the test assurance activities for FCS_SSHS_EXT.1, FCS_TLSS_EXT.1 and FCS_IPSEC_EXT.1.

2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.3.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

Where the TSS identifies keys that are stored in a non-plaintext form, the evaluator shall check that the TSS identifies the encryption method and the key-encrypting-key used, and that the key-encrypting-key is either itself stored in an encrypted form or that it is destroyed by a method included under FCS_CKM.4.

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

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE is designed to overwrite secret and private keys when they are no longer required by the TOE. Overwriting the keys is accomplished by overwriting the secret or private key with zeroes. This section includes a table which presents the crypto security parameters (CSPs), secret keys, and private keys provided by the TOE. The table also identifies where each CSP is stored and when each CSP or key is cleared.

In the event of an unexpected shutdown (e.g. crash or power loss), keys stored in volatile storage would be cleared from memory as a result of the loss of power/shutdown. For private keys in NVRAM (non-volatile storage), if an unexpected shutdown occurs during administrator-initiated zeroization of a private key, the administrator should run the command again when the TOE is back in its operational state to ensure no residual portions of the private keys remain.

Component Guidance Assurance Activities: A TOE may be subject to situations that could prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and any other supporting information used). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command [Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table)] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

Section "FCS_CKM.4" in the **Admin Guide** states that the GX G42 TOE provides management facilities to request, create, validate, use and destroy keys. These facilities operate on storage of the system database, X.509 certificates, TLS session keys, SSH authentication keys, and shared keys (e.g. for RADIUS and TACACS sessions). The destruction of individual keys as well as system zeroization is supported. The destruction process invokes internal operations that guarantee the destruction request has completed before the operation response is provided. Zeroization is done by invoking the command: fips zeroize. Zeroization affects all of the keys listed above, zeroizing the keys. It also reboots the system.

Component Testing Assurance Activities: None Defined

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

2.2.4.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports AES GCM (128, 192 and 256 bits) for data encryption/decryption in IPSec. The TOE supports AES GCM (128 and 256 bits) and AES CTR (128 and 256 bits) for data encryption/decryption in SSH. The TOE supports AES_128_CBC, AES_256_CBC, AES_128_GCM and AES_256_ GCM for data encryption/decryption in TLS.

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

Section “Add/remove SSH encryption ciphers available for negotiation” in the **Admin Guide** provides the commands for configuring the supported encryption algorithms used in SSH: aes128-ctr, aes256-ctr, aes128-gcm-at-openssh-com and aes256-gcm-at-openssh-com.

Section “Configure IPsec Instance and Peer association” in the **Admin Guide** provides the command for configuring the supported encryption algorithms in the IPsec proposal: AES-GCM-128, AES-GCM-192 and AES-GCM-256.

Section “Configure TLS based applications” includes sub-sections “Configure TLS version” with the command for configuring the TLS 1.2 version and “Configure TLS version cipher precedence” with the command for configuring the supported TLS cipher suites consistent with the ST. The supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS ciphersuites.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

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

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

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

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

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

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

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of keys and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall

have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

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

PT = IV

else:

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

PT = CT[i-1]

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

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

AES-GCM Test

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

128 bit and 256 bit keys

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

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

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

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

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

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

AES-CTR Known Answer Tests

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

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

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

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

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

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

AES-CTR Multi-Block Message Test

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

AES-CTR Monte-Carlo Test

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

Input: PT, Key

for i = 1 to 1000:

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

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

There is no need to test the decryption engine.

The evaluator performed a CAVP certificate analysis and review which demonstrated that tests identified in this assurance activity for FCS_COP.1/DataEncryption were successfully performed. Refer to the CAVP certificates identified in Section 1.1.2 of this AAR.

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

2.2.5.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Section 6.2 (Cryptographic Support) states that the TOE supports SHA-1/256/384/512 (digest sizes 160, 256, 384, and 512 bits) for cryptographic hashing. Hash functions are used in RSA/ECDSA digital signature generation/verification as well as in the key exchange algorithms for IPsec, TLS and SSH.

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

Section “Add/remove SSH message authentication codes (MACs) available” in the **Admin Guide** provides the commands for configuring the supported MAC algorithms: hmac-sha2-256, hmac-sha2-512.

Section “Configure encryption proposal” in the **Admin Guide** provides the command for configuring the hmac algorithms in the ike sa encryption proposal. These algorithms are: hmac-sha2-256-128, hmac-sha2-384-192, hmac-sha2-512-256, or hmac-sha1-160.

Section “Provide an encryption proposal for the SA” in the **Admin Guide** provides the command for configuring the hmac algorithms in the ipsec sa encryption proposal. These algorithms are: hmac-sha2-256-128, hmac-sha2-384-192, hmac-sha2-512-256, or hmac-sha1-160

Section “Configure TLS based applications” includes sub-sections “Configure TLS version” with the command for configuring the TLS 1.2 version and “Configure TLS version cipher precedence” with the command for configuring the supported TLS cipher suites consistent with the ST. The supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS ciphersuites.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

The evaluator performed a CAVP certificate analysis and review which demonstrated that tests identified in this assurance activity for FCS_COP.1/Hash were successfully performed. Refer to the CAVP certificates identified in Section 1.1.2 of this AAR.

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

2.2.6.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports the following keyed hash algorithms:

	Key Length (bits)	Block size (bits)	Output MAC length (bits)	Protocol(s) used in
HMAC-SHA-1	160	512	160	IPsec/IKEv2
HMAC-SHA2-256	256	512	256	IPsec/IKEv2, TLS, SSH
HMAC-SHA2-384	384	1024	384	IPsec/IKEv2, TLS
HMAC-SHA2-512	512	1024	512	IPsec/IKEv2, SSH

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

Section “Add/remove SSH message authentication codes (MACs) available” in the **Admin Guide** provides the commands for configuring the supported MAC algorithms: hmac-sha2-256, hmac-sha2-512.

Section “Configure encryption proposal” in the **Admin Guide** provides the command for configuring the hmac algorithms in the ike sa encryption proposal. These algorithms are: hmac-sha2-256-128, hmac-sha2-384-192, hmac-sha2-512-256, or hmac-sha1-160.

Section “Provide an encryption proposal for the SA” in the **Admin Guide** provides the command for configuring the hmac algorithms in the ipsec sa encryption proposal. These algorithms are: hmac-sha2-256-128, hmac-sha2-384-192, hmac-sha2-512-256, or hmac-sha1-160

Section “Configure TLS based applications” includes sub-sections “Configure TLS version” with the command for configuring the TLS 1.2 version and “Configure TLS version cipher precedence” with the command for configuring the supported TLS cipher suites consistent with the ST. The supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS ciphersuites.

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 evaluator performed a CAVP certificate analysis and review which demonstrated that tests identified in this assurance activity for FCS_COP.1/KeyedHash were successfully performed. Refer to the CAVP certificates identified in Section 1.1.2 of this AAR.

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

2.2.7.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 (Cryptographic Support) states that the TOE supports RSA (modulus 2048, 3072 and 4096) and ECDSA with elliptical curve size P-256, P-384 and P-521 for signature generation and verification.

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

Section “Generate host SSH keys” in the **Admin Guide** provides the commands for generating RSA 2048, 3072 and 4096-bit keys and ECDSA p256, p384 and p521 keys. RSA and ECDSA signature services using these algorithms are automatically configured when SSH is configured as instructed.

Section “Generate TOE Asymmetric Key Pair & issue Certificate Signing Request (CSR)” in the **Admin Guide** provides the command for generating TLS and IPsec certificate signing requests using RSA 2048, 3072 and 4096 bit keys and ECDSA p256, p384 and p521 keys. RSA and ECDSA signature services are automatically configured when TLS and IPsec are configured as instructed.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

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

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

Signature Verification Test

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

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

The evaluator performed a CAVP certificate analysis and review which demonstrated that tests identified in this assurance activity for FCS_COP.1/SigGen were successfully performed. Refer to the CAVP certificates identified in Section 1.1.2 of this AAR.

2.2.8 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.2.8.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.8.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.8.3 NDcPP22e:FCS_HTTPS_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE complies to RFC 2818 by implementing HTTP over TLS channels. The TOE provides an HTTPS/TLS interface for remote administration via the Web UI and RESTCONF. These interfaces do not require or support client certificate authentication. The TOE follows common practice identified in RFC 2818 by not using the standard HTTP port 80. In accordance with RFC2818 the TOE is prepared to receive an incomplete close from the client and the TOE attempts to initiate an exchange of closure alerts with the client before closing 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.

Section "Configure TLS based applications" and sub-sections in the **Admin Guide** provide the commands for configuring the TOE as an HTTP/TLS server including commands for configuring TLS v1.2, commands to configure the claimed ciphersuites and for associating the TOE identity certificate with the TOE TLS applications.

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.

The TOE is never an HTTPS client. When acting as an HTTPS server, the TOE does not authenticate its peer on the WebUI interface using x509 certificates. Users must authenticate via a user login process on these interfaces.

Testing of the TLS server protocol for the WebUI was demonstrated in FCS_TLSS_EXT.1.

2.2.9 IPsec Protocol (NDcPP22e:FCS_IPSEC_EXT.1)

2.2.9.1 NDcPP22e:FCS_IPSEC_EXT.1.1

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

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE implements IPsec to provide an authenticated and encrypted channel between the TOE and the following remote IT entities: a Syslog server, a RADIUS server, a TACACS+ server and an NTP server.

The TOE can be configured with SPD rules that will either Bypass (send/receive the packets without IPsec protection), Protect (secure the packet via IPsec), or Discard (drop the packets) based on IP addresses and port numbers. The TOE can also be configured with a default discard SPD rule that will discard all traffic that does not match any other SPD rule. The configuration of the TOE's SPD includes settings for the IP protocol number, local IP address and port number, remote IP address and port number, the IP processing choices or rule actions (ie. protect, bypass or discard), the mode (tunnel or transport) and the IPsec ESP cryptographic algorithms.

The SPD priority is configurable for each SPD rule. The SPD is used if the traffic selector matches the traffic and the priority is the lowest (ie. 1 has priority over 9999) in the system. On ingress traffic, interface ACLs are processed prior to IPsec. On egress traffic, interface ACLs are processed after IPsec.

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

Section "Configure Security Policy Definitions" in the **Admin Guide** states that Security Policy Definitions identify the traffic to be placed in a security association, and the encryption suite to be used. The encryption suite used for an SPD must be of the same strength or stronger than the suite used for the IPSEC Peer. The Security Policy Definitions (SPD) priority is configurable for each SPD rule. The SPD is used if the traffic selector matches the traffic, and the priority is the lowest (ie. 1 has priority over 9999) in the system. On ingress traffic, interface ACLs are processed prior to IPsec. On egress traffic, interface ACLs are processed after IPsec.

Section "Create SPD Entry" in the **Admin Guide** provides the commands for creating an SPD entry including specifying the peer, traffic mode (tunnel or transport) and the action: protect, bypass or drop. Section "Add Traffic Selectors" provides instructions for identifying traffic to send in the SA, while section "Provide an encryption proposal for the SA" provides the commands for specifying the encryption to be used in the ipsec sa proposal. This proposal covers the encryption used for the IPSEC SA. A proposal is not needed for bypass or discard SPDs.

Section "Configure IPSEC Bypass Security Policy Definitions for TLS and SSH-based protocols" in the **Admin Guide** provides instructions for configuring bypass on the TOE in an SPD entry. Section "Configure Default Security Policy Definition" in the **Admin Guide** describes how to configure an SPD to discard traffic that is not encrypted, either by IPSEC or by the application itself.

Section "Configure Remote Audit Logging" in the **Admin Guide** provides specific instructions for configuring the SPD to be entered to pass the Syslog communications over an IPSEC SA.

