www.GossamerSec.com

Assurance Activity Report for Cisco Embedded Services Router (ESR) 6300 v17.15

Version 0.2
07/14/25

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

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

Revision History

Revision	Date	Authors	Summary
Version 0.1	06/26/25	Le	Initial draft
Version 0.2	07/14/25	Le	ECR comments addressed

The TOE Evaluation was Sponsored by:
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706 USA

Evaluation Personnel:

- Linh Le
- Cody Cummins
- Douglas Kalmus

Common Criteria Versions:

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

Common Evaluation Methodology Versions:

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

Table of Contents

1. Introduction

This document presents evaluations results of the Cisco Embedded Services Router (ESR) 6300 v17.15 NDcPP30e/SSH10/VPNGW13 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 CAVP Certificates

The TOE provides the cryptography to support all security functions. All algorithms claimed have Cryptographic Algorithm Validation Program (CAVP certificates running on the processor specified in the table below.

Hardware Model	Processor
ESR-6300-NCP-K9	Marvell Armada ARMv8 Cortex A72
ESR-6300-CON-K9	Marvell Armada ARMv8 Cortex A72

The TOE leverages the IOS Common Cryptographic Module (IC2M), firmware version Rel5a (CAVP cert. #A1462).

SFR	Selection	Algorithm	Implementation	Standard	Certificate Number
FCS_CKM.1 – Cryptographic Key Generation	2048 3072	RSA	IC2M	FIPS PUB 186-4	A1462
FCS_CKM.1 – Cryptographic Key Generation and Verification	P-256 P-384	ECDSA	IC2M	FIPS PUB 186-4	A1462
FCS_CKM.1 – Cryptographic Key Generation	DH-14, DH-15, DH-16	FFC with safe-primes	IC2M	NIST SP 800-56A Rev 3	Tested with a known good implementation
FCS_CKM.2 – Cryptographic Key Establishment	P-256 P-384	KAS-ECC	IC2M	NIST SP 800-56A Rev 3	A1462
FCS_CKM.2 – Cryptographic Key Establishment	DH-14, DH-15, DH-16	FFC with safe-primes	IC2M	NIST SP 800-56A Rev 3	Tested with a known good implementation

SFR	Selection	Algorithm	Implementation	Standard	Certificate Number
FCS_COP.1/DataEncryption – AES Data Encryption/Decryption	AES-CBC-128 AES-CBC-192 AES-CBC-256 AES-GCM-128 AES-GCM-192 AES-GCM-256	AES	IC2M	ISO/IEC 18033-3 (AES) ISO/IEC 10116 (CBC) ISO/IEC 19772 (GCM)	A1462
FCS_COP.1/SigGen – Cryptographic Operation (Signature Generation and Verification)	2048 3072	RSA	IC2M	FIPS PUB 186-4	A1462
FCS_COP.1/SigGen – Cryptographic Operation (Signature Generation and Verification)	P-384	ECDSA	IC2M	FIPS PUB 186-4	A1462
FCS_COP.1/Hash – Cryptographic Operation (Hash Algorithm)	SHA-1 SHA-256 SHA-384 SHA-512	SHS	IC2M	ISO/IEC 10118-3:2004	A1462
FCS_COP.1/KeyedHash – Cryptographic Operation (Keyed Hash Algorithm)	HMAC-SHA-256 HMAC-SHA-384	HMAC	IC2M	ISO/IEC 9797-2:2011	A1462
FCS_RBG_EXT.1– Random Bit Generation	CTR_DRBG (AES) 256 bits	DRBG	IC2M	ISO/IEC 18031:2011	A1462

Table 1: Algorithm Certificate Numbers

The evaluator performed full testing on the following platforms and microarchitectures.

- ESR-6300-CON-K9 (Marvell Armada ARMv8 Cortex A72)

The only processor/microarchitecture combination is tested fully.

1.2 Platform Equivalence

The TOE is comprised of both software and hardware. The hardware models included in the evaluation are: ESR-6300-NCP-K9 and ESR-6300-CON-K9. The software is comprised of the Cisco IOS-XE 17.15. The evaluation tested the ESR-6300-CON-K9 device. The two hardware models use the exact same software, as well as the same management module and processor. They only differ in hardware aspects not relevant to the evaluation. Specifically, the ESR-6300-CON- K9 has a cooling plate while the ESR-6300-NCP-K9 does not. These characteristics affect only the non-TSF relevant functions of the devices. As a result, testing of just one device, the ESR-6300-CON-K9 device was sufficient to cover both devices.

During testing, the TOE was enclosed within a Cisco developed hardened enclosure. It is a specially designed enclosure used for Cisco internal testing purposes only. It has no compute capabilities and is not a commercially available product. In addition, the enclosure used for testing contains the ESR6300 board including the integrated multi-pin BTB interface connector with pins dedicated for power input, ethernet ports, and console ports (two combo Gigabit Ethernet WAN ports, four Gigabit Ethernet LAN ports, and one UART RS232 RJ-45 console port).

1.3 References

The following evidence was used to complete the Assurance Activities:

- Cisco Embedded Services Router (ESR) 6300 v17.15 Common Criteria Security Target, Version 1.0, 07/14/2025 [**ST**]
- Cisco Embedded Services Router (ESR) 6300 v17.15 Common Criteria Configuration Guide, Version 1.0, 07/14/2025 [**Admin Guide**] or **[AGD]**

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 (NDcPP30e:FAU_GEN.1)

2.1.1.1 NDcPP30e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP30e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6 of the [ST], FAU_GEN.1, states each of the events is specified in syslog records in enough detail to identify the user for which the event is associated, when the event occurred, where the event occurred, the

outcome of the event, and the type of event that occurred such as generating keys, including the type of key and a key reference.

The types of events that cause audit records to be generated include: startup and shutdown of the audit mechanism, cryptography related events, identification and authentication related events, and administrative events

The TOE is not distributed.

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

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

Section 5 of the Admin Guide contains two tables containing all the audit records. Table 5 General Auditable Events contains all of the relevant syslog messages that are produced by the TOE, including a description of each audit function.

Additionally, the use of administrative command relating to TSF configuration changes are covered in each of the Component Guidance Assurance Activities in this AAR. When the TOE is configured according to the AGD, all administrative actions (each command entered within the CLI) is audited which covers all necessary configuration changes. This is covered in the AGD section 3.3.5 Usage of Embedded Event Manager.

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

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

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

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

The TOE is not Distributed.

2.1.2 Audit Data Generation (VPN Gateway) (VPNGW13:FAU_GEN.1/VPN)

2.1.2.1 VPNGW13:FAU_GEN.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 VPNGW13:FAU_GEN.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the audit mechanisms that the TOE uses to generate audit records for VPN gateway behavior. If any audit mechanisms the

TSF uses for this are not used to generate audit records for events defined by FAU_GEN.1 in the Base-PP, the evaluator shall ensure that any VPN gateway-specific audit mechanisms also meet the relevant functional claims from the Base-PP.

For example, FAU_STG_EXT.1 requires all audit records to be transmitted to the OE over a trusted channel.

This includes the audit records that are required by FAU_GEN.1/VPN. Therefore, if the TOE has an audit mechanism that is only used for VPN gateway functionality, the evaluator shall ensure that the VPN gateway related audit records meet this requirement, even if the mechanism used to generate these audit records does not apply to any of the auditable events defined in the Base-PP.

Section 6 of the [ST], FAU_GEN.1/VPN, states the audit functionality is the same for VPN Gateway related events as is is for all other auditing behavior.

Refer to FAU_GEN.1 above.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it identifies all security-relevant auditable events claimed in the ST and includes sample records of each event type. If the TOE uses multiple audit mechanisms to generate different sets of records, the evaluator shall verify that the operational guidance identifies the audit records that are associated with each of the mechanisms such that the source of each audit record type is clear.

See NDcPP30e: FAU_GEN.1 where the evaluator confirmed that the Admin Guide identifies all auditable events claimed in the [ST] and includes sample records.

Component Testing Assurance Activities: The evaluator shall test the audit functionality by performing actions that trigger each of the claimed audit events and verifying that the audit records are accurate and that their format is consistent with what is specified in the operational guidance. The evaluator may generate these audit events as a consequence of performing other tests that would cause these events to be generated.

See NDcPP30e:FAU_GEN.1 where the evaluator also collected audit events relevant to VPNGW13. The evaluator found these audits events to be consistent with what is specified in the Admin Guide.

2.1.3 User identity association (NDcPP30e:FAU_GEN.2)

2.1.3.1 NDcPP30e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See NDcPP30e:FAU_GEN.1.

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

See NDcPP30e:FAU_GEN.1.