Section "Configure NTPv4 Synchronization" in the **Admin Guide** provides specific instructions for configuring the SPD to be entered to pass the NTP communications over an IPSEC SA.

Section “Configure SPDs so that RADIUS/TACACS+ traffic is carried by IPSEC” and sub-sections in the **Admin Guide** provide specific instructions for configuring the SPD to be entered to pass the RADIUS and TACACS communications over an IPSEC SA.

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

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

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

The evaluator tested the TOE to ensure that it appropriately applies its SPDs to traffic. The TOE applies SPDs in the order of defined priority, 1 being the highest and 9999 being the lowest priority. For each action, PROTECT, DISCARD, AND BYPASS, the evaluator configured an SPD rule with a priority and specified traffic selectors.

Test 1: The evaluator sent a series of packets to the TOE. The evaluator sent packets to the IPsec tunnel interface that matched the rules to be encrypted (Protect) and observed that the traffic was encrypted by IPsec. The evaluator then sent the packets that matched a Bypass rule to allow unencrypted traffic and observed that the traffic was sent in plaintext (Bypass). The evaluator also sent packets that matched rules to be Discarded and observed that the packets were dropped (Discard). Additionally, the evaluator configured three SPD rules, each with a different remote subnet for bypass, discard and protect. The evaluator then sent packets from each of the configured remote subnet addresses and an additional unspecified address. As expected, the appropriate packets were protected, allowed through (bypass), or discarded. The packet from the unspecified address was discarded by the default discard rule. The use of the default discard SPD rule, shows that the TOE correctly discards traffic not specified by the other configured SPD rules.

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

access list on the interface so the specific discard rule was first. The evaluator sent packets from the defined host and viewed that they were denied.

Next the evaluator confirmed that ordering was enforced regardless of if a rule was subset of another one. First the evaluator created a bypass rule using a subnet. Then a discard rule with a specific IP address in that subnet was placed with a lower priority than the bypass rule. The packets sent were accepted as expected. The evaluator then created the same set of rules this time with the discard rule having a higher priority than the bypass rule and confirmed that the packets were dropped as expected.

2.2.9.2 NDcPP22e:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

This was performed as part of NDcPP22e:FCS_IPSEC_EXT.1.1 test 1.

2.2.9.3 NDcPP22e:FCS_IPSEC_EXT.1.3

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE implements IPsec in accordance with RFC 4301 and supports both transport and tunnel mode.

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

Section “Create SPD Entry” in the **Admin Guide** provides the commands for creating an SPD and specifying the traffic mode as either tunnel mode or transport mode.

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

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

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

Test 1 and Test 2: For this test, the evaluator alternately configured a test peer to require only tunnel mode or only transport mode. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connections were successful with both tunnel mode and transport mode.

2.2.9.4 NDcPP22e:FCS_IPSEC_EXT.1.4

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE uses the Encapsulating Security Payload (ESP) protocol to provide authentication and encryption supporting the following algorithms: AES-GCM-128, AES-GCM-192 and AES-GCM-256 together with a Secure Hash Algorithm (SHA)-based HMAC (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512) for integrity. These SHA-based HMAC algorithms conform to the algorithms specified in FCS_COP.1/KeyedHash.

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

Section “Provide an encryption proposal for the SA” in the **Admin Guide** provides the commands for configuring the ipsec sa proposal and selecting the claimed encryption and integrity algorithms to be used. These algorithms are consistent with those specified in the ST.

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

The evaluator configured the TOE to use ESP algorithms AES-GCM-128, AES-GCM-192 and AES-GCM-256, and verified via logs and packet captures that the connection was successfully established for each algorithm.

2.2.9.5 NDcPP22e:FCS_IPSEC_EXT.1.5

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

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

Section 6.2 (Cryptographic Support) in the ST states that TOE implements IKEv2.

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

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

Section "Configure IPSEC Instance and Peer association" and sub-sections in the **Admin Guide** provides instructions for configuring the IPsec peer entity with commands for adding an "ikev2" peer. NAT traversal is not selected.

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

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

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

Test 1: Not applicable. The TOE does not support IKEv1.

Test 2: Not applicable. The TOE does not support NAT traversal.

2.2.9.6 NDcPP22e:FCS_IPSEC_EXT.1.6

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

Section 6.2 (Cryptographic Support) in the ST states that TOE implements IKEv2. The TOE implements IKEv2 and supports the following encryption algorithms for setting up the IKE SA as part of IKEv2 negotiation with a remote peer: AES-GCM-128, AES-GCM-192 and AES-GCM-256. The IKEv2 protocol also supports the following integrity algorithms: HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512.

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

Section "Configure encryption proposal" in the **Admin Guide** provides the commands for configuring the ike sa proposal and selecting the claimed encryption and integrity algorithms to be used. These algorithms are consistent with those specified in the ST.

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

The evaluator configured the IKEv2 proposal (AES-GCM-128, AES-GCM-192 and AES-CBC-256) on the TOE. The evaluator then confirmed that the TOE could establish a session with each algorithm and that the tunnel successfully established with the selected algorithm.

2.2.9.7 NDcPP22e:FCS_IPSEC_EXT.1.7

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE supports IKEv2 session establishment and configuration of session lifetimes for both IKEv2 IKE_SAs and IKEv2 Child_SAs. The time values for IKEv2 IKE_SA lifetime SAs can be configured up to 24 hours and for IKEv2 Child_ SAs up to 8 hours. The IKEv2 Child_SA lifetime can also be configured based on number of bytes with configurable values being from between: 1048576 and 4294967295 bytes. The selection of IKEv2 corresponds with the selection made in FCS_IPSEC_EXT.1.5.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g. configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (TD0800 applied)

Section "FCS_IPSEC_EXT.1.7" in the **Admin Guide** provides instructions and the command for configuring the IKEv2 re-key threshold. It states that the TOE supports automatic IKE re-keying based on the amount of time the session has been operating. The specific limits that trigger re-keying may be lower in reality so the re-keying process can

be completed in-time. For example, the IKE re-keying time limit may need to be set to 23hr45 minutes so it is completed within 24 hours.

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

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

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

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the lifetime Phase 1 SA on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that a new Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation. (TD0800 applied)

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

Test 2: For this test, the evaluator configured the TOE to have a 24 hour IKE and 8 hour ESP limits and the test server was configured to have 25 hour IKE and 9 hour ESP limits. The evaluator established an IPsec connection between the TOE being tested and the test server and then waited for over 24 hours before terminating the test. The evaluator observed through logs and packet captures that the connection was successful and that the TOE rekeyed well before the configured time limits were reached ensuring that the keys were renegotiated successfully and there was no data loss during the rekey.

2.2.9.8 NDcPP22e:FCS_IPSEC_EXT.1.8

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE supports IKEv2 session establishment and configuration of session lifetimes for both IKEv2 IKE_SAs and IKEv2 Child_SAs. The time values for IKEv2 IKE_SA

lifetime SAs can be configured up to 24 hours and for IKEv2 Child_ SAs up to 8 hours. The IKEv2 Child_SA lifetime can also be configured based on number of bytes with configurable values being from between: 1048576 and 4294967295 bytes. The selection of IKEv2 corresponds with the selection made in FCS_IPSEC_EXT.1.5.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g. configure a time value of 7h 45min to ensure the actual rekey is performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (TD0800 applied)

Section “FCS_IPSEC_EXT.1.8” in the **Admin Guide** provides instructions and the commands for configuring the IPsec Child SA re-key threshold based on time and bytes. It states that the TOE supports automatic IPSEC Child SA re-keying based on 1) the amount of time and 2) the amount of data. As with the IPSEC SA rekeying, the specific limits that trigger re-keying may be lower in reality so the re-keying process can be completed in time.

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

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

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

Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once a new Phase

2 SA is negotiated when or before 8 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 2 negotiation. (TD0800 applied)

Test 1: The evaluator configured a max lifetime in bytes on the TOE and established a connection with a test peer. The Phase 2 SA/IKEv2 Child SA timed out after the packet size was exceeded and the TOE rekeyed the negotiation successfully.

Test 2: This test was performed as part of the FCS_IPSEC_EXT.1.7 test 2 where the evaluator observed that the TOE rekeyed before the configured time limits for the Phase 1 SA/IKEv2 SA and the Phase 2 SA/IKEv2 Child SA.

2.2.9.9 NDcPP22e:FCS_IPSEC_EXT.1.9

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

Section 6.2 (Cryptographic Support) in the ST states that the TSF generates the secret value 'x' used in the IKEv2 Diffie-Hellman key exchange ('x' in gx mod p) using the NIST approved DRBG specified in FCS_RBG_EXT.1 and having possible lengths of at least

- 224 bits (for DH Group 14),
- 256 bits (for DH Group 15),
- 320 bits (for DH Group 16,
- 384 bits (for DH Group 17),
- 512 bits (for DH Group 18),
- 256 bits (for DH Group 19),
- 384 bits (for DH Group 20) and
- 521 bits (for DH Group 21)

The TOE supports the following PRF hash functions: PRF_HMAC_SHA1, PRF_HMAC_SHA256, PRF_HMAC_SHA384 and PRF_HMAC-SHA512 and generates nonces used in the IKEv2 exchanges of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash. The nonce is likewise generated using the TOE's ISO/IEC 18031:2011 AES-256 CTR DRBG.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.10 NDcPP22e:FCS_IPSEC_EXT.1.10

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

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE supports the following PRF hash functions: PRF_HMAC_SHA1, PRF_HMAC_SHA256, PRF_HMAC_SHA384 and PRF_HMAC-SHA512 and generates nonces used in the IKEv2 exchanges of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash. The nonce is likewise generated using the TOE's ISO/IEC 18031:2011 AES-256 CTR DRBG.

Guidance Assurance Activities: None Defined

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

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

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

This activity is performed as part of the TSS Assurance Activity above where the evaluator verified that the TSS indicates that the random number generated and the length of the nonces meets the requirements.

2.2.9.11 NDcPP22e:FCS_IPSEC_EXT.1.11

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

Section 6.2 (Cryptographic Support) in the ST states that the IKEv2 protocol implements DH Group(s) 14 (2048-bit MODP), 15 (3072-bit MODP), 16 (4096-bit MODP), 17 (6144-bit MODP), 18 (8192-bit MODP, 19 (256-bit Random ECP), 20 (384-bit Random ECP) and 21 (521-bit Random ECP). In the IKEv2 IKE_SA and IKEv2 CHILD_SA exchanges, the TOE and peer will agree on the best DH group both can support. When the TOE initiates the IKE negotiation, the DH group is sent in order according to the peer's configuration. When the TOE receives an IKE proposal, it will select the first match and the negotiation will fail if there is no match.

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

Section “Configure encryption proposal” in the **Admin Guide** provides the commands for configuring the ike sa proposal and selecting the claimed encryption and integrity algorithms and the DH groups to be used. The selectable dh groups are consistent with those specified in the ST.

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

The evaluator made a successful IPsec connection to an IPsec peer using the claimed DH groups (14, 15, 16, 17, 18, 19, 20, 21). The evaluator was able to capture the DH group using a packet capture to ensure the correct DH group was used.

2.2.9.12 NDcPP22e:FCS_IPSEC_EXT.1.12

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

Section 6.2 (Cryptographic Support) in the ST states that the resulting potential strength of the symmetric key will be 128 or 256 bits of security depending on the algorithms negotiated between the two IPsec peers. As part of this negotiation, the TOE verifies that the negotiated IKEv2 Child SA symmetric algorithm key strength is at most as large as the negotiated IKEv2 IKE SA key strength as configured on the TOE and peer via an explicit check. The TOE checks the IKEv2 IKE SA strength against the IKEv2 Child SA strength and will reject attempts to configure a Child SA with a higher strength.