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

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

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

The TOE is not Distributed.

2.1.4 Protected Audit Trail Storage (NDcPP30e:FAU_STG.1)

2.1.4.1 NDcPP30e:FAU_STG.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 NDcPP30e:FAU_STG.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 (FAU_STG.1) of the ST states that for audit records stored internally to the TOE the audit records are stored in a circular log file where the TOE overwrites the oldest audit records when the audit trail becomes full. The size of the logging files on the TOE is configurable by the administrator with the minimum value being 4096 (default) to 2147483647 bytes of available disk space. It goes on to state that only Authorized Administrators are able to clear the local logs, and local audit records are stored in a directory that does not allow administrators to modify the contents.

The TOE is not distributed.

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

The TOE protects local audit storage from modification or deletion by restricting these abilities to Security Administrators. Section 4 “Secure Management” of the Admin Guide describes the steps to configure a user and an enable password which controls and protects these abilities.

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

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

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

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

Test 1: In other test cases, the evaluator found that the only action available to a user without authentication is to view the login banner. Unauthenticated users have no method available to modify or delete audit records.

Test 2: The evaluator logged into the TOE as an administrator. The evaluator issued a command to clear the local audit storage and found that this did delete the local audits stored on the TOE. The TOE does not include a method to mark specific records to be authorized for deletion.

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

2.1.5.1 NDcPP30e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.2 NDcPP30e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.3 NDcPP30e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.4 NDcPP30e:FAU_STG_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.5 NDcPP30e:FAU_STG_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.6 NDcPP30e:FAU_STG_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

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

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

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

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

Section 6.3 of the [ST], FAU_GEN.1, states the TOE is configured to export syslog records to a specified, external syslog server in real-time. The TOE protects communications with an external syslog server via IPsec. If the IPsec connection fails, the TOE will store audit records on the TOE when it discovers it can no longer communicate with its configured syslog server. When the connection is restored, the TOE will transmit the buffer contents when connected to the syslog server.

For audit records stored internally to the TOE the audit records are stored in a circular log file where the TOE overwrites the oldest audit records when the audit trail becomes full. The size of the logging files on the TOE is configurable by the administrator with the minimum value being 4096 (default) to 2147483647 bytes of available disk space.

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

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall also examine the guidance documentation to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements

on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

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

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

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

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

Section 3.3.6 of the Admin Guide discusses how to protect the audit log with IPsec. It provides instructions for an IPsec endpoint or peer. The Admin Guide also explains that when a Syslog server is configured on the TOE, generated audit events are simultaneously sent to the external server and the local logging buffer.

Section 5 of the Admin Guide states the local log buffer is circular. Newer messages overwrite older messages after the buffer is full. Administrators are instructed to monitor the log buffer using the show logging privileged EXEC command to view the audit records. The first message displayed is the oldest message in the buffer. The buffer size is configurable and can be set using the “logging buffer <size> command

When configured for a syslog backup the TOE will simultaneously offload events from a separate buffer to the external syslog server. This buffer is used to queue events to be sent to the syslog server if the connection to the server is lost. It is a circular buffer, so when the events overrun the storage space overwrites older events.

Only overwrite previous audit records according to the following rule: the newest audit record will overwrite the oldest audit record is selected for FAUU_STG_EXT.1.5.

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

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

a. Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this

transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing. The evaluator shall verify that the TOE is capable of transferring audit data to an external audit server automatically without administrator intervention.

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

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

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

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

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

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

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

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

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

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

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

a. Test 1: The evaluator configured the system (per guidance) to securely transfer audit data. As part of testing for FTP_ITC.1 the evaluator then generated audit data and captured network traffic between the TOE and the external audit server. The evaluator verified that the packet capture showed the audit data was not cleartext on the network and that the TOE initiated the connection to the audit server on its own. The external audit server used was rsyslog version 8.16.0.

b. Test 2: Not applicable, the TOE is not distributed.

c. Test 3: Persistent logging is selected, so case i.) applies. The evaluator performed an action to generate an audit event then found that audit locally on the TOE. The evaluator then power cycled the TOE. Upon fully rebooting, the evaluator was able to find the same audit on the TOE again, demonstrating that the log event was still present and unmodified within the local audit storage after a power cycle.

d. Test 4: Within test 4, only point ii.) is applicable, as the TOE claims to overwrite the oldest audit records first. The evaluator configured the system (per guidance) to securely transfer audit data. The evaluator captured network traffic between the TOE and the external audit server, and continued to generate audit data until the local storage space on the TOE was exceeded. The evaluator verified that when the local audit storage is filled to the maximum, the existing audit data is overwritten based on the following rule: overwrite oldest records first.

e. Test 5: Not applicable, the TOE is not distributed.

f. Test 6: The evaluator interrupted the connection between the TOE and its configured remote audit server (from "a. Test 1"). The evaluator then performed an action on the TOE that generated an auditable event. The evaluator found that despite the connection to the remote syslog server being interrupted, the TOE still generated the corresponding audit and stored it locally.

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation (NDcPP30e:FCS_CKM.1)

2.2.1.1 NDcPP30e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FCS_CKM.1, states The TOE implements DH-14, DH-15, and DH-16 key establishment schemes that meets *NIST Special Publication 800-56A Revision 3 and* RFC 3526. The TOE acts as both a sender and receiver for Diffie-Helman based key establishment schemes.

The TOE complies with section 5.6 and all subsections regarding asymmetric key pair generation and key establishment in the NIST SP 800-56A and with section 6.

Asymmetric cryptographic keys used for IKE peer authentication are generated according to FIPS PUB 186-4, Appendix B.3 for RSA schemes and NIST Special Publication 800-56A Revision 3 for FCC Schemes using 'safe-prime' groups.

The TOE can create an RSA public-private key pair, with a minimum RSA key size of 2048-bit (for CSfC purposes, the TOE is capable of a minimum RSA key size of 3072-bit) and ECDSA key pairs using NIST curves P-256 and P-384, and FFC key pairs. RSA schemes can be used to generate a Certificate Signing Request (CSR).

Scheme	**SFR**	**Service**
RSA Key generation	FCS_SSHS_EXT.1	SSH Remote Administration
	FCS_IPSEC_EXT.1	Transmit generated audit data to an external IT entity
	FIA_X509_EXT.1/Rev FIA_X509_EXT.2 FIA_X509_EXT.3	RSA certificate based authentication
FFC Key generation Key establishment	FCS_SSHS_EXT.1	SSH Remote Administration
	FCS_IPSEC_EXT.1	Transmit generated audit data to an external IT entity

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 4.6.4.1 of the Admin Guide explains how to generate a key pair for RSA with size 2048 and 3072 for IPsec certificates. Section 4.6.1.1 and 4.6.1.2 describe how to configure the Diffie-Hellman key exchange algorithms for IPsec.

Section 3.3.1.1 explains how to generate a key pair for RSA with size 2048 and 3072 and explains how to configure the Diffie-Hellman and ECDH key exchange algorithms for SSH.

This matches the requirements in the [ST].

Component Testing Assurance Activities: Key Generation for FIPS PUB 186-4 or FIPS PUB 186-5 RSA Schemes

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

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

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

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

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

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

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

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

To test the key generation method for the Random Provable primes method or Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. If the TSF provides the input to the key generation function, such input must be recorded and verified. For each RSA key length (modulo) claimed, the evaluator shall generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing Key Pair values generated by the TSF with those generated using the same set of input values using a known good implementation.

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

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

(TD0918 applied)

Key Generation for Elliptic Curve Cryptography (ECC)

Key Generation for ECDSA Schemes

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

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

- ECC Key Pair Generation

- ECC Public Key Validation (PKV)

ECC Key Pair Generation Test

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

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

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

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

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

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

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

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

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

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

(TD0918 applied)

ECC Public Key Verification (PKV) Test

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

(TD0918 applied)

Key Generation for FIPS PUB 186-5

FIPS 186-5 Key Generation Test

For the Ed25519 curve, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-5 Key Verification Test

For the Ed25519 curve, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

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

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

- Primes q and p shall both be provable primes

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

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

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

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

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

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

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

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

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

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

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

2.2.2 Cryptographic Key Generation (for IKE Peer Authentication) - per TD0878 (VPNGW13:FCS_CKM.1/IKE)

2.2.2.1 VPNGW13:FCS_CKM.1.1/IKE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure that the TSS describes how the key-pairs are generated. In order to show that the TSF implementation complies with FIPS PUB 186-5, the evaluator shall ensure that the TSS contains the following information:

- The TSS shall list all sections of Appendix B to which the TOE complies.

- For each applicable section listed in the TSS, for all statements that are not 'shall' (that is, 'shall not', 'should', and 'should not'), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as 'shall not' or 'should not' in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;

- For each applicable section of Appendix B, any omission of functionality related to 'shall' or 'should' statements shall be described;

Any TOE-specific extensions, processing that is not included in the Appendices, or alternative implementations allowed by the Appendices that may impact the security requirements the TOE is to enforce shall be described.

(TD0878 applied)

Section 6.3 of the [ST], FCS_CKM.1/IKE, states the TOE complies with section 5.6 and all subsections regarding asymmetric key pair generation and key establishment in the NIST SP 800-56A and with section 6. Asymmetric cryptographic keys used for IKE peer authentication are generated according to FIPS PUB 186-4, Appendix B.3 for RSA schemes and NIST Special Publication 800-56A Revision 3 for FCC Schemes using 'safe-prime' groups.

Component Guidance Assurance Activities: The evaluator shall check that the operational guidance describes how the key generation functionality is invoked, and describes the inputs and outputs associated with the process for each signature scheme supported. The evaluator shall also check that guidance is provided regarding the format and location of the output of the key generation process.

Section 4.6.4.1 of the Admin guide explains how to generate a key pair for RSA with size 2048 and 3072 for certificates for IPsec. The keys generated are saved in the private configuration in NVRAM (which is never displayed to the user or backed up to another device) the next time the configuration is written to NVRAM.

Section 4.6.1.1 and 4.6.1.2 describe how to configure the Diffie-Hellman key exchange algorithms for IPsec.

This matches the requirements in the [ST].

Component Testing Assurance Activities: For FFC Schemes using 'safe-prime' groups:

Testing for FFC Schemes using safe-prime groups is done as part of testing in FCS_CKM.2.

For all other selections:

The evaluator shall perform the corresponding tests for FCS_CKM.1 specified in the NDcPP SD, based on the selections chosen for this SFR. If IKE key generation is implemented by a different algorithm than the NDcPP key generation function, the evaluator shall ensure this testing is performed using the correct implementation.

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

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

2.2.3 Cryptographic Key Establishment (NDcPP30e:FCS_CKM.2)

2.2.3.1 NDcPP30E:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be as shown in the table below. The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

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

Section 6.3 of the [ST], FCS_CKM.2, states The TOE implements DH-14, DH-15, and DH-16 establishment schemes that NIST Special Publication 800-56A Revision 3 and RFC 3526 and DH-19 (256-bit Random ECP), and DH-20 (384-bit Random ECP) according to RFC 5114. The TOE acts as both a sender and receiver for Diffie-Helman based key establishment schemes.

The TOE complies with section 5.6 and all subsections regarding asymmetric key pair generation and key establishment in the NIST SP 800-56A and with section 6.

Asymmetric cryptographic keys used for IKE peer authentication are generated according to FIPS PUB 186-4, Appendix B.3 for RSA schemes and NIST Special Publication 800-56A Revision 3 for FCC Schemes using ‘safe-prime’ groups.

The TOE can create an RSA public-private key pair, with a minimum RSA key size of 2048-bit (for CSfC purposes, the TOE is capable of a minimum RSA key size of 3072-bit) and ECDSA key pairs using NIST curves P-256 and P-384, and FFC key pairs. RSA schemes can be used to generate a Certificate Signing Request (CSR).

Scheme	SFR	Service
RSA Key generation	FCS_SSHS_EXT.1	SSH Remote Administration
	FCS_IPSEC_EXT.1	Transmit generated audit data to an external IT entity
	FIA_X509_EXT.1/Rev FIA_X509_EXT.2 FIA_X509_EXT.3	RSA certificate based authentication
FFC Key generation Key establishment	FCS_SSHS_EXT.1	SSH Remote Administration
	FCS_IPSEC_EXT.1	Transmit generated audit data to an external IT entity

The TOE can create an RSA public-private key pair, with a minimum RSA key size of 2048-bit (for CSfC purposes, the TOE is capable of a minimum RSA key size of 3072-bit) and ECDSA key pairs using NIST curves P-256 and P-384, and FFC key pairs. RSA schemes can be used to generate a Certificate Signing Request (CSR).

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).

Section 4.6.1.1 and 4.6.1.2 of the Admin Guide describe how to configure the Diffie-Hellman key exchange algorithms for IPsec.

Section 3.3.1.1 how to configure the Diffie-Hellman and ECDH key exchange algorithms for SSH.

Component Testing Assurance Activities: Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests for ECC and FIPS186-type. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and

the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

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

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

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

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

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

Validity Test

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

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

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

RSA-based key establishment

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

FFC Schemes using 'safe-prime' groups

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

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

Testing for RSA-based key establishment was performed using a known implementation (OpenSSH for SSH and strongswan for IPsec). FFC Schemes was performed using a known implementation (OpenSSH for SSH and strongswan for IPsec).

2.2.4 Cryptographic Key Destruction (NDcPP30e:FCS_CKM.4)

2.2.4.1 NDcPP30e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

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

Section 7 of the [ST] provides a table that identifies all keys, provides a description of each key including where it is stored, and identifies how it’s cleared.

There are no keys that are stored in a non-plaintext form.

The [ST] does not identify any special configuration or circumstances needed to ensure keys are cleared.

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

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command [Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the

keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table)] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

The [ST] and the guidance do not identify any circumstances or configurations that do not strictly conform to the key destruction requirements.

Component Testing Assurance Activities: None Defined

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

2.2.5.1 NDcPP30e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST] states the TOE provides symmetric encryption and decryption capabilities using AES in GCM and CBC mode (128, 192 and 256 bits) as described in ISO 18033-3, ISO 19772 and ISO 10116 respectively.

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 4.6.1 and 4.6.2 of the Admin Guide specify how to select the modes and keys sizes for IPsec.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

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

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all

zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message,

using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

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

PT = IV

else:

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

PT = CT[i-1]

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

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

AES-GCM Test

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

128 bit and 256 bit keys

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

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

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

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

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

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

AES-CTR Known Answer Tests

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Due to the fact that Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested if the TSF is validated against the requirements of the Functional Package for Secure Shell referenced in Section 2.2 of the cPP. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

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

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

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

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

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

AES-CTR Multi-Block Message Test

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

AES-CTR Monte-Carlo Test

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

Input: PT, Key

for i = 1 to 1000:

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

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

There is no need to test the decryption engine.

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

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

2.2.6.1 NDcPP30e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3, FCS_COP.1/Hash, of the [ST] states the TOE provides cryptographic hashing services using SHA-1, SHA-256, and SHA-512 as specified in ISO/IEC 10118-3:2004.

The TOE provides keyed-hashing message authentication services using HMAC-SHA-256 which operates on 512-bit blocks and HMAC-SHA-384 operate on 1024-bit blocks of data, with key sizes and message digest sizes of 256 bits and 384 bits respectively) as specified in ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”.

For IKE (ISAKMP) hashing, administrators can select any of HMAC-SHA-256 and/or HMAC-SHA-384 (with message digest sizes of 256 and 384 bits respectively) to be used with remote IPsec endpoints.

SHA-512 hashing is used for verification of software image integrity.

The TOE provides Secure Hash Standard (SHS) hashing in support of SSH for secure communications. Management of the cryptographic algorithms is provided through the CLI with auditing of those commands.

For IPsec SA authentication integrity options administrators can select any esp-sha256-hmac, or esp-sha384-hmac (with message digest sizes of 256 and 384 bits respectively) to be part of the IPsec SA transform-set to be used with remote IPsec endpoints.

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 4.6.1 and 4.6.2 the Admin Guide specifies how to select hash mode for IPsec. Section 3.3.1 explains how to configure the hash size for SSH.

Hashing for Radius and software image integrity do not require any configuration beyond the enabling of FIPS mode which is described in Section 3.2.3.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

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

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

2.2.7.1 NDcPP30e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST] states that the TOE provides keyed-hashing message authentication services using HMAC-SHA-256 which operates on 512-bit blocks and HMAC-SHA-384 which operates on 1024-bit blocks of data, with key sizes and message digest sizes of 256 bits and 384 bits respectively) as specified in ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2".

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 3.3.1 of the Admin Guide explains how to configure the HMACs for SSH. Section 4.6.2 explains how to configure HMACs for IPsec.

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

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

2.2.8 Cryptographic Operation (Signature Generation and Verification) (NDcPP30e:FCS_COP.1/SigGen)

2.2.8.1 NDcPP30e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST] states that the TOE provides cryptographic signature services using RSA Digital Signature Algorithm with key size of 2048 and greater as specified in ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3.

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 4.6.4 of the Admin Guide explains how to use X.509 certificates and how to generate RSA certificates. These certificates can then be used for signature services.