Guidance Assurance Activities: None Defined

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

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

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

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

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

The TOE only supports IKEv2.

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

Test 2: The evaluator attempted to establish a connection with a test server configured with an ESP algorithm with greater strength than the IKEv2 SA algorithm and observed that the attempt failed.

Test 3: The evaluator attempted to establish a connection first with an unsupported IKE encryption algorithm, then with an unsupported IKE integrity hash and finally with an unsupported ESP encryption algorithm. In each case, the connection attempt failed.

Test 4: This test was performed as part of Test 3 above where an IPsec connection is attempted using an unsupported ESP encryption algorithm and fails.

2.2.9.13 NDcPP22e:FCS_IPSEC_EXT.1.13

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

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

Section 6.2 (Cryptographic Support) in the ST states that the TOE supports the following authentication mechanisms to authenticate remote IKE peers.

- Pre-shared Key (PSK) - using PSK as the authentication mechanism to authenticate remote IKE peers. The TOE does not generate pre-shared keys, but uses pre-shared keys configured by the authorized administrator. The PSK can be entered as a string with a range of 8 to 128 characters in length (ASCII format) or 8 to 128 bytes in length (Hexadecimal format).
- X.509 certificates (RSA and ECDSA) - using X.509 digital certificates as the authentication mechanism to authenticate remote IKE peers. The local entity certificates used by the local IKE protocol daemon as it's identity, must be configurable by the user.

For peer authentication using RSA and ECDSA certificates, the TOE validates the presented identifier provided supporting the following fields and types: SAN: IP address, SAN: Fully Qualified Domain Name (FQDN) and Distinguished Name (DN).

This is consistent with FCS_COP.1/SigGen which specifies the RSA and ECDSA digital signature algorithms for RSA and ECDSA schemes.

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

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

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

Section “Configure System X.509 Certificate” in the **Admin Guide** provides instructions for loading CA certificates to the TOE’s trust store. Section “Establish the TOE’s Identity Certificate” and sub-sections provide instructions on how to generate RSA and ECDSA keypairs, generate a certificate request and manually import the signed certificate to the TOE.

Section “Alternate: IPSEC peer using pre-shared key authentication” in the **Admin Guide** provides instructions for configuring pre-shared keys instead of X.509 certificates. Pre-shared key is specified as the authentication scheme. The pre-shared key is a string that is 8 to 128 characters (ascii) or 8 to 128 bytes (hex) in length.

Section “FCS_IPSEC_EXT.1.13” in the **Admin Guide** provides an example of configuring IPsec for communication with a peer using pre-shared key authentication.

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

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

2.2.9.14 NDcPP22e:FCS_IPSEC_EXT.1.14

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

Section 6.2 (Cryptographic Support) in the ST states that for peer authentication using RSA and ECDSA certificates, the TOE validates the presented identifier provided supporting the following fields and types: SAN: IPv4 address

(the TOE does not support IPv6), SAN: Fully Qualified Domain Name (FQDN) and Distinguished Name (DN). The SAN identifier type is explicitly configured as part of the reference identifier by an authorized administrator.

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

Section "FCS_IPSEC_EXT.1.14" in the **Admin Guide** states that the TOE supports a number of types of identifiers for IPSEC authentication. The supported identifier types are: ipv4 address, fqdn, and X.509 DN. This is configured per peer, in the peer-identity-type attribute.

Section "IPSEC peer using X.509 certificate authentication" in the **Admin Guide** provides the command for configuring the ikev2 peer entity which includes the peer identity type.

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

The evaluator shall perform the following tests:

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

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

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

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

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

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

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

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

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

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

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

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

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

Test 2: For the first part of this test, the evaluator alternately configured a test peer to use an authentication certificate with the correct SAN: FQDN and one with the correct SAN:IPv4 address (IPv6 is not supported). The evaluator then attempted to establish an IPsec connection between the TOE and the test peer and confirmed that the connection was successful. CN identifiers are not supported by the TOE so the second part of the test is not applicable. These tests were iterated for both IPsec RSA and IPsec ECDSA.

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

Test 4: For this test, the evaluator configured the TOE to look for the supported SAN reference identifier (SAN: FQDN, SAN:IP address). The evaluator then configured the test peer to use a certificate that would present an incorrect SAN reference identifier and a correct CN reference identifier as CN checking is not prioritized over SAN (*CN is not supported*). As expected, the TOE rejected the attempt to establish an IPsec connection. These tests were iterated for both IPsec RSA and IPsec ECDSA.

Test 5: For this test, the evaluator configured a test peer to send an authentication certificate with an authorized DN and verified that the connection succeeded. This test was iterated for both IPsec RSA and IPsec ECDSA.

Test 6: Part A: For this test, the evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate but with a DN containing a duplicate CN. The evaluator confirmed that the connection was rejected. Part B: The evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate in which the Organization field of the DN has a trailing null character (52) appended. The evaluator confirmed that the connection was rejected. These tests were iterated for both IPsec RSA and IPsec ECDSA.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.10 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1)

2.2.10.1 NDcPP22e:FCS_NTP_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS to ensure it identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained. The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp.

Section 6.2 (Cryptographic Support) in the ST states that the TOE can synchronize its time with an external NTP server using the NTPv4 protocol. The TOE must establish a successful IPsec connection between itself and the NTP server. The IPsec session ensures that data is encrypted and that the two peers are authenticated before data is transmitted, thus ensuring the integrity of the time has been maintained.

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

Section “Configure NTPv4 Synchronization” in the **Admin Guide** provides instructions for configuring NTPv4 using the “set system ntp ntp-enabled true ntp-server-***<NTPADDR>*** admin-state unlock” command. This command is used to specify one or multiple NTP servers. This section also includes the commands for configuring one or more SPDs to pass the NTP communications from each configured NTP server over an IPsec SA.

Section “Configure System X.509 Certificates” in the **Admin Guide** provides instructions for configuring the TOE to use X.509 certificates for establishing identity for communications using IPsec.

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

The evaluator set the time on the NTP Server ahead by 1 hour and enabled the NTP service. At the same time the evaluator configured the TOE to receive NTP time updates from the NTP server. After a short time, the TOE updated its time to match that of the NTP server. Inspection of the network traffic confirmed that the TOE supports NTPv4.

2.2.10.2 NDcPP22e:FCS_NTP_EXT.1.2

TSS Assurance Activities: None Defined

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

Assurance Activity Note:

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

Section “Configure System X.509 Certificates” in the **Admin Guide** provides instructions for configuring the TOE to use X.509 certificates for establishing identity for communications using IPsec.

Section “Configure IPsec” in the **Admin Guide** provides instructions for configuring IPsec certificate-based authentication or IPsec pre-shared key authentication, the IPsec peer entity and the ike sa encryption proposal. Section “Configure Security Policy Definition” provides instructions for identifying traffic to be placed in a security association (SA) and defining the ipsec sa encryption proposal.

Section “Configure NTPv4 Synchronization” in the **Admin Guide** provides instructions for configuring NTP servers including the commands for configuring one or more SPDs to pass the NTP communications from each configured NTP server over an IPsec SA.

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

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

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

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

The TOE uses IPSEC to provide trusted communication between itself and a NTP time source.

The evaluator set the time on the NTP Server ahead by 2 hours and enabled the NTP service. At the same time the evaluator configured the TOE to receive NTP time updates from the NTP server. After a short time, the TOE updated its time to match that of the NTP server. Inspection of the network traffic confirmed that the TOE is able to utilize IPsec to protect NTP communications.

2.2.10.3 NDcPP22e:FCS_NTP_EXT.1.3

TSS Assurance Activities: None Defined

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

Section “Configure NTPv4 Synchronization” in the **Admin Guide** states that the GX G42 TOE only supports directed NTPv4 sessions. It will not react to receiving a broadcast NTP or multicast NTP packet. No configuration is needed for this behavior.

Testing Assurance Activities: The evaluator shall configure NTP server(s) to support periodic time updates to broadcast and multicast addresses. The evaluator shall confirm the TOE is configured to not accept broadcast and multicast NTP packets that would result in the timestamp being updated. The evaluator shall check that the time stamp is not updated after receipt of the broadcast and multicast packets.

The evaluator began with the NTP server not broadcasting and the TOE with NTP disabled. The evaluator then set the time on the NTP server ahead 2 hours and started the NTP service to send broadcast and multicast messages. At the same time, the TOE was set to enable NTP but no NTP server was specified. The evaluator found that the TOE was not responding to the broadcast or multicast messages. After waiting a short time, NTP configuration on the TOE was set to specify an NTP server. With the TOE now specifying an NTP server, the TOE ignored the broadcast and multicast time update, and attempted to update time using traditional authenticated updates.

2.2.10.4 NDcPP22e:FCS_NTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Test 2: (The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers).

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

(TD0528 applied)

The evaluator set the time on the NTP server ahead 3 hours using the command 'date --set="+3 hour"', configured the TOE with 3 valid NTP connections, and then monitored the current time and sync status with the configured NTP servers on the TOE. During this monitoring, the evaluator captured network traffic. The evaluator verified that the TOE attempted to update time using traditional authenticated updates and syncs with one of the three valid NTP servers.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.11 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.11.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports an AES-CTR 256-bit ISO/IEC 18031:2011 DRBG. The DRBG is seeded by an entropy source that accumulates entropy from a platform-based noise source.

The DRBG is seeded with a minimum of 256 bits of entropy, which is at least equal to the greatest security strength of the keys and hashes that it will generate.

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

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

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

Section “Place the TOE into FIPS operation mode” and section “Enable FIPS mode” in the **Admin Guide** provide instructions for configuring the TOE into FIPS mode. When running in FIPS mode, only approved algorithms that have been CAVP tested are used, including the TOE's CAVP AES-256-CTR DRBG. There is no further configuration required for configuring RNG functionality.

Section “FCS_RBG_EXT.1” in the **Admin Guide** describes the Entropy Alarm which is designed to identify the entropy low condition in the system that arises due to a dip in the entropy level below a pre-defined threshold (low amount of available Entropy bits). Guidance on clearing this alarm and when to contact technical support is also provided.

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

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

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

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

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

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

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

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

The evaluator performed a CAVP certificate analysis and review which demonstrated that tests identified in this assurance activity for FCS_RBG_EXT.1 were successfully performed. Refer to the CAVP certificates identified in Section 1.1.2 of this AAR.

2.2.12 SSH Server Protocol (NDcPP22e:FCS_SSHS_EXT.1)

2.2.12.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 NDcPP22e:FCS_SSHS_EXT.1.2

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

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

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE implements both password-based and public key-based user authentication methods in SSHv2. The TOE supports SSH public-key user authentication using rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384 and ecdsa-sha2-nistp521. These algorithms are consistent with the selections in FCS_COP.1/SigGen. When RSA or ECDSA authentication is used, the TOE checks the presented public key against its authorized keys database and verifies the user's possession of a private key by negotiating a secure channel using the public key associated with that private key.

Guidance Assurance Activities: None Defined

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

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

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

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

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

(TD0631 applied)

Test 1: The TOE supports rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521 for client public key authentication. The evaluator attempted to connect to the TOE from an SSH client using a user authenticated by each of the supported public keys and observed that the logins were successful.

Test 2: The evaluator next demonstrated an unsuccessful login using an unrecognized public key.

Test 3: The evaluator attempted to connect to the TOE using a SSH client alternately using the correct and incorrect password. The evaluator found that only the correct password would yield a successful SSH session. A successful SSH protected NETCONF connection was demonstrated in FIA_UIA_EXT.1.

Test 4: This was performed as part of Test 3 above where the evaluator attempted to login with an incorrect password and observed that the attempt failed.

2.2.12.3 NDcPP22e:FCS_SSHS_EXT.1.3

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE checks the packet length of an incoming packet. If a packet greater than 262130 bytes is sent to the TOE, the TOE drops the packet and terminates the SSH session.