Component Testing Assurance Activities: ECDSA Signature Algorithm Tests

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

Signature Generation Test

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

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

Signature Verification Test

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

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

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

RSA Signature Algorithm Tests

Signature Generation Test

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

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

a. RSASSA-PKCS1-v1.5

b. RSASSA-PSS

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

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

Signature Verification Test

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

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

a. RSASSA-PKCS1-v1.5

b. RSASSA-PSS

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

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

(TD0918 applied)

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

2.2.9 IPsec Protocol - per TD0868 (NDcPP30e:FCS_IPSEC_EXT.1)

2.2.9.1 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1 explains how SPDs are processed. A crypto map (the Security Policy Definition (SPD)) set can contain multiple entries, each with a different access list. The crypto map entries are searched in a sequence - the router attempts to match the packet to the access list (acl) specified in that entry. Separate access lists define blocking and permitting at the interface). For example:

Router# access-list 101 permit ip 192.168.3.0 0.0.0.255 10.3.2.0 0.0.0.255

When a packet matches a permit entry in a particular access list, the method of security in the corresponding crypto map is applied. If the crypto map entry is tagged as ipsec-isakmp, IPsec is triggered. For example:

Router# crypto map MAP_NAME 10 ipsec-isakmp

The match address 101 command means to use access list 101 in order to determine which traffic is relevant. For example:

Router# (config-crypto-map)#match address 101

The traffic matching the permit acls would then flow through the IPSec tunnel and be classified as “PROTECTED”. Traffic that does not match a permit acl and is also blocked by other non-crypto acls on the interface would be DISCARDED.

Traffic that does not match a permit acl in the crypto map, but that is not disallowed by other acls on the interface is allowed to BYPASS the tunnel. For example, a non-crypto permit acl for icmp would allow ping traffic to flow unencrypted if a permit access list entry was not configured that matches the ping traffic.

If there is no SA that the IPsec can use to protect this traffic to the peer, IPsec uses IKE to negotiate with the remote peer to set up the necessary IPsec SAs on behalf of the data flow. The negotiation uses information specified in the access list entry as well as the data flow information from the specific access list entry.

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 4.6.5 of the Admin Guide explains how to establish the SPDs. It provides a specific reference for bypass, discard, and protect. The section then provides an example on how to set a rule and how to apply it to an interface.

Testing Assurance Activities: The evaluator shall use the guidance documentation to configure the TOE to perform the following tests:

a. Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator shall perform both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator shall observe via the audit trail, and packet captures that the TOE exhibited the

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

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

Test 1 - The TOE implements SPD rules using access lists. The test performed was intended to demonstrate the relationship of different access lists and how they can be used to implement the various PROTECT, BYPASS, and DISCARD behaviors. This specific test example performed used IPV4, however access list using IPV6 were fully tested as part of VPNGW13:FPF_RUL_EXT.1.

The evaluator created access list rules for each type of SPD policy. The evaluator then verified the rules by establishing an IPsec tunnel and successfully establishing an administrator connection. It was observed that both the Bypass rule for administrator traffic and the Permit rule for IPsec traffic was successfully enforced against the traffic going through the IPsec tunnel (positive cases). The evaluator used the packet capture to verify the Discard rule as well as the default deny when the traffic matched no rule (negative cases). The selectors that were used involve both specific IP address source/destinations, as well as the protocol of traffic (TCP).

Test 2 - The results for VPNGW13:FPF_RUL_EXT.1.5 and VPNGW13:FPF_RUL_EXT.1.6 demonstrates correct TOE operation with respect to ordering and specificity of access list rules (overlapping and conflicting entries). The test results for IPSEC_EXT.1.1-t1 also demonstrated the case of packets that establish the SA, and packets that belong to an SA.

No deviations from the behavior outlined in the TSS and AGD were observed over the course of testing.

2.2.9.2 NDcPP30E: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 shall use the guidance documentation to configure the TOE to perform the following tests:

a. Test 1:The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet. The evaluator shall observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a 'TOE create'' final entry that discards packets that do not match any previous entries). The evaluator shall send the packet and observes that the packet was dropped.

The evaluator followed the instructions found within the AGD to setup an IPsec connection and create access-lists to control the flow of IPsec traffic.

The evaluator performed this test in conjunction with FCS_IPSEC_EXT.1.1. In that test, there were rules created for dropping, encrypting, and bypassing packets. Additionally, a final entry was created in the access list that discards packets that do not match any rules. The evaluator sent traffic matching traffic and observed that the rules were enforced as expected when the traffic matched the rule. The evaluator also sent traffic that did not match any of the access list rules and verified that a default deny was enforced.

2.2.9.3 NDcPP30e:FCS_IPSEC_EXT.1.3

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

Section 6.3 of the [ST] states that n addition to tunnel mode, which is the default IPsec mode, the TOE also supports transport mode, allowing fori only the payload of the packet to be encrypted. If tunnel mode is explicitly specified, the router will request tunnel mode and will accept only tunnel mode. This matches the requirement and the selections.

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

Section 4.6.2 of the Admin Guide explains how to configure tunnel mode on the TOE.

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

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

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

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

Test 1 and Test 2: For this test, the evaluator configured a test peer to require both tunnel mode and transport mode. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that a successful connection was established in tunnel mode as well as a successful connection was observed in transport mode.

2.2.9.4 NDcPP30e: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 shall ensure 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 6.3 of the [ST], FCS_IPSEC_EXT.1, states that the IPsec protocol ESP, as defined by RFC 4303, is implemented using the cryptographic algorithms AES-CBC-128 (RFC 3602), AES-CBC-256 (RFC 3602), AES-GCM-128 (RFC 4106), AES-GCM-256 (RFC 4106) and AES-CBC-192 (RFC 3602), AES-GCM-192 (RFC 4106) together with a Secure Hash Algorithm (SHA)-based HMAC [HMAC-SHA-256, HMAC-SHA-384].

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

Section 4.6.2 of the Admin Guide explains how to configure the ESP algorithm.

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 for AES-CBC-128, AES-CBC-192, AES-CBC-256, AES-GCM-128, AES-GCM-192 and AES-GCM-256. The evaluator then configured the TOE according to the guidance and verified that the connection could be established for each algorithm. For these tests the TOE and a test server were configured as peers and the evaluator configured the TOE to successfully establish the tunnel with the selected algorithm. The evaluator observed successful connections only for the algorithms claimed in the SFR selection.

2.2.9.5 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1, states the TOE supports both IKEv1 and IKEv2 session establishment. As part of this support, the TOE can be configured to disable aggressive mode for IKEv1 exchanges and to only use main mode using the 'crypto ISAKMP aggressive-mode disable' command.

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 4.6.1.1 explains how to configure IKEv1, which also includes instructions to disable aggressive mode for IKEv1 Phase 1 exchanges. Section 4.6.1.2 explains how to configure IKEv2. Section 4.6.3 provides the commands for establish NAT traversal.

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

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

b. Test 2: If NAT traversal is selected within the IKEv2 selection, the evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 7296, Section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

The evaluator configured the TOE using IKEv1. The evaluator initiated an IPsec connection from a test peer using aggressive mode and observed that the TOE rejected the IPsec connection.

The evaluator also configured the TOE using IKEv2 such that a VPN session from a test server traversed a NAT device. The evaluator initiated an IPsec connection and observed that the TOE correctly negotiated the NAT connection to establish a protected IPsec connection.

2.2.9.6 NDcPP30e:FCS_IPSEC_EXT.1.6

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

Section 6.1 of the [ST], FCS_IPSEC_EXT.1 states the TOE provides AES-CBC-128, AES-CBC-192 and AES-CBC-256 for encrypting the IKEv1 payloads, and AES-CBC-128, AES-CBC-192, AES-CBC-256, AES-GCM-128 and AES-GCM-256 for IKEv2 payloads.

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

Section 4.6.1.1 explains how to configure IKEv1. Section 4.6.1.2 explains how to configure IKEv2. Both sections contain instructions for selecting algorithms.

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 shall confirm the algorithm was that used in the negotiation.

The evaluator configured IKE profiles (for AES-CBC-128, AES-CBC-192, AES-CBC-256, AES-GCM-128 and AES-GCM-256) on the TOE. The evaluator then confirmed that the TOE could establish a session with each algorithm. For these tests the TOE and a test server were configured as peers and the evaluator configured each to use the same selected algorithms to establish the tunnel (the test was repeated for each algorithm). The GCM versions of the algorithms connected in IKEv2 only, as specified in the [ST].

2.2.9.7 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1, states the TOE supports configuration lifetimes of both Phase 1 SAs and Phase 2 SAs using the following command, lifetime. The time values for Phase 1 SAs can be limited up to 24 hours and for Phase 2 SAs up to 8 hours. The Phase 2 SA lifetimes can also be configured by an Administrator based on number of packets. The TOE supports configuring the maximum amount of traffic that is allowed to flow for a

given IPsec SA using the following command, 'crypto ipsec security-association lifetime'. The default amount is 2560KB, which is the minimum configurable value. The maximum configurable value is 4GB.

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

Section 4.6.2 of the Admin Guide discusses IPsec lifetimes. It provides the command to configure the value between 1-24 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 Phase1 SA lifetime that exceeds the Phase 1SA lifetime on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that a new

Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

A byte lifetime is not supported for phase 1.

The evaluator configured the Child SA value on the IPsec peers for 24 hours and the Phase 2 lifetime for 8 hours. The evaluator kept the connection alive for 24 hours. The evaluator observed that the phase 1 SA was renegotiated about 2 minutes before the 24-hour mark, and that the next phase 2 SA negotiation occurred approximately (but before) every 8 hours.

2.2.9.8 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1, states the TOE supports configuration lifetimes of both Phase 1 SAs and Phase 2 SAs using the following command, lifetime. The time values for Phase 1 SAs can be limited up to 24 hours and for Phase 2 SAs up to 8 hours. The Phase 2 SA lifetimes can also be configured by an Administrator based on number of packets. The TOE supports configuring the maximum amount of traffic that is allowed to flow for a given IPsec SA using the following command, 'crypto ipsec security-association lifetime'. The default amount is 2560KB, which is the minimum configurable value. The maximum configurable value is 4GB.

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

Section 4.6.2 of the Admin Guide discusses IPsec lifetimes. It provides the command to configure the value for 8 hours. This section also contains the command for configuring the kilobytes before a new association is required.

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

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

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

b. Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the lifetime of 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.

Test 1: The evaluator configured a maximum lifetime in number of bytes on the TOE and then configured a VPN peer with a byte lifetime that exceeded the lifetime of the TOE. The evaluator then established a connection with the VPN peer. The IPsec SA timed out after the packet size was exceeded and the connection reset. A new Phase 2 SA negotiation was required.

Test 2: The evaluator configured a Phase 2 SA lifetime timeout of 8 hours on the IPsec client. The evaluator then established a connection with an IPsec peer. The IPsec SA timed out just before 8 hours and the connection reset.

2.2.9.9 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1, states that the TOE supports Diffie-Hellman Group 14 (2048-bit keys), 19 (256-bit Random ECP), 20 (384-bit Random ECP), 15 (3072-bit MODP), and 16 (4096-bit MODP) in support of IKE

Key Establishment. The secret value 'x' used in the IKE Diffie-Hellman key exchange ("x" in gx mod p) is generated using a NIST-approved AES-CTR Deterministic Random Bit Generator (DRBG).

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.10 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1, states that the TOE supports Diffie-Hellman Group 14 (2048-bit keys), 19 (256-bit Random ECP), 20 (384-bit Random ECP), 15 (3072-bit MODP), and 16 (4096-bit MODP) in support of IKE Key Establishment. These keys are generated using the AES-CTR Deterministic Random Bit Generator (DRBG), as specified in SP 800-90, Table 2 in NIST SP 800-57 "Recommendation for Key Management -Part 1: General" and the following corresponding key sizes (in bits) are used: 224 (for DH Group 14), 256 (for DH Group 19), 256 (for DH Group 24), 384 (for DH Group 20) and 256 (for DH Group 15) bits.

Guidance Assurance Activities: None Defined

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

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

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

See TSS assurance activity above.

2.2.9.11 NDcPP30e: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 shall check to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Section 6.3 of the [ST] states that the TOE supports Diffie-Hellman Group 14 (2048-bit keys), 19 (256-bit Random ECP), 20 (384-bit Random ECP), 15 (3072-bit MODP), and 16 (4096-bit MODP) in support of IKE Key Establishment.

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

Section 4.6.1.1 explains how to configure IKEv1. Section 4.6.1.2 explains how to configure IKEv2. Both sections contain instructions for selecting the DH group. The sections also state what the evaluated groups are.

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.

Test 1 - The evaluator made an IPsec connection to an IPsec peer using each of the claimed DH groups. The evaluator was able to capture each DH group using a packet capture.

2.2.9.12 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1, states the strength of the symmetric algorithm negotiated to protect the IKEv1 Phase 1 and IKEv2 IKE_SA connection is greater than or equal to the strength of the symmetric algorithm negotiated to protect the IKEv1 Phase 2 or IKEv2 CHILD_SA connection. The administrator is instructed in the [AGD] to ensure that the size of key used for ESP must be greater than or equal to the key size used to protect the IKE payload.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall follow the guidance to configure the TOE to perform the following tests.

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

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

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

d. Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP (assumes the proper parameters 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.

Each test case was performed using IKEv1 and IKEv2.

Test 1: The evaluator attempted to establish a connection with each supported algorithm/hash combination. The connection attempts succeeded in each case.

Test 2: The evaluator attempted to establish a connection with an ESP algorithm that's stronger than the IKE algorithm. The evaluator viewed that the TOE rejected the connection.

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

Test 4: The evaluator attempted to establish a connection with an unsupported ESP algorithm. The connection attempt failed.

2.2.9.13 NDcPP30e:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: The evaluator shall ensure that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(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 6.3 of the [ST], FCS_IPSEC_EXT.1, explains the The IKE protocols implement Peer Authentication using RSA along with X.509v3 certificates, or pre-shared keys. RSA is claimed in the FCS_COP.1/SigGen requirement. Preshared keys can be configured using the 'crypto isakmp key' key command and may be proposed by each of the peers negotiating the IKE establishment.

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

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

Section 4.6.4 of the Admin Guide explains certificate usage in detail. It discusses how to create a certificate, connect to a CA, load a certificate, and how to use it for IPsec series.

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 performed in the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1.

2.2.9.14 NDcPP30e: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 6.3 of the [ST], FCS_IPSEC_EXT.1 states that when certificates are used for authentication, the distinguished name (DN) is verified to ensure the certificate is valid and is from a valid entity. The DN naming attributes in the certificate is compared with the expected DN naming attributes and deemed valid if the attribute types are the same and the values are the same and as expected. The fully qualified domain name (FQDN) can also be used as verification where the attributes in the certificate are compared with the expected SAN: FQDN, SAN: user FQDN and SAN: IPv4 Address.

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 4.8 of the Admin Guide describes how to set the reference identifier. It states that CNs are not supported. The accepted identifiers are Distinguished Name (DN), SAN: FQDN, SAN: user FQDN and SAN: IP Address.

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:

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

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

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

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

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

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

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

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

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

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

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

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

Test 1: (conditional) Not applicable, as the TOE does not support reference identifiers that are CNs.

Test 2: (conditional) The evaluator alternately configured Strongswan on a test peer to use an authentication certificate with the correct SAN of each supported type: FQDN, user FQDN and IPv4 address. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to be successful.

Test 3: (conditional) Not applicable, as the TOE does not support reference identifiers that are CNs.

Test 4: (conditional) The evaluator configured the TOE to accept an SAN reference identifier. The evaluator then configured the Strongswan VPN peer to use a certificate that would present an incorrect SAN: FQDN, user FQDN, and IPv4 address reference identifier. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to be rejected in each case.

Test 5: (conditional) The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN. The evaluator observed that the TOE accepted the connection.

Test 6a: (conditional) The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN except that the DN contained a duplicate CN field. The evaluator observed that the TOE did not accept the connection.

Test 6b: (conditional) The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN except that the DN contained a NULL character within the DN. The evaluator observed that the TOE did not accept the connection.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.10 NTP Protocol (NDcPP30e:FCS_NTP_EXT.1)

2.2.10.1 NDcPP30e: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 6.3 of the [ST], FCS_NTP_EXT.1, states that The TOE synchronizes with an NTP server for its reliable and accurate timestamp. The TOE can be configured to support at least three (3) NTP servers. The TOE supports NTPv4 and validates the integrity of the time-source using IPsec to provide trusted communication between itself and an NTP time source. In addition, the TOE does not allow the timestamp to be updated from broadcast addresses. NTP use UTC to synchronize computer clock times to a millisecond, and sometimes to a fraction of a millisecond.

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 4.3 Clock Management of the Admin Guide provides instructions to configure an NTP server (including multiple NTP servers) and specifies the command to enforce NTPv4 which is the version selected in the [ST]. It also specifies that an IPsec connection to the same server(s) as the configured NTP server(s) is required. Previous sections in the Admin Guide explain how to configure an IPsec connection.