Guidance Assurance Activities: None Defined

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

The evaluator created and sent a packet to the TOE that was larger than the maximum packet size. The TOE rejected the packet and the connection was closed.

2.2.12.4 NDcPP22e:FCS_SSHS_EXT.1.4

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports the following algorithms for Encryption: aes128-ctr, aes256-ctr, aes128-gcm@openssh.com and aes256-gcm@openssh.com. These algorithms are identical to those listed for this component.

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

Section “Add/remove SSH encryption ciphers available for negotiation” in the **Admin Guide** provides instructions for configuring the SSH encryption algorithms on the TOE using the ‘ssh-ciphers’ attribute in the system security-policies. Modifying the list is done using the set command: “set system security security-policies ssh-ciphers aes256-gcm-at-openssh-com, aes256-ctr, aes128-gcm-at-openssh-com,aes128-ctr”.

Testing Assurance Activities: The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall start session establishment for an SSH connection from a remote client (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool

or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

The evaluator connected to the TOE using an SSH client alternately using the claimed encryption algorithms. The evaluator confirmed that the supported algorithms resulted in successful connections.

2.2.12.5 NDcPP22e:FCS_SSHS_EXT.1.5

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports the following host public key algorithms: rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384 and ecdsa-sha2-nistp521. These algorithms are identical to those listed for this component.

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

Section “Add/remove SSH key-authentication algorithms” in the **Admin Guide** provides instructions for configuring the host public key algorithms using the following command: “set system security security-policies ssh-host-key-algorithms ecdsa-sha2-nistp521, ecdsa-sha2-nistp384, ecdsa-sha2-nistp256, rsa-sha2-512, rsa-sha2-256”.

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

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

Has effectively been moved to FCS_SSHS_EXT.1.2.

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

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

(TD0631 applied)

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

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

2.2.12.6 NDcPP22e:FCS_SSHS_EXT.1.6

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports the following algorithms for data integrity: hmac-sha2-256, hmac-sha2-512 and implicit (when aes*-gcm@openssh.com is negotiated as the encryption algorithm, the MAC algorithm field is ignored and GCM is implicitly used as the MAC.). These algorithms are identical to those listed for this component.

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

Section “Add/remove SSH message authentication codes (MACs) available” in the **Admin Guide** provides the commands for configuring the supported MAC algorithms: hmac-sha2-256, hmac-sha2-512. When aes*-gcm@openssh.com is negotiated as the encryption algorithm, the MAC algorithm field is ignored and GCM is implicitly used as the MAC.

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

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

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

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

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

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

2.2.12.7 NDcPP22e:FCS_SSHS_EXT.1.7

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports the following key exchange algorithms: diffie-hellman-group14-sha256, ecdh-sha2-nistp256, ecdh-sha2-nistp384 and ecdh-sha2-nistp521. These algorithms are identical to those listed for this component.

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

Section "Add/remove SSH key exchange algorithms" in the **Admin Guide** provides the following instructions to configure the SSH key exchange algorithms using the following command: "set system security security-policies ssh-key-exchanges ecdh-sha2-nistp521, ecdh-sha2-nistp384, ecdh-sha2-nistp256, diffie-hellman-group14-sha256."

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

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

Test 1: The evaluator attempted to establish an SSH connection with the TOE using an SSH client configured to only allow diffie-hellman-group1-sha1 key exchange. The connection attempt failed.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using each allowed key exchange method: diffie-hellman-group14-sha1, ecdh-sha2-nistp256 and ecdh-sha2-nistp384. The connections succeeded.

2.2.12.8 NDcPP22e:FCS_SSHS_EXT.1.8

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

a) Both thresholds are checked by the TOE.

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

Section 6.2 (Cryptographic Support) of the ST states that the TOE automatically initiates a rekey of the SSH session keys at either 1 hour or 1 gigabyte of traffic. Both thresholds are checked. Rekeying is performed upon reaching whichever threshold is met first.

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

Section "FCS_SSHS_EXT.1.8" in the **Admin Guide** states that the TOE supports SSH re-keying, but does not support configuration of the re-keying thresholds. The thresholds are statically defined at 1hr and 1Gb of traffic exchanged. Re-keying will occur when either of these events are satisfied, and the timers/counters will be reset to 0 with the use of the new key.

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

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

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

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

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

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

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

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

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

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

The evaluator connected to the TOE using an SSH client. The evaluator then repeatedly issued a command on the TOE's CLI that outputs a large amount of text. This process causes the TOE to transmit more data than it receives from the SSH client, causing the TOE transmit data rekey threshold to be hit first. The SSH client's rekey threshold was disabled, and the evaluator continued issuing this command until the SSH client's debug logs indicated that it received a rekey message from the TOE. The test stops immediately and closes the connection once the rekey is observed. Next the evaluator attempted to connect to the TOE using a SSH client waiting and verified that a rekey happened at the 1 hour threshold.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.13 TLS Server Protocol Without Mutual Authentication (NDcPP22e:FCS_TLSS_EXT.1)

2.2.13.1 NDcPP22e: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.

Section 6.2 (Cryptographic Support) of the ST states that the TOE supports TLSv1.2 sessions for remote administration via the Web UI. The TOE supports the TLS ciphersuites identified in the FCS_TLSS_EXT.1 requirement in Section 5.1.2.13 of the ST. The TOE does not support mutual authentication and does not require client authentication.

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 may have to be restricted to meet the requirements).

Section “Configure TLS based applications” includes sub-sections “Configure TLS version” with the command for configuring the TLS 1.2 version and “Configure TLS version cipher precedence” with the command for configuring

the supported TLS cipher suites consistent with the ST. The cipher suites identified in this section are consistent with those specified in the ST. The supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS ciphersuites.

Testing Assurance Activities: 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).

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

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

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

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

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

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

These tests were iterated for TLS RSA and TLS ECDSA.

Test 1: The evaluator attempted to connect to the TOE using each of the ciphersuites identified by the Protection Profile. A packet capture was obtained for each connection attempt. The evaluator verified that successful connections were established with all claimed ciphersuites.

Test 2: The evaluator attempted to connect to the TOE using the TLS_NULL_WITH_NULL_NULL ciphersuite and observed that the connection failed. The evaluator then attempted to connect to the TOE using all ciphers except those defined in the PP and observed that the connections were all rejected.

Test 3: Part A: The evaluator attempted a connection to the TOE where the evaluator modified a byte in the Finished handshake message and verified that the TOE rejected the connection attempt after receiving the modified Finished message and that the TOE sent no application data. Part B: The evaluator established a TLS connection from a test server to the TOE, while capturing packets associated with that connection. 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).

2.2.13.2 NDcPP22e:FCS_TLSS_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS contains a description of how the TOE technically prevents the use of old SSL and TLS versions.

Section 6.2 (Cryptographic Support) of the ST states that an authorized administrator can initiate inbound TLSv1.2 connections using the Web UI for remote administration of the TOE. Any session where the client offers the following in the client hello: SSL 2.0, SSL 3.0, TLS 1.0 and TLS 1.1 will be rejected by the TOE's TLS server. If the remote TLS client does not support TLSv1.2 the TLS connections will fail and the administrators will not establish a HTTPS web-based session with 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 "Configure TLS version" provides the command for configuring only TLSv1.2.

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

The evaluator alternately attempted to connect to the TOE using a control case (TLS not specified), SSL2.0, SSL3.0, TLSv1.0, TLSv1.1 and TLSv1.2 and confirmed that only the TLS1.2 connection was successful while all other TLS versions resulted in rejected connections.

2.2.13.3 NDcPP22e:FCS_TLSS_EXT.1.3

TSS Assurance Activities: If using ECDHE and/or DHE ciphers, 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. For example, if the TOE supports TLS_DHE_RSA_WITH_AES_128_CBC_SHA cipher and Diffie-Hellman parameters with size 2048 bits, then list Diffie-Hellman Group 14.

Section 6.2 (Cryptographic Support) of the ST states that the TOE uses RSA or ECDSA X.509 certificates for authentication. The TOE performs key establishment using Diffie-Hellman groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 or ECDHE curves: secp256r1, secp384r1, secp521r1 depending on the chosen cipher suite. The key agreement parameters of the server key exchange message are specified in the RFC 5246 (section 7.4.3) for TLSv1.2.

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

Section "Configure TLS based applications" includes sub-sections "Configure TLS version" with the command for configuring the TLS 1.2 version and "Configure TLS version cipher precedence" with the command for configuring the supported TLS cipher suites consistent with the ST. The cipher suites identified in this section are consistent with those specified in the ST. The supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS ciphersuites. The TOE performs key establishment using Diffie-Hellman groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 or ECDHE curves: secp256r1, secp384r1, secp521r1 depending on the chosen cipher suite. The key agreement parameters of the server key exchange message are specified in the RFC 5246 (section 7.4.3) for TLSv1.2.

Testing Assurance Activities: Test 1: [conditional] If ECDHE ciphersuites are supported:

a) The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single supported elliptic curve specified in the Elliptic Curves Extension. The Evaluator shall verify (though a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange message and successfully establishes the connection.

b) The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single unsupported elliptic curve (e.g. secp192r1 (0x13)) specified in RFC4492, chap. 5.1.1. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

Test 2: [conditional] If DHE ciphersuites are supported, 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 a 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 message are the ones configured Diffie-Hellman parameter size(s).

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.

Test 1: Part A: 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. The evaluator observed that the claimed elliptic curve key exchange methods resulted in successful connections. Part B: The evaluator attempted a connection using an unsupported elliptic curve and verified that the connection failed.

Test 2: 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. The evaluator observed that the claimed DH group key exchange methods resulted in successful connections.

Test 3: Not applicable. The TOE does not support RSA key establishment ciphersuites.

2.2.13.4 NDcPP22e: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) and/or if session resumption based on session tickets is supported (RFC 5077).

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.

Section 6.2 (Cryptographic Support) of the ST states that the TOE does not support session resumption or session tickets. It is explicitly disabled in the TOE.

Guidance Assurance Activities: None Defined

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).

Test 1 [conditional]: If the TOE does not support session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2) or session tickets according to RFC5077, the evaluator shall perform the following test:

a) The client sends a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

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

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps: Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

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

e) 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).

f) 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.

Test 2 [conditional]: If the TOE supports session resumption using session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2), 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):

a) 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).

b) 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.

Test 3 [conditional]: If the TOE supports session tickets according to RFC5077, 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):

a) 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 a ServerHello with an empty SessionTicket extension, NewSessionTicket, ChangeCipherSpec and Finished messages (as seen in figure 2 of RFC 5077).

b) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator will 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.

Test 1: The evaluator initiated a connection attempt in which the Client Hello has a zero-length Session ID and a zero-length Session Ticket extension and verified that the TLS server did not send a NewSessionTicket message and contained a zero length session id. The evaluator concluded that the TOE does not support session resumption using session ids and the TOE does not support the use of session tickets.

Test 2: Not applicable. The TOE does not support session resumption using session ids.

Test 3: Not applicable. The TOE does not support the use of session tickets.

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 (NDcPP22e:FIA_AFL.1)

2.3.1.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 NDcPP22e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 (Identification and authentication) of the ST states that for all methods of remote administration- SSH (CLI, NETCONF) and HTTPS/TLS (Web UI, RESTCONF) - the TOE tracks each unsuccessful authentication attempt for each user account. The number of unsuccessful authentication attempts can be configured between 1 and 255 with the default being 5. The administrator defined lockout time period can be configured from 0 to 1440 minutes. If the counter reaches the administrator specified number of failed login attempts, then the account will be locked out and prevented from gaining access even with a valid password until an administrator specified time period has expired. The account is thereby unlocked upon expiration of the lockout period. The counter is reset to zero upon successful authentication. Failed login attempts are tracked across all remote interfaces with a single counter. Local serial console port access to the CLI is not subject to the lockout and ensures that administrator access is always available

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

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

Section “Configure user security policies” in the **Admin Guide** provides instructions for configuring the TOE to suspend user accounts that have a specified number of invalid login attempts for a specified duration. Local serial console port access to the CLI is not subject to the lockout and ensures that administrator access is always available.