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.

This test was performed as part of FCS_NTP_EXT.1.4-t1 where the packet capture confirms that the TOE establishes a calid connection to the external NTP server using NTPv4.

2.2.10.2 NDcPP30e: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 4.3 of the Admin Guide explains that an IPsec connection to the same server(s) as the configured NTP server(s) is required. Previous sections in the Admin Guide explain how to configure an IPsec connection.

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

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

The NTP claims in the [ST] are that NTP traffic flows over IPsec. The TOE's IPsec protocol was demonstrated in FCS_IPSEC_EXT.1.

2.2.10.3 NDcPP30e: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 4.3 of the Admin Guide explains that by default, NTP broadcast is disabled and the TOE will by default, not respond to broadcast & multicast NTP traffic. No configuration is necessary.

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 set the time on the NTP server ahead by approximately 10 minutes, configured the NTP server to send broadcast and multicast packets, and monitored the current time on the TOE. During this monitoring, the evaluator captured network traffic and saw that the TOE was not responding to the broadcast and multicast packets. The evaluator then configured the NTP on the TOE while continuing to monitor the time and network traffic. As confirmed by packet capture, the TOE ignored the broadcast and multicast time update, and attempted to update time using traditional authenticated updates after NTP was configured.

2.2.10.4 NDcPP30e:FCS_NTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

The evaluator shall confirm that the TOE would not synchronize to other, not explicitly configured time sources by sending an otherwise valid but unsolicited NTP Server 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.

Test 1 - The evaluator configured 3 NTP time sources on the TOE. The evaluator found that after a few minutes the TOE was able to sync up with one of the configured NTP servers. After a few additional minutes, the evaluator noted that the other 2 configured servers remained as valid NTP candidates.

Test 2 - Test 1 above demonstrates the successful syncing with a configured NTP server. To test that the TOE does not sync with an unconfigured server, the evaluator removed the supported NTP server from test 1 from the TOE configuration. A bogus NTP server address was configured in its place, to ensure that the TOE's time did not sync with a configured NTP server when performing this test. The evaluator set the time on the newly unconfigured NTP server ahead by 1 day and sent NTP packets from the NTP server to the TOE while monitoring the current time on the TOE compared to the time on the NTP server. As confirmed by packet capture, the TOE ignored the NTP packets sent from the now unconfigured NTP server.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.11 Random Bit Generation (NDcPP30e:FCS_RBG_EXT.1)

2.2.11.1 NDcPP30e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 NDcPP30e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST], FCS_RBG_EXT.1, states that the TOE implements a NIST-approved AES-CTR Deterministic Random Bit Generator (DRBG), as specified in ISO/IEC 18031:2011 seeded by an entropy source that accumulates entropy from a TSF-hardware based noise source (for CSfC purposes, AES-256).

The deterministic RBG 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 3.2.3 of the Admin Guide instructs the administrator to enable FIPS mode. This will ensure the RNG functionality is configured properly. No further configuration is needed.

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

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

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

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

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

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

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

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

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

2.2.12 SSH Protocol - per TD0732 & TD0777 (SSH10:FCS_SSH_EXT.1)

2.2.12.1 SSH10:FCS_SSH_EXT.1.1

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

The selections chosen for standards in this component (4251, 4252, 4253, 4254, 5647, 5656, 8308 section 3.1, 8332) are consistent with selections made is subsequent components.

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

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

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

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

2.2.12.2 SSH10:FCS_SSH_EXT.1.2

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

Section 6.3 of the [ST], FCS_SSH_EXT.1, states that the TOE's implementation of SSHv2 supports both public-key and password-based authentication. SSH public key authentication supports the rsa-sha2-256, rsa-sh2-512 and ecdsa-sha2-nistp384 algorithms.

This is consistent with the selections made in the SFR component, and additionally describes password-based authentication.

Keyboard-interactive is not selected.

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

The TOE supports SSH via passwords and public keys. Support for SSH passwords are enabled for authentication by default, outlined in section 4.2 "Passwords" in the Admin Guide**.** This section describes how to configure a password for a specific user.

Section " 3.3.1 Remote Administration Protocols" contains instructions to configure SSH public keys for authentication.

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

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

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

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

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

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

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

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

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

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

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

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

b. The evaluator shall verify that the connection fails.

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

Test 1a: The evaluator utilized an SSH client with debugging messages enabled, and attempted to connect to the TOE SSHS. The evaluator observed that the SSHC reported that the TOE only offered the configured authentication methods (public key, and password) as the possible authentication methods. These methods match the claim in the Security Target.

Test 1b: Not applicable, as the TOE does not support X509 based Client authentication.

Test 2 (all): Not applicable, as the TOE is not an SSH Client.

Test 3 (all): Not applicable, as the TOE is not an SSH Client.

2.2.12.3 SSH10:FCS_SSH_EXT.1.3

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

Section 6.3 of the [ST], FCS_SSH_EXT.1, states that packets greater than 65,806 bytes in an SSH transport connection are dropped. Large packets are detected by the SSH implementation, and dropped internal to the SSH process.

Guidance Assurance Activities: None Defined

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

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

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

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

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

(TD0732 applied)

Test 1: The evaluator established an SSH connection to the TOE SSHS. The evaluator then sent a packet of the maximum allowed size, as specified by the Security Target (65,806 bytes) over the established connection. The evaluator found that the TOE accepted this packet (i.e., the session remained active and was not terminated).

Test 2:

a: The evaluator established a successful SSH connection with the peer.

b: The evaluator crafted a packet of size 65,807 bytes. This is within the allowable range specified by the ATE. The evaluator then sent this crafted packet over the established connection.

c: The evaluator found that immediately upon receiving this packet, the TOE terminated the SSH connection.

2.2.12.4 SSH10:FCS_SSH_EXT.1.4

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

Section 6.3 of the [ST], FCS_SSH_EXT.1, states that the TOE uses the encryption algorithms, AES-CBC-128, AES-CBC-256 and aes256-gcm@openssh.com to ensure confidentiality of the session. This matches the requirement.

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

Section 3.3.1 of the Admin Guide explains how to configure SSH so that it meets the [ST] claims. It states to secure and control SSH sessions, the evaluated configuration requires SSHv2 session to only use AES-CBC-128, AES-CBC-256, and aes256-gcm@openssh.com encryption key algorithms.

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

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

Test 1: The evaluator must ensure that only claimed algorithms and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall establish an SSH connection with a remote endpoint. The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator

shall verify from the captured traffic that the TOE offers only the algorithms defined in the ST for the TOE for SSH connections. The evaluator shall perform one successful negotiation of an SSH connection and verify that the negotiated algorithms were included in the advertised set. If the evaluator detects that not all algorithms defined in the ST for SSH are advertised by the TOE or the TOE advertises additional algorithms not defined in the ST for SSH, the test shall be regarded as failed.

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

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

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

Test 1: The evaluator established a connection to the TOE SSHS. During this connection, traffic between the test SSHC and the TOE SSHS was captured. After the connection attempt, which was successful, the evaluator analyzed the captured traffic and derived algorithms that were offered by the TOE SSHS. From this, the evaluator was able to determine the following: The TOE SSHS offered only the algorithms claimed in the ST for SSH connections, and that the negotiated algorithms in the successful connection were in fact included in this advertised set. The evaluator did not detect any additional algorithms offered by the TOE SSHS that were not defined in the ST. These values derived from this test were used in later test cases.

Test 2: At the end of Test 1, the evaluator issued the 'exit' command specified in the AGD to terminate the SSH session. The evaluator found that the TOE terminated the SSH connection immediately after receiving this command.

Test 3: The evaluator attempted to connect to the TOE using an SSHC which was configured to use a mechanism (encryption algorithm) that was not included in the ST selection. The evaluator found that the TOE SSHS rejected this connection attempt.

2.2.12.5 SSH10:FCS_SSH_EXT.1.5

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

Section 6.3 of the [ST], FCS_SSH_EXT.1, states that The TOE's implementation of SSHv2 supports hashing algorithms hmac-sha2-256, hmac-sha2-384 and implicit to ensure the integrity of the session. This description matches the requirement selections.

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

Section 3.3.1 of the Admin Guide explains how to configure SSH so that it meets the [ST] claims. It states the TOE needs to be configured to only support the hmac-sha2-256, hmac-sha2-384 and implicit algorithms.

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

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

Test 1: Using the data collected in FCS_SSH_EXT.1.4 from the TOE SSHS, the evaluator derived the MAC algorithms that the TOE offered. The evaluator found that the appropriate mechanisms were advertised. Only algorithms that were claimed in the ST were advertised.