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.

Test 1 & 2: The evaluator performed lockout testing on the TOE device with two different values to verify that the configuration works correctly. The first case was performed with an invalid attempt threshold of 3 attempts and a lockout time of 1 minute. The second case was performed with an invalid attempt threshold of 5 attempts and a lockout time of 3 minutes. In each case, the evaluator observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator also observed that the account was unlocked after the configured time period had passed.

2.3.2 Password Management (NDcPP22e:FIA_PMG_EXT.1)

2.3.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the TSS lists the supported special character(s) for the composition of administrator passwords.

The evaluator shall check the TSS to ensure that the minimum_password_length parameter is configurable by a Security Administrator.

The evaluator shall check that the TSS lists the range of values supported for the minimum_password_length parameter. The listed range shall include the value of 15.

(TD0792 applied)

Section 6.3 (Identification and authentication) of the ST states that the TOE allows administrators to configure a minimum password length for users that is between 8 and 200 characters. Along with upper and lower case letters and numbers, the TOE allows the following special characters: [*'!', '@', '#', '$', '%', '^', '&', '*', '(', ')' [and additional special characters: <sp>* ~ _ - + = ` | { } " [ ] : ; < > , . ? / ' \ *]*]. The TOE provides obscured feedback to

the administrative user while the authentication is in progress at the local console. The TOE can be configured with either a local database or a remote RADIUS or TACACS+ database for authenticating users.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it:

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

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

Section "Configure user security policies" in the **Admin Guide** provides instructions for configuring the TOE to require complex passwords and prevent the user from reusing previously used passwords.

Section "FMT_SMR.2" in the **Admin Guide** provides instructions on configuring a TOE administrator and identifies the characters that may be used in passwords and the command for configuring a minimum password length.

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

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

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

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

2.3.3 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.3.3.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

There are no preparatory steps needed to ensure authentication data is not revealed. Section "FMT_SMR.2" in the **Admin Guide** states that the TOE provides obscured feedback to the administrative user while the authentication is in progress at the local console. The TOE can be configured with either a local database or a remote RADIUS or TACACS+ database for authenticating users.

Component Testing Assurance Activities: The evaluator shall perform the following test for each method of local login allowed:

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

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

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

2.3.4.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See NDcPP22e:FIA_UIA_EXT.1.

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

See NDcPP22e:FIA_UIA_EXT.1.

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

See NDcPP22e:FIA_UIA_EXT.1.

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

2.3.5.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

For distributed TOEs, the evaluator shall examine the TSS to determine that it describes for each TOE component which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration. For each TOE component that does not support

authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2 the TSS shall describe any unauthenticated services/services that are supported by the component.

Section 6.3 (Identification and authentication) of the ST states that the TOE requires all administrators to be successfully identified and authenticated before allowing any TSF mediated actions to be performed except for acknowledging the pre-login warning banner. The TOE allows administrators to login to the TOE locally via a directly connected console serial port or to login remotely via SSH (CLI, NETCONF) and via HTTPS/TLS (Web UI, RESTCONF). SSH supports both password and public key authentication, while HTTPS/TLS supports password authentication only.

The process for password authentication is the same for administrative access whether administration is occurring via a directly connected console cable or remotely via SSH or TLS. Once a potential administrative user attempts to access the TOE through either a directly connected console or remotely through SSH or HTTPS/TLS, the TOE prompts the user for a user name and password. Only after the administrative user presents the correct authentication credentials will access to the TOE administrative functionality be granted. The TOE either grants administrative access (if the combination of username and password is correct) or indicates that the login was unsuccessful. No access is allowed to the administrative functionality of the TOE until an administrator is successfully identified and authenticated.

For SSH remote administration the TOE can also be configured to authenticate using a public key mechanism (RSA or ECDSA). Note that the TOE does not support configuration of both password and public key authentication at the same time. When SSH user public key authentication is configured, the TOE ensures and verifies that the SSH client's presented public key matches one that is stored within the TOE's SSH server's authorized keys file. If a user attempts public key-based authentication and it succeeds, the authentication process is completed and the user is granted access, otherwise, the TOE will indicate that the public key login was unsuccessful and the connection will be rejected.

The TOE also supports authentication using external authentication servers. The TOE communicates with a Remote Authentication Dial-In User Service (RADIUS) server and a Terminal Access Controller Access Control Server (TACACS+) for authentication of configured users. The communication with the RADIUS and TACACS+ servers is protected via IPsec.

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.

Section "FTP_TRP.1.3/Admin and FIA_UIA_EXT.1" in the **Admin Guide** states that all remote administration of the TOE takes place over a secure communication path between the remote administrator and the TOE using either an SSHv2 client (to access the CLI and the NETCONF API over a secure SSHv2 session, or a Web browser to access the Web UI and the RESTCONF API over a secure HTTPS/TLS connection.

Connection to the serial interface should be via a serial terminal operating at 15200 bps, 8bits, no parity, 1 stop bit. For SSH, an SSH session is established to the TOE's IP address, TCP port 22. For NETCONF, an SSH session is established to the TOE's IP address, TCP port 830. For RESTCONF, an HTTPS session is established to the TOE's IP address, TCP port 8181.

The SSH and NETCONF sessions will negotiate the cryptographic algorithms used for their session from the algorithms configured on the client and the TOE. For RESTCONF, the negotiation is done based on the TLS1.2 algorithms configured on the client and the TOE.

Section "Connect the Craft terminal to the TOE and login" in the **Admin Guide** describes the steps for initial configuration of the TOE including the first user login and password establishing the default security administrator on the TOE. Section "Enable FIPS mode" places the TOE into FIPS mode after a reboot. Configuring NIAP compliance mode is described in Section "Check NIAP compliance, set expected NIAP compliance mode, retrieve NIAP compliance alarm state and resolve compliance issues".

Section "Configure TOE security policies" and sub-sections in the **Admin Guide** provides instructions for configuring the login banner, generating host SSH keys and configuring remote administration using SSH. This includes SSH client authentication using either password or SSH public key.

Section "Configure user security policies" in the **Admin Guide** provides instructions on requiring complex passwords, preventing reuse of passwords and configuring authentication lockout for too many invalid login attempts.

Section "Configure System X.509 Certificate" in the **Admin Guide** provides instructions for loading CA certificates to the TOE's trust store. Section "Generate TOE Asymmetric Key Pair & issue Certificate Signing Request (CSR)" in the **Admin Guide** provides the command for generating TLS and IPsec certificate signing requests using RSA key sizes 2048, 3072 and 4096 bits and ECDSA key sizes eccp256, eccp384 and eccp521. The CSR block should be captured manually and sent to a Certificate Authority (CA) for processing. Section "Load the Signed Certificate for the TOE" provides further instructions on how to import the signed TOE certificate received from the Certificate Authority.

Section "Configure TLS application identity" in the **Admin Guide** provides the command for associating the TOE local X.509 certificate to be used as an identity certificate for TLS.

Section "WebGUI" in the **Admin Guide** provides the command for associating the TOE local X.509 certificate with the Web GUI.

Section "Configure RADIUS/TACACS+ servers" in the **Admin Guide** provides the steps for configuring the use of a RADIUS or TACACS+ server and setting the authentication method/policy for the ordering of local and remote servers. Section "Configure SPDs so that RADIUS/TACACS+ traffic is carried by IPSEC" describes how to configure SPDs to ensure that RADIUS/TACACS+ authentication is protected with IPSec.

Section "FMT_SMR.2" in the **Admin Guide** provides instructions on configuring a TOE administrator and identifies the characters that may be used in passwords and the command for configuring a minimum password length.

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

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

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

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

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

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

- TOE Command Line Interface using Local
- TOE Command Line Interface using RADIUS
- TOE Command Line Interface using Tacacs+
- SSH (CLI) to TOE Using Local passwords
- SSH (CLI) to TOE using public/private key pairs
- SSH (CLI) connection using RADIUS
- SSH (CLI) connection using Tacacs+
- SSH (NETCONF) to TOE using local passwords
- HTTPS/TLS (Web UI) to TOE using local passwords
- HTTPS/TLS (Web UI) to TOE using RADIUS
- HTTPS/TLS (Web UI) to TOE using Tacacs+
- HTTPS/TLS (RESTCONF) to TOE using local passwords

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 that the TOE displayed a banner to the user before login and that there were no other services available nor any configuration options offered to administrators to control services available prior to authentication.

Test 3 - This test was performed as part of test 1 & 2. Using each interface, the evaluator found that, prior to login,

no functions were available to the administrator with the exception of acknowledging the banner.

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

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

2.3.6.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Test 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed

by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

(TD0527 12/2020 update applied)

Test 1a: Successful IPsec connections with valid certificate chains were demonstrated in the Control Tests in FIA_X509_EXT.1.1/Rev-t2 and t3.

Test 1b: For this test, the evaluator configured the TOE to not have the trusted root CA used by the test peer to anchor all of its certificates. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and observed that the connection failed without a valid certificate chain.

Test 2: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid and 2) that is expired and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the TOE to the test server and observed that the connection succeeded only if there were no expired certificates.

Test 3: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that is revoked, and 3) issued by an intermediate CA that is revoked. In each case, the evaluator then attempted to connect the TOE to the test server and confirmed that the connection only succeeded if there were no revoked certificates. This test was executed using CRL. OCSP is not supported by the TOE.

Test 4: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) issued by an intermediate CA referring to a CRL revocation server where the signer lacks cRLSign, and 3) issued by an intermediate CA whose issuer CA refers to a CRL revocation server where the signer lacks cRLSign. In each case, the evaluator then attempted to connect the TOE to the test server and confirmed that the connection only succeeded if all retrieved CRLs were signed using certificates with cRLSign.

Test 5: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator attempted to connect the TOE to the test server and verified that the connection only succeeded if the certificate was not modified/corrupted.

Test 6: This test was performed as part of Test 5 above where the evaluator configured the test peer to send an authentication certificate with the last byte in the certificate signature modified and confirmed that the connection failed.

Test 7: This test was performed as part of Test 5 above where the evaluator configured the test peer to send an authentication certificate with one byte in the certificate public key modified and confirmed that the connection failed.

Test 8: Part 1&2: For this test, the evaluator alternately configured a test peer to send an authentication certificate issued by an intermediate CA with a valid elliptic curve and an explicitly defined elliptic curve. In each

case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and verified that the connection was rejected in the second case. Part 3: The evaluator established a subordinate CA with a named curve and one with an explicit curve and confirmed that the connection with the second case was rejected.

2.3.6.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

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

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

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

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 (Identification and authentication) in the ST states that during configuration of the trusted channel (either TLS or IPsec), an administrator must specify the certificate to be used with the trusted channel. The trust chain is a shared set of CAs for both TLS and IPsec operations. The TOE does not support mutual authentication for TLS.

The TOE will perform certificate verification when it receives a certificate from a peer or when a certificate is installed. For IPsec, the TOE checks the validity of the certificates it receives from its peers (such as the certificate's extended key usage, expiration, revocation, basic constraints, and reference identifiers) as part of the authentication step of the IPsec connections. The TOE performs these checks on the peer's certificate before moving up the chain to check each intermediate CA in the chain for revocation and basic constraints. The TOE will not authenticate using only a leaf certificate - full path resolution is always required. If a partial path is provided by the peer, the system will ask for the missing certificates to resolve the path to a known root. If the peer does not provide them then the connection is aborted. If the certificate is invalid, the TOE rejects the connection attempt.