Test 2: The evaluator attempted to connect to the TOE using a SSH client with a MAC algorithm that is not included in the ST selection. The evaluator found that the TOE SSHS rejected the SSH connection attempt because of the mismatched algorithms.

2.2.12.6 SSH10:FCS_SSH_EXT.1.6

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

Section 6.3 of the [ST] states that the TOE's implementation of SSHv2 can be configured to allow Diffie-Hellman Group 14 (2048-bit keys) and ecdh-sha2-nistp384 Key Establishment. This description matches the requirement selections.

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

Section 3.3.1 of the Admin Guide explains how to configure SSH so that it meets the [ST] claims. It states the TOE needs to be configured to only support the diffie-hellman-group14-sha1 and ecdh-sha2-nistp384 algorithms.

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

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

Test 1: Using the data collected in FCS_SSH_EXT.1.4 from the TOE SSHS, the evaluator derived the Key Exchange algorithms that the TOE offered. The evaluator found that the appropriate mechanisms were advertised. Only algorithms that were claimed in the ST were advertised.

Test 2: The evaluator attempted to connect to the TOE using a SSH client with a Key Exchange algorithm that is not included in the ST selection. The evaluator found that the TOE SSHS rejected the SSH connection attempt because of the mismatched algorithms.

2.2.12.7 SSH10:FCS_SSH_EXT.1.7

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

Section 5.3.2 of the [ST] states that the TOE supports Key Derivation Functions (KDFs) as specified in RFC 4253, which includes the use of KDFs derived from the specified RFCs to ensure secure key establishment.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.8 SSH10:FCS_SSH_EXT.1.8

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

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

a. An argument describing this hardware-based limitation and

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

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

Section 6.3 of the [ST], FCS_SSHS_EXT.1, states the TOE can be configured to ensure that SSH re-key of no longer than one hour and no more than one gigabyte of transmitted data for the session key. Rekeying is performed upon reaching the threshold that is hit first.

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

Section 3.3.1 of the Admin Guide explains how to configure SSH so that it meets the [ST] claims. It provides instructions for configuring SSH rekey time-based rekey (in minutes) and volume-based rekey values (in kilobytes). The range of values meets the requirement. The Admin Guide includes a warning that says *Note: When configuring an SSH rekey time or volume interval, the TOE will begin re-key based upon the first threshold reached*.

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

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

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

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

Test 1: The evaluator first set the TOE's SSH rekey threshold to 10 minutes by following the AGD, using the command "ip ssh rekey time 10". The evaluator then attempted to connect to the TOE using a SSH client waiting to ensure that a rekey happened.

The evaluator attempted to connect to the TOE using a SSH client waiting an hour to ensure that a rekey happened the 10-minute threshold.

Test 2: The evaluator first set the TOE's SSH rekey threshold to exactly 1GB by following the AGD, by issuing the command "ip ssh rekey volume 1048576" (the value is in KB). The evaluator then attempted to connect to the TOE

using a SSH client waiting to ensure that a rekey happened. The evaluator connected to the TOE using an SSH client. The evaluator then repeatedly issued a command on the TOE's CLI that outputs more text than sent in the initial command. 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.

Test 3: The evaluator connected to the TOE using a SSH client. The evaluator then repeatedly issued the NULL character (ASCII value = 0) to the TOE. Sending the NULL character sends data to the TOE but the TOE does not output this character. This process causes the TOE to receive more data than it transmits back to the SSH client, causing the TOE received 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.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.13.1 SSH10:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

Section " 3.3.1 Remote Administration Protocols" contains instructions to configure/generate RSA SSH host keys.

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

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

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

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

(TD0682 applied)

Test 1: Using the data collected in FCS_SSH_EXT.1.4 from the TOE SSHS, the evaluator derived the Host Key algorithms that the TOE offered. The evaluator found that the appropriate mechanisms were advertised. Only algorithms that were claimed in the ST were advertised by the TOE SSHS.

Test 2: The evaluator attempted to connect to the TOE using a SSH client alternately using each of the authentication algorithms that can be claimed to determine which ciphers are supported with successful connections.

Test 3: The evaluator attempted to connect to the TOE SSHS using an SSHC that was configured to use a disallowed host key algorithm (ssh-dss). The evaluator found that the TOE rejected this connection attempt with a disconnect message.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication Failure Management (NDcPP30e:FIA_AFL.1)

2.3.1.1 NDcPP30e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 NDcPP30e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST], FIA_AFL.1, states that The TOE provides the privileged administrator the ability to specify the maximum number of unsuccessful authentication attempts before privileged administrator or non-privileged administrator is locked out through the administrative CLI using a privileged CLI command. While the TOE supports a range from 1-25, in the evaluated configuration, the maximum number of failed attempts is recommended to be set to 3. All successive unsuccessful authentication attempts are logged on the router.

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

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

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

Section 3.2.6 of the Admin Guide explains how to configure the maximum failed login value. The section then explains how to unlock the locked account. It also notes that the lockout does not apply to the local console.

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 configured a limit on failed local authentication attempts. The first attempt set a limit of 3 failure and the second attempt set a limit of 5 attempts. In each instance, the evaluator performed the same number of login attempts using incorrect credentials than the configured limit. After exceeding the limit, the evaluator observed via the “show aaa local user blocked command” as a separate admin user that the first user was locked out. The evaluator observed that the use of valid credentials does not result in a successful login. The evaluator then proceeded to unlock the account that exceeded the authentication attempts via an administrator account.

The time period selection in FIA_AFL.1.2 is not included in the [ST].

2.3.2 Password Management (NDcPP30e:FIA_PMG_EXT.1)

2.3.2.1 NDcPP30e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the TSS:

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

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

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

Section 6.3 of the [ST], FIA_PMG_EXT.1 states that The TOE supports the local definition of users with corresponding passwords. The passwords can be composed of any combination of upper and lower case letters, numbers, and special characters (that include: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")", and other special characters listed in the SFR. Minimum password length is settable by the Authorized Administrator, and supports passwords of 1 to 127 characters. A minimum password length of 15 is recommended.

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 4.2 of the Admin Guide describes passwords. It provides the character set and discusses how to set the minimum length of a password. The section also makes recommendations on selecting strong passwords.

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

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

b. Test 2: The evaluator shall compose passwords that do not meet the requirements in some way. For each password, the evaluator shall verify that the TOE does not support the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that the TOE

enforces the allowed characters and the minimum length listed in the requirement and justify the subset of those characters chosen for testing.

The evaluator attempted to set/change a password for a user's account using several attempts. Throughout those attempts, every upper-case letter, lower case letter, digit, and special character (as specified by the SFR in the [ST]) were used in a password. The evaluator also confirmed that a minimum length of 8 was required by attempting to set passwords with 7 characters (and observing the TOE reject the password) and of 8 characters (and observing that the TOE accepted the password change). The recommended value of 15, which is the minimum allowable value by the requirement, was also tested.

Initially, for the above test steps the password complexity requirements were disabled to make it easier to test the suite of possible characters. In the CC configuration, complexity requirements must be enabled. The evaluator enabled these complexity requirements and demonstrated they were enforced correctly.

2.3.3 Pre-Shared Key Composition - per TD0838 (VPNGW13:FIA_PSK_EXT.1)

2.3.3.1 VPNGW13:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.3.2 VPNGW13:FIA_PSK_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it identifies all protocols that allow pre-shared keys. For each protocol identified by the requirement, the evaluator shall confirm that the TSS states which pre-shared key selections are supported.

Section 6.3 of the [ST], FIA_PSK_EXT.1 states that through the implementation of the CLI, the TOE supports use of IKEv1 (ISAKMP) and IKEv2 pre-shared keys for authentication of IPsec tunnels. Pre-shared keys can be entered as ASCII character strings, or HEX values. The TOE supports keys that are from 22 characters in length up to 127

bytes in length and composed of any combination of upper- and lower-case letters, numbers, and special characters (that include: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”, and other special characters listed in the SFR. The data that is input is conditioned by the cryptographic module prior to use via SHA-1.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure all selected pre-shared key options if any configuration is required.

The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure the mandatory_or_not flag per RFC 8784. (TD0838 applied)

Section 4.6.1 contains instructions to configure both text-based and bit-based pre-shared keys for IPsec authentication.

Section 4.6.7 also contains instructions for configuring a PPK on the TOE.

Component Testing Assurance Activities: The evaluator shall also perform the following tests for each protocol (or instantiation of a protocol, if performed by a different implementation on the TOE).