The TOE supports the following Certificate revocation methods: Certificate Revocation List and CRL Distribution Point (CDP) and Online Certificate Status Protocol (OCSP). OCSP and CRL revocation methods can be simultaneously enabled. If both methods are configured, then OCSP is used by default to query the Certificate revocation status. CRL will be used if a definite revocation status is not determined by the OCSP method. The TOE checks the IPsec peer's certificate revocation during the authentication step of a IPsec connection. The TOE checks each level of the certificate chain sent by the peer. If any certificate in the chain is revoked, the TOE will reject the connection attempt.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore

claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

Section “Certificate Revocation” in the **Admin Guide** states that the TOE will perform certificate verification when it receives a certificate from a peer or when a certificate is installed. The TOE checks the validity of the certificates it receives from its peers (such as the certificate's extended key usage, expiration, revocation, basic constraints, and reference identifiers) as part of the authentication step for the IPsec connections. The TOE performs these checks on the peer's certificate before moving up the chain to check each intermediate CA. This is done because any certificate in the chain must be checked for revocation and basic constraints. The TOE will not authenticate using only a leaf certificate - full path resolution is always required. If a partial path is provided by the peer, the system will ask the peer for the missing certificates to resolve the path to a known root. If the peer does not provide them then the connection will be aborted. If the certificate is invalid, the TOE also rejects the connection attempt.

The TOE supports the following Certificate revocation methods: Certificate Revocation List, CRL Distribution Point (CDP) and Online Certificate Status Protocol (OCSP). OCSP and CRL revocation methods can be simultaneously enabled. If both methods are configured, then OCSP is used by default to query the Certificate revocation status. CRL will be used if a definite revocation status is not determined by the OCSP method. The TOE checks the IPsec peer's certificate revocation during the authentication step of a connection. The TOE checks each level of the certificate chain sent by the peer. If any certificate in the chain is revoked, the TOE will reject the connection attempt.

Component Testing Assurance Activities: None Defined

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

2.3.7.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.7.2 NDcPP22e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 (Identification and authentication) in the ST states that the TOE will perform certificate verification when it receives a certificate from a peer or when a certificate is installed. The TOE checks the IPsec peer's certificate revocation during the authentication step of an IPsec connection. The TOE checks each level of the certificate chain sent by the peer. If any certificate in the chain is revoked, the TOE will reject the connection attempt.

TOE supports the following Certificate revocation methods: Certificate Revocation List and CRL Distribution Point (CDP) and Online Certificate Status Protocol (OCSP). OCSP and CRL revocation methods can be simultaneously enabled. If both methods are configured, then OCSP is used by default to query the Certificate revocation status. CRL will be used if a definite revocation status is not determined by the OCSP method.

The TOE relies on certificate revocation checking by communicating with an external server specified in the certificate's AIA extension for OCSP or CDP extension for CRL. If the TOE cannot establish communications with the external server for OCSP revocation checks, then the TOE will not accept the certificate and will terminate the connection attempt with the peer and there will be no fall back to CRL-based revocation checking. If only CRL revocation checking is enabled and the TOE cannot establish communications with the external server, the TOE will not accept the certificate and will terminate the connection attempt.

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

Section "Configure System X.509 Certificate" in the **Admin Guide** provides instructions for loading CA certificates to the TOE's trust store. Section "Generate TOE Asymmetric Key Pair & issue Certificate Signing Request (CSR)" in the **Admin Guide** provides the command for generating TLS and IPsec certificate signing requests using RSA key sizes 2048, 3072 and 4096 bits and ECDSA key sizes eccp256, eccp384 and eccp521. The CSR block should be captured manually and sent to a Certificate Authority (CA) for processing. Section "Load the Signed Certificate for the TOE" provides further instructions on how to import the signed TOE certificate received from the Certificate Authority. The same name must be used for the csr-gen command and for the download local-certificate command so that the TOE may associate the asymmetric key pair with the received certificate.

Section “Certificate Revocation” in the **Admin Guide** states that if the TOE cannot establish communications with the external server for OCSP revocation checks, then the TOE will not accept the certificate and will terminate the connection attempt with the peer and there will be no fall back to CRL-based revocation checking. If only CRL revocation checking is enabled and the TOE cannot establish communications with the external server, the TOE will not accept the certificate and will terminate the connection attempt. In this case, the Administrator should troubleshoot and resolve the issue before proceeding.

Section “Dealing with revoked certificates” in the **Admin Guide** states that the Security Administrator should review all certificates, starting at the leaf and progressing to the root certificate. Each certificate should be checked (for CA certificates) for a set Basic Constraints CAFlag and (for leaf certificates) applicable Extended Key Usage. Each certificate should be checked for signature validity. The system will log why a certificate was not accepted. The Security Administrator will need to work with the peer system’s Security Administrator to resolve any issues.

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

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

This test was iterated for 2 variations: CRL and OCSP

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

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

2.3.8.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 NDcPP22e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Not applicable as 'device-specific information' is not selected in the ST.

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

Section "Generate TOE Asymmetric Key Pair & issue Certificate Signing Request (CSR)" in the **Admin Guide** provides the command for generating TLS and IPsec certificate signing requests using RSA key sizes 2048, 3072 and 4096 bits and ECDSA key sizes eccp256, eccp384 and eccp521. The command includes specifying the key algorithm and establishing the subject- name which includes specification of the Common Name (CN). The CSR block is then captured manually and sent to a Certificate Authority (CA) for processing.

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

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

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

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

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

2.4 SECURITY MANAGEMENT (FMT)

2.4.1 MANAGEMENT OF SECURITY FUNCTIONS BEHAVIOUR (NDCPP22E:FMT_MOF.1/MANUALUPDATE)

2.4.1.1 NDCPP22E:FMT_MOF.1.1/MANUALUPDATE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

There are no specific requirements for non-distributed TOEs.

The TOE is not a distributed TOE.

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

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

Section "FMT_MOF.1/ManualUpdate, FPT_TUD_EXT.1 and AGD_OPE.1" in the **Admin Guide** provides the steps and commands for performing a software update. Since the process involves rebooting before an upgrade can be completed, the TOE does not pass traffic during the update.

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

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

The set of functions available to administrators prior to login do not include TOE update as specified in NDcPP22e:FIA_UIA_EXT.1-t2. The successful update of the TOE by an authorized administrator was demonstrated in NDcPP22e:FPT_TUD_EXT.1-t1.

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

2.4.2.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 (Security management) in the ST states that the TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed. Prior to being granted access, a login warning banner is displayed. Only Security Administrators can manage TSF data. There are no security functions available through any interfaces prior to administrator login.

The term “Security Administrator” used in the ST refers to any user account assigned access privileges (see FMT_SMR.2 below) allowing them to perform all TOE security management functions.

The trust store is accessed when Security administrators import/remove and assign certificates to endpoints as described in the **Admin Guide**. Only TOE Security Administrators have the required access privileges to manage the trust store.

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 TSF data manipulating functions and the corresponding configuration information are identified or referenced throughout this AAR with the requirement to which they apply.

Section "FMT_SMR.2" in the **Admin Guide** provides instructions on configuring the TOE administrator with full access to all TOE security functions which includes importing X.509v3 certificates to the TOE's trust store and otherwise maintaining the trust store securely.

See NDcPP22e:FIA_X509_EXT.2.2 which identifies the sections in the Guide(s) that describe how the administrator role can configure and maintain the trust store including loading of CA certificates.

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

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

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

2.4.3.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 (Security management) in the ST states that no administrative functionality is available prior to administrative login. Only TOE Security Administrators can control (generate/import/delete) the following keys:

RSA and ECDSA key pairs to be used in the TLS, SSH and IPsec protocols. The **Admin Guide** provides instructions for generating these key pairs and for assigning keys to users for SSH user public key authentication.

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

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

Section "Generate host SSH keys" in the **Admin Guide** provides the command for generating RSA 2048, 3072 and 4096 bit keys and ECDSA p-256, p-384 and -p-521 keys for use in SSH. Section "Add/remove SSH key exchange algorithms" provides the commands for configuring the SSH server key exchange algorithms, while Section "Add/remove SSH key-authentication algorithms" and "SSH Client Public Key Authentication" provide the commands for specifying the SSH host key and user public key for client authentication.

Section "Generate TOE Asymmetric Key Pair & issue Certificate Signing Request (CSR)" in the **Admin Guide** provides the command for generating TLS and IPsec certificate signing requests using RSA key sizes 2048, 3072 and 4096 bits and ECDSA key sizes eccp256, eccp384 and eccp521.

Section "FCS_CKM.4" in the **Admin Guide** indicates that the TOE provides the ability to request, create, validate, use and destroy keys including X.509 certificates (using RSA and ECDSA keys), TLS authentication and session keys, SSH authentication and session keys and shared keys (ascii keys for RADIUS and TACACS sessions). The destruction of individual keys as well as system zeroization is supported. Zeroization is done by invoking the command: 'fips zeroize'.

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

The set of functions available to administrators prior to login do not include performing any cryptographic functions as specified by NDcPP22e:FIA_UIA_EXT.1-t2. The successful generation of a new CSR and private key and the subsequent import of a signed certificate by an authorized administrator is demonstrated in NDcPP22e:FIA_X509_EXT.3.

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

2.4.4.1 NDcPP22e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

(containing also requirements on Guidance Documentation and Tests)

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

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

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

Section 6.4 (Security management) in the ST states that all TOE security functions can be performed via the remote SSH interfaces (CLI, NETCONF) and remote HTTPS/TLS interfaces (Web UI, RESTCONF). All TOE security functions, with the exception of managing cryptographic keys, can also be performed via the local console.

The TOE provides the Security Administrator with capabilities to manage all security functions identified in this Security Target, including the following:

- 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 configure the list of TOE-provided services available before an entity is identified and authenticated, as specified in FIA_UIA_EXT.1,
- Ability to manage the cryptographic keys,
- Ability to configure the cryptographic functionality,
- Ability to configure the lifetime for IPsec SAs,
- Ability to set the time which is used for time-stamps;
- Ability to configure NTP,
- Ability to configure the reference identifier for the peer;
- Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors,
- Ability to import X509v3 certificates to the TOE's trust store,
- Ability to manage the trusted public keys database

Component Guidance Assurance Activities: See TSS Assurance Activities

See TSS Assurance Activities.

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

All of the management functions were demonstrated throughout the course of testing.

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

2.4.5.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.5.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.5.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Section 6.4 (Security management) in the ST states that the TOE maintains the security role of Security Administrator who can manage the TOE via both local and remotely accessible administrator interfaces.

The TOE defines access privileges to restrict user access to resources. Each access privilege allows a specific set of actions to be performed. One or more access privileges can be assigned to each user account.

In the evaluated configuration, the TOE Security Administrator role is a user account that is assigned the following access privilege levels:

- Security Administrator (SA)—allows the user to perform network element security management and administration related tasks.
- Network Administrator (NA)—allows the user to monitor the network element, manage equipment, turn-up network element, provision services, administer various network-related functions such as Auto-discovery and topology.
- Encryption Access (EA)—allows the user to perform data encryption procedures.

Thus, the term “Security Administrator” refers to any user assigned the access privileges identified above allowing them to perform all TOE security management functions. Management functions are exclusively restricted to Security Administrators.

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

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

Component Testing Assurance Activities: In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of

administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

The TOE is administered through either an HTTPS/TLS protected WebUI, an SSH protected CLI, or local console CLI. All interfaces were exercised throughout the course of testing. The interfaces were also exercised for the NDcPP22e:FIA_UIA_EXT.1 tests and the various FTA test cases.

2.5 Protection of the TSF (FPT)

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

2.5.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.5 (Protection of the TSF) in the ST states that passwords are stored encrypted in the TOE database using a PBKDF2 (Password-Based Key Derivation Function 2) as detailed in RFC 2898. Passwords are salted and hashed using PBKDF2 with hmac-sha512 and 100000 iterations. The TOE does not offer any functions that will disclose to any users a stored cryptographic key. The TOE provides no interfaces to read pre-shared, symmetric keys, private keys or passwords.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.2.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 (Cryptographic Support) of the ST includes a table which presents the crypto security parameters (CSPs), secret keys, and private keys provided by the TOE. The table also identifies where each CSP is stored and when each CSP or key is cleared. Section 6.5 (Protection of the TSF) in the ST states that the TOE does not offer any functions that will disclose to any users a stored cryptographic key. The TOE provides no interfaces to read pre-shared, symmetric keys, private keys or passwords.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.3.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.3.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.5 (Protection of the TSF) in the ST states that the TOE has an internal, real-time system clock that maintains time across reboots and power loss. The internal clock can be manually set or configured to sync with an external NTP source. The clock function provides date and time information which is used for the following security functions:

- To provide a timestamp for audit records
- To support timing elements of cryptographic functions (e.g. certificate validity checks to determine if a certificate is expired and certificate revocation checks)
- Monitoring and measuring local and remote administrative sessions for inactivity
- Unlocking of administrator accounts which have been locked as a result of authentication failure for a specified time period
- Determine when SSH and TLS session keys have expired and to initiate a rekey
- Determine when to renegotiate SAs for IPsec tunnels

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

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

Section “Configure Time” in the **Admin Guide** provides instructions for the administrator to manually set the time on the TOE using the ‘set-time’ command.

Refer to NDcPP22e:FCS_NTP_EXT.1.1 where the guidance assurance activities reference the sections in the Guide where instructions are provided for configuring NTP.

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

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

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

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

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

Test 1: The evaluator set the clock from the local console and then observed via the console display and the audit records that the time was set as intended.

Test 2: The evaluator followed the guidance documentation to enable NTP on the TOE and set an NTP server. The evaluator first displayed the current time on both the TOE and NTP server and then set the time on the NTP server ahead by 1 hour and started the NTP service. The evaluator monitored the time on the NTP server and the TOE and confirmed that the time updated as expected.

Test 3: Not applicable. The TOE does not obtain time from an underlying VS.

2.5.4 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.5.4.1 NDCPP22E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather

than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

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

Section 6.5 (Protection of the TSF) in the ST states that the TOE performs a suite of self-tests to verify the correct operation of the cryptographic module. Software images are cryptographically signed, and an image with an invalid signature will not be loaded by the TOE. If a self-test fails, the TOE goes into “Error” state and disables all access to cryptographic functions and Critical Security Parameters (CSPs). The management interfaces do not respond to any commands until the TOE is operational again. The system will be accessible only through the serial interface. The administrator will need to login to the local console and perform an on demand self-test. If this does not clear the Error state on the system, the administrator must contact Infinera Technical Support.

Self-tests can be performed on demand and periodically during normal operation by powering off and then powering on the TOE device or by executing the ‘fips self-test’ command. The full suite of self-tests is then executed.

The TOE performs self-tests at power-up to verify the integrity of the firmware images and the correct operation of the FIPS-approved algorithm implementations for AES, SHA, HMAC, ECDSA, RSA, DRBG, and KDFs.

The TOE executes both “known answer self-tests” and “pair-wise consistency tests” during power-up. The known answer tests involve inputting known data into each algorithm and comparing the outputs against expected results. For example, the TOE’s AES-CBC Encrypt KAT takes a known key, encrypts known plaintext using AES-CBC, and compares the result against known ciphertext. The KAT fails if the comparison results in the calculated ciphertext not matching the known ciphertext. For the Pair-Wise consistency test, the TOE takes a public/private key pair to calculate and verify a digital signature. If a pair-wise consistency test fails, the TOE shuts down the data output interface. If all self-tests pass, the TOE proceeds to normal operation.

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

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

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

Section “FPT_TST_EXT.1” in the **Admin Guide** states that the TOE performs self-tests at system startup. These tests include pair-wise consistency tests, KDF tests and software integrity test and Cryptographic known answer

tests. These tests are all performed as a part of the boot process and failures of any test are noted by an alarm being raised. The 'show alarm' command can be used to view TOE alarms.

If the TOE is in FIPS error, the TOE will reboot and will not re-establish its network connection. In this case, the only way to query TOE alarms or view logs is via the serial console using a locally authenticated account. To attempt clearing the fips-error condition, the administrator needs to manually attempt a self-test using the 'fips self-test' command. This will cause the TOE to reset and execute the self-test. When it is complete, the TOE will either be out of fips-error (and operational) or will be in fips-error again. If it persists in fips-error, the TOE will need to be RMAed.

Component Testing Assurance Activities: It is expected that at least the following tests are performed:

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

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

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

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

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

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

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

The evaluator initiated the FIPS self-test by using the command "fips self-test". This caused the TOE to begin a reboot. The evaluator viewed the boot-up output and observed that the self-tests ran and completed successfully.

2.5.5 TRUSTED UPDATE (NDCPP22E:FPT_TUD_EXT.1)

2.5.5.1 NDCPP22E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.3 NDcPP22e:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

Section 6.5 (Protection of the TSF) in the ST states that the administrator can query the current version of the TOE via the TOE management interfaces (local console, SSH (CLI, NETCONF) and HTTPS/TLS (Web UI, RESTCONF). For example, the software version can be viewed via System Properties in the Web UI dashboard or by issuing the 'swversion' command at the CLI command-line prompt (local console or SSH). The software update file is downloaded from the Infinera Customer Support Portal (https://support.infinera.com). Signature verification is performed when the image is uploaded to the TOE for both full software package upgrades and delta/patch upgrades. The file's digital signature is verified before it is saved to the TOE's flash. The TOE checks the update image integrity using ECDSA P-521 with SHA2-512. If the integrity verification fails, the TOE does not save the uploaded image to flash and the installation fails. Once the file is verified successfully and saved to the flash, the administrator issues commands or performs actions via the Web UI to install and activate the image. The TOE will then reboot and will come up following the reboot with the newly installed version of the software.

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

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

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

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

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

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

Section "FMT_MOF.1/ManualUpdate, FPT_TUD_EXT.1 and AGD_OPE.1" in the **Admin Guide** provides the steps and commands for performing a software update. The TOE supports administrator requested upgrade. A software release consists of a manifest file as well as a .tar.gz file. The manifest file and the .tar.gz file must have the same prefix and must be in the same directory. Only the manifest filename is used in the download command.

To determine the current version of software on the system use this command: 'swversion'. This will return the running version on the active controller and on the backup controller.

The software update file is downloaded from the Infinera Customer Support Portal (https://support.infinera.com). Signature verification is performed when the image is uploaded to the TOE for both full software package upgrades and delta/patch upgrades. The file's digital signature is verified before it is saved to the TOE's flash. The TOE checks the update image integrity using ECDSA P-521 with SHA2-512. If the integrity verification fails, the TOE does not save the uploaded image to flash and the installation fails.

Once the file is verified successfully and saved to the flash, the administrator issues commands or performs actions via the Web UI to install and activate the image. The TOE will then reboot and will come up following the reboot with the newly installed version of the software.

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

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

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

The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

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

2) An image that has not been signed

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

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

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

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

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

to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE.

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

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

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

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

Test 1: 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 was changed as expected.

Test 2: The vendor provided versions of an update that included a Valid Update, an update with no signature, and an update with an invalid signature. A hex editor was utilized to modify the package from the initially valid update. The evaluator performed the version verification activity, attempted each update, verified that the update failed, and performed version verification once again. Upon update failure, the "ERROR: Signature verification ... failed" message can be seen.

Test 3: Not applicable. The TOE utilizes digital signatures to verify updates.

2.6 TOE ACCESS (FTA)

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

2.6.1.1 NDCPP22E:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 (TOE access) in the ST states that an authorized administrator can configure both remote or local session idle timeout based on a time period of inactivity. The session idle timeout is the maximum amount of time an administrator may remain idle and is configured per user from 1 to 1440 minutes. A session (local or remote) that is inactive for the defined time period will be terminated and will require re-identification and authentication to establish a new session.

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

Section "FTA_SSL.3" in the **Admin Guide** provides the command for configuring the inactivity time period for remote administrative session termination.

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

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

The evaluator performed a test that demonstrates operation of inactivity timeouts over SSH with timeout values of 1 minute, 3 minutes and 5 minutes. For each time period, the evaluator recorded the time immediately prior to login, logged in, and performed no further actions. When the TOE terminated the SSH session for inactivity the evaluator recorded the time again. The period of time observed matched the configured timeout values. This test was performed via SSH and HTTPS Web GUI.

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

2.6.2.1 NDcPP22e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 (TOE access) in the ST states that Authorized administrators can terminate their own interactive sessions. They can log out of both local and remote CLI administrative sessions by issuing the 'exit' command and from the Web UI via the "logout" icon.

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

Section "FTA_SSL.4" in the **Admin Guide** states that the TOE supports administrator shutdown of their own interactive sessions. They can log out of both local and remote CLI administrative sessions by issuing the 'exit' command and from the Web UI via the "logout" icon.

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

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

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

Test 1: The evaluator successfully logged into the TOE via the serial CLI connection and then logged out of the TOE using the 'exit' command.

Test 2: The evaluator successfully logged into the TOE via the SSH CLI connection and then logged out of the TOE using the 'exit' command. This test was also performed via the Web UI using the "logout" icon.

2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.6.3.1 NDcPP22e:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 (TOE access) in the ST states that an authorized administrator can configure both remote or local session idle timeout based on a time period of inactivity. The session idle timeout is the maximum amount of time an administrator may remain idle and is configured per user from 1 to 1440 minutes. A session (local or remote) that is inactive for the defined time period will be terminated and will require re-identification and authentication to establish a new session.

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

Section "FTA_SSL_EXT.1" in the **Admin Guide** provides the command for configuring the inactivity time period for local interactive session termination.

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.

Test 1: The evaluator performed a test that demonstrates operation of inactivity timeouts on the local console with timeout values of 1 minute, 3 minutes and 5 minutes. For each time period, the evaluator recorded the time immediately prior to login, logged in, and performed no further actions. When the TOE terminated the local console session for inactivity the evaluator recorded the time again. The period of time observed matched the configured timeout values.

2.6.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.6.4.1 NDcPP22e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message

might be different for different administrative methods of access, and might be configured during initial configuration (e.g. via configuration file).

Section 6.6 (TOE access) in the ST states that at each administrative interface, the TOE displays an access banner with an administrator specified advisory notice and consent warning regarding use of the TOE. The banner displays on the local console, SSH CLI and the Web UI (HTTPS/TLS) interfaces prior to allowing any administrative access to the TOE.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Section "Establish the Pre-Login Banner message" in the **Admin Guide** provides the command for configuring the pre-login banner message. The banner is displayed on all interfaces (local console, SSH CLI and Web UI).

Component Testing Assurance Activities: The evaluator shall also perform the following test:

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

The evaluator configured a banner and verified that the banner was displayed appropriately for local console, SSH CLI and Web UI logins.

2.7 Trusted path/channels (FTP)

2.7.1 Inter-TSF trusted channel (NDcPP22e:FTP_ITC.1)

2.7.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 NDcPP22e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 (Trusted path/channels) in the ST states that the TOE protects communications between itself and external authentication (RADIUS, TACACS+), SYSLOG, and NTP servers using IPsec/IKE with pre-shared keys or X509 certificates for authentication and assured identification of the non-TSF endpoint. The TOE can either initiate IKE sessions or respond to IKE INIT requests from IPsec peers.

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.

Refer to NDcPP22e:FCS_IPSEC_EXT.1.1 where the guidance assurance activities describe the instructions provided in the **Admin Guide** for configuring the IPsec encryption parameters and the IPsec SPDs for each external IT entity.

Refer to NDcPP22e:FIA_X509_EXT.2.2 where the guidance assurance activities describe the instructions in the **Admin Guide** for loading CA certificates to the TOE's trust store, generating a certificate signing request and importing the signed certificate to the TOE are described.

Section "FTP_ITC.1.3" in the **Admin Guide** states that If an IPSEC connection fails, the status can be found by reviewing the SYSLOG for IPSEC. This is done with the following command: "show log security". Reconnection of an IPSEC session can be attempted using commands to lock and unlock the ikev2 peer. If a SYSLOG session fails, the syslog connection should be reattempted using commands to disable and then re-enable syslog. Similarly, if a TACACS+ or RADIUS server session fails, the TACACS+ or RADIUS adjacency will need to be re-established using commands to disable and re-enable the aaa server.

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.

Test 1-3: The evaluator followed the guidance documentation to configure secure IPsec connections between the TOE and external Syslog, RADIUS, TACACS+ and NTP test servers. The evaluator observed and confirmed via the packet captures that the TOE initiated the connection to in each case. The evaluator also observed that the channel data was encrypted and no channel data was sent in plaintext.

Test 4: The evaluator started a packet capture and established an IPsec connection between the TOE and the external test server. The evaluator then initiated the physical interruption of the connection for roughly 30 seconds and then restored the connection. The packet capture shows no traffic between the TOE and external

server during the disruption. Upon reconnection the TOE continued to use the original IKE and ESP sessions and did not establish a new IPsec tunnel between the TOE and the test server. The evaluator repeated this disruption test for a period of roughly 15 minutes. Upon reconnection this time, the evaluator observed that the TOE initiated a new IKE negotiation to establish a new IPsec tunnel between the TOE and the test server.

2.7.2 Trusted Path (NDcPP22e:FTP_TRP.1/Admin)

2.7.2.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 (Trusted path/channels) in the ST states that the TOE uses SSHv2 and HTTPS/TLS to secure communications between itself and authorized security administrators that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated

data from disclosure and detection of modification of the communicated data. The protocol allows administrators to initiate communications via the trusted path.

All remote administration of the TOE takes place over a secure communication path between the remote administrator and the TOE using either an SSHv2 client or a web browser.

- SSH Client – The remote administrator uses an SSH client to access the CLI and the NETCONF API over a secure, encrypted SSHv2 session.
- Web browser – The remote administrator uses a web browser to access the Web UI and the RESTCONF API over a secure HTTPS/TLS connection.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Section "Configure TOE security policies" and sub-sections in the **Admin Guide** provides instructions for configuring the login banner, generating host SSH keys and configuring remote administration using SSH. This includes SSH client authentication using either password or SSH public key.

Section "Configure System X.509 Certificate" in the **Admin Guide** provides instructions for loading CA certificates to the TOE's trust store. Section "Generate TOE Asymmetric Key Pair & issue Certificate Signing Request (CSR)" in the **Admin Guide** provides the command for generating TLS and IPsec certificate signing requests using RSA key sizes 2048, 3072 and 4096 bits and ECDSA key sizes eccp256, eccp384 and eccp521. The CSR block should be captured manually and sent to a Certificate Authority (CA) for processing. Section "Load the Signed Certificate for the TOE" provides further instructions on how to import the signed TOE certificate received from the Certificate Authority.

Section "Configure TLS application identity" in the **Admin Guide** provides the command for associating the TOE local X.509 certificate to be used as an identity certificate for TLS.

Section "WebGUI" in the **Admin Guide** provides the command for associating the TOE local X.509 certificate with the Web GUI.

Section "FTP_TRP.1.3/Admin and FIA_UIA_EXT.1" in the **Admin Guide** states that all remote administration of the TOE takes place over a secure communication path between the remote administrator and the TOE using either an SSHv2 client to access the CLI and the NETCONF API over a secure SSHv2 session, or a Web browser to access the Web UI and the RESTCONF API over a secure HTTPS/TLS connection.

For SSH, an SSH session is established to the TOE's IP address, TCP port 22. For NETCONF, an SSH session is established to the TOE's IP address, TCP port 830. For RESTCONF, an HTTPS session is established to the TOE's IP address, TCP port 8181.

The SSH and NETCONF sessions will negotiate the cryptographic algorithms used for their session from the algorithms configured on the client and the TOE. For RESTCONF, the negotiation is done based on the TLS1.2 algorithms configured on the client and the TOE.

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.

Test 1 and Test 2: Remote SSH and TLS/HTTPS administration was tested throughout the course of the evaluation. NDcPP22e:FCS_TLSS_EXT.1 testing demonstrates that the TLS/HTTPS management connection is not in plaintext. NDcPP22e:FCS_SSHS_EXT.1 testing demonstrates that the SSH channel data is not sent in plain text.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

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

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

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

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

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

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

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

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

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

For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps (per TD0536):

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

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

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

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

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

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

Section “Place the TOE into FIPS operation mode” and section “Enable FIPS mode” in the **Admin Guide** provide instructions for configuring the TOE into FIPS mode so that only approved and CAVP certified cryptographic algorithms are available. As detailed in the guidance assurance activities throughout this AAR, it also provides further instructions for configuring the specifically allowed cryptographic algorithms and parameters as claimed in the Security Target. This ensures that the TOE is configured into its evaluated configuration.

Section “AGD_OPE.1” in the **Admin Guide** further clarifies the scope of the evaluation by listing functionality that is excluded in the evaluation. It further specifies the cryptographic modules that provide all TOE cryptographic functions in the evaluated configuration and provides a warning indicating that no other cryptographic modules were used in the testing of the TOE. All sections in the **Admin Guide** which describe configuring TOE security functions include only those settings th at should be enabled and can be used in the evaluated configuration.

The process for updating the TOE is described above in NDcPP22e:FPT_TUD_EXT.1 where the guidance assurance activities identify the section of the **Admin Guide** that describes how the updates can be obtained and made accessible to the TOE, as well as the process for verifying updates to the TOE by verifying a digital signature . If the integrity verification fails, the TOE will not save the uploaded image to flash and the installation fails.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

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

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

In addition, the evaluator performs the EAs specified below.

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

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

The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

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

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

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

The preparative procedures must

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

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

The evaluation team had the following document to use when configuring the TOE:

- Infinera GX-G42 Optical Network Platform Release 6.2.10 Hardening Guide, Revision V002, January 2025 (**Admin Guide**)

The completeness of the documentation is addressed by its use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a 'hard label' (e.g., stamped into the metal, paper label) or a 'soft label' (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

The evaluator verified that the ST, TOE, and **Admin Guide** are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

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

See section 3.3.1 for an explanation of how all CM items are addressed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes 'evaluated configuration' instructions). The focus of the testing is to confirm that the requirements specified in Section 5.1.7 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan

for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

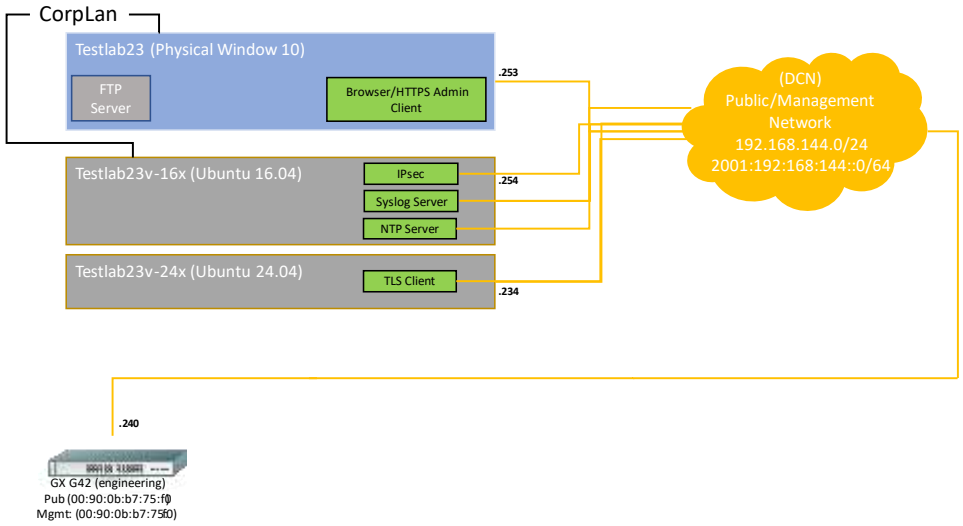
The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2, 3 and 4.

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

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

Testing took place from March 2024 through the date on this report within the Gossamer Security Solutions laboratory in Columbia, MD.

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.



TOE Platform:

- Infinera GX G42 Optical Network Platform running Converged OS (COS) 6.2.10

Supporting Products:

The Gossamer Test servers utilized both Windows and Ubuntu environments.

- Windows 10, 64-bit
  - Standard Windows utilities (e.g., notepad, snip tool)
  - Wireshark version 4.2.0 (used to exam network packets)
- Ubuntu 16.04
- Ubuntu 22.04

Supporting Software:

- SSH Client – Putty version 0.70
- Wireshark version 4.2.0
- Nmap version 7.01
- Ryslogd 8.16.0
- OpenSSL 1.0.2g-fips
- OpenSSL 3.0.13+GSS
- OpenSSH_9.3p1
- ntpd version 4.2.8p4
- Linux strongSwan U5.3.5
- curl 7.47.0
- scapy v2.2.0 running on python v2.7.12

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

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

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

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

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

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

The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic libraries, (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. (TD0547 applied)

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

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

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

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

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

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuzz testing. None of the public search for vulnerabilities, or the fuzz testing uncovered any residual vulnerability.

The evaluation team performed a public search for vulnerabilities in order to ensure there are no publicly known and exploitable vulnerabilities in the TOE from the following sources:

- National Vulnerability Database (https://web.nvd.nist.gov/vuln/search)

- Vulnerability Notes Database (http://www.kb.cert.org/vuls/)
- Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities)
- Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search)
- Offensive Security Exploit Database (https://www.exploit-db.com/)

The search was performed on 12/19/2024 with the following search terms: "Infinera", "GX-G42", "Converged OS", "COS", "Infinite Capacity Engine", "ICE6", "ICE7", "Infinera GX G40 G42 Cryptographic Module", "Intel Atom C3558", "NXP LS1012A", "NXPLS1012A".

3.5.2 Additional Flaw Hypothesis (AVA_VAN.1)

Assurance Activities: The following additional tests shall be performed:1.) [Conditional]: If the TOE is a TLS server and supports ciphersuites that use RSA transport (e.g. supporting TLS_RSA_WITH_* ciphers) the following test shall be performed. Where RSA Key Establishment schemes are claimed and especially when PKCS#1 v1.5* padding is used, the evaluators shall test for implementation flaws allowing Bleichenbacher and Klima et al. style attacks, including Bock et al's ROBOT attacks of 2017 in the flaw analysis. Even though Bleichenbacher's original paper is two decades old, Bock et al. found these attacks to still be effective in weakening the security of RSA key establishment in current products. Bleichenbacher and Klima et al. style attacks are complex and may be difficult to detect, but a number of software testing tools have been created to assist in that process. The iTC strongly recommends that at least one of the tools mentioned in Bock et al's ROBOT attacks of 2017 webpage or paper, as effective to detect padding oracle attacks, be used to test TOE communications channels using RSA based Key Establishment (related sources: http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf, https://eprint.iacr.org/2003/052, https://robotattack.org/). Network Device Equivalency Consideration.

The TOE is a TLS server, however, the TOE does not support cipher suites that use RSA transport, so this is not applicable.