Test 1: For each mechanism selected in FIA_PSK_EXT.1.2 the evaluator shall attempt to establish a connection and confirm that the connection requires the selected factors in the PSK to establish the connection in alignment with table 1 from RFC 8784. (TD0838 applied)

The evaluator configured the TOE and a test server to use generated bit-based pre-shared keys for authentication during IPsec IKEv2 negotiations. Using this configuration, the evaluator was able to establish an IPsec connection between the TOE and the IPsec test peer. Generated bit-based pre-shared keys is the only mechanism selected in FIA_PSK_EXT.1.2.

During this test the evaluator additionally configured a Postquantum Pre-shared Key (PPK) on the TOE and observed that it was able to be used in a connection to confirm the TOE’s ability to comply with RFC 8784.

2.3.4 Generated Pre-Shared Keys (VPNGW13:FIA_PSK_EXT.2)

2.3.4.1 VPNGW13:FIA_PSK_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'generate' is selected, the evaluator shall confirm that this process uses the RBG specified in FCS_RBG_EXT.1 and the output matches the size selected in FIA_PSK_EXT.2.1.

Not applicable, 'generate' is not selected in the [ST].

Component Guidance Assurance Activities: The evaluator shall confirm the operational guidance contains instructions for entering generated pre-shared keys for each protocol identified in the FIA_PSK_EXT.1.1.

Section 6.1 of the [ST], FCS_IPSEC_EXT.1 states that pre-shared keys can be configured using the 'crypto isakmp key' key command and may be proposed by each of the peers negotiating the IKE establishment. IPsec is the only protocol that the TOE identifies support for PSK.

Component Testing Assurance Activities: Test 1: [conditional] If generate was selected the evaluator shall generate a pre-shared key and confirm the output matches the size selected in FIA_PSK_EXT.2.1.

Not applicable, as "generate" was not selected.

2.3.5 Protected Authentication Feedback (NDcPP30e:FIA_UAU.7)

2.3.5.1 NDcPP30e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

Section 6.1 of the [ST], FIA_UAU.7, states that when a user enters their password at the local console, the TOE displays only '*' characters so that the user password is obscured. For remote session authentication, the TOE does not echo any characters as they are entered. The TOE does not provide a reason for failure in the cases of a login failure. No special configuration is required so no guidance is needed.

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

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

Test 1: The evaluator observed during testing that passwords are obscured on the console login.

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

2.3.6.1 NDcPP30e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.6.2 NDcPP30e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.6.3 NDcPP30e:FIA_UIA_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.6.4 NDcPP30e:FIA_UIA_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

Section 6.3 of the [ST], FIA_UAU_EXT.1, explains the authentication process. The TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed except for the login warning banner that is displayed prior to user authentication.

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

The TOE provides a local password-based authentication mechanism as well as RADIUS AAA server for remote authentication.

The administrator authentication policies include authentication to the local user database or redirection to a remote authentication server. Interfaces can be configured to try one or more remote authentication servers, and then fail back to the local user database if the remote authentication servers are inaccessible.

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

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

The TOE is not distrbibuted.

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

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

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

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

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

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

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

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

- Local Console
- SSH using passwords
- SSH using public/private key pairs
- SSH Connection using Radius Account

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

Test 2 - Using each interface the evaluator was able to observe the TOE displayed a banner to the user before login. In addition to entering password, viewing the login banner was the only action available to an unauthenticated user. The evaluator additionally performed a scan of open services on the TOE and found that only the SSH service (on port 22) was presented.

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

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

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

2.3.7.1 NDcPP30e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is expected that either OCSP or CRL revocation checking is performed when a certificate is presented to the TOE (e.g. during authentication). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3

certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The TOE uses X.509v3 certificates as defined by RFC 5280 (and RFC 8603 for CSfC purposes) to support authentication for IPsec connections. The following tests were performed using IPsec with certificate authentication.

Test 1a/b -- The evaluator configured the TOE and a peer with valid certificates. The evaluator then attempted to make a connection between the peer devices. A successful connection was made. The evaluator then configured a server certificate with an invalid certification path by deleting an intermediate root CA so that the certificate chain was invalid because of a missing (or deleted) certificate. The connection between the peers was refused.

Test 2 -- The evaluator initiated an IPsec connection to the TOE from a test server. The test server then presented a client certificate during the IPsec negotiation where the client certificate was expired. The TOE rejected the expired certificate and did not establish a connection. This was repeated with each intermediate CA in the certificate chain.

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

Test 4 -- The evaluator configured a server to present a certificate using a CRL signed by a CA that does not have the CRL signing purpose. The evaluator established an IPsec connection from the test server such that the TOE receives the invalid CRL and ensured that the IPsec connection was not negotiated successfully.

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

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

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

Test 8a/b/c -- Not applicable, as the TOE does not claim support for ECDSA certificates.

2.3.7.2 NDcPP30e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

a. Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the

leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

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

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

Test 1: The evaluator configured a test server to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then initiated an IPsec connection to the TOE and observed that the TOE rejected the connection.

Test 2: The evaluator configured a test server to present a certificate chain containing a CA certificate having the basicConstraints section but with the cA flag not set (i.e., FALSE). The evaluator then initiated an IPsec connection to the TOE and observed that the TOE rejected the connection.

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

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

Section 6.3 of the [ST], FIA_X509_EXT.1/Rev, explains the certificate chain establishes a sequence of trusted certificates, from a peer certificate to the root CA certificate. Within the PKI hierarchy, all enrolled peers can validate the certificate of one another if the peers share a trusted root CA certificate or a common subordinate CA. Each CA corresponds to a trust point. When a certificate chain is received from a peer, the default processing of a certificate chain path continues until the first trusted certificate, or trust point, is reached. The administrator may configure the level to which a certificate chain is processed on all certificates including subordinate CA certificates.

To verify, the authorized administrator could 'show' the PKI certificates and the PKI trust points.

The authorized administrator can also configure one or more certificate fields together with their matching criteria to match. Such as:

- alt-subject-name
- expires-on
- issuer-name
- name
- serial-number
- subject-name
- unstructured-subject-name
- valid-start

This allows for installing more than one certificate from one or more CAs on the TOE. For example, one certificate from one CA could be used for one IPsec connection, while another certificate from another CA could be used for a different IPsec connection. However, the default configuration is a single certificate from one CA that is used for all authenticated connections.

CRL is configurable and may be used for certificate revocation. The authorized administrator could use the "revocation-check" command to specify at least one method of revocation checking; CRL is the default method and must be selected in the evaluated configuration as the 'none' option is not allowed. The authorized administer sets the trust point and its name and the revocation-check method.

Checking is also done for the basicConstraints extension and the CA flag to determine whether they are present and set to TRUE. The local certificate that was imported must contain the basic constraints extension with the CA flag set to true, the check also ensure that the key usage extension is present, and the keyEncipherment bit or the keyAgreement bit or both are set. If they are not, the certificate is not accepted.

The certificate chain path validation is configured on the TOE by first setting crypto pki trustpoint name and then configuring the level to which a certificate chain is processed on all certificates including subordinate CA certificates using the chain-validation command. If the connection to determine the certificate validity cannot be established, the certificate is not accepted, and the connection will not be established.

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 4.6.4 of the Admin Guide explains the rules for EKU fields including the required field, and how the fields are validated for server/client authentication.

This same section also states that CRL is the default revocation checking method and explains that it is configured under the trust point settings.

Component Testing Assurance Activities: None Defined

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

2.3.8.1 NDcPP30e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 NDcPP30e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST], FIA_X509_EXT.2, explains the certificate chain establishes a sequence of trusted certificates, from a peer certificate to the root CA certificate. Within the PKI hierarchy, all enrolled peers can validate the certificate of one another if the peers share a trusted root CA certificate or a common subordinate CA. Each CA corresponds to a trust point. When a certificate chain is received from a peer, the default processing of a certificate chain path continues until the first trusted certificate, or trust point, is reached. The administrator may configure the level to which a certificate chain is processed on all certificates including subordinate CA certificates.

To verify, the authorized administrator could 'show' the PKI certificates and the PKI trust points.

The authorized administrator can also configure one or more certificate fields together with their matching criteria to match. Such as:

- alt-subject-name
- expires-on
- issuer-name
- name
- serial-number
- subject-name
- unstructured-subject-name
- valid-start

This allows for installing more than one certificate from one or more CAs on the TOE. For example, one certificate from one CA could be used for one IPsec connection, while another certificate from another CA could be used for a different IPsec connection. However, the default configuration is a single certificate from one CA that is used for all authenticated connections.

CRL is configurable and may be used for certificate revocation. The authorized administrator could use the "revocation-check" command to specify at least one method of revocation checking; CRL is the default method and must be selected in the evaluated configuration as the 'none' option is not allowed. The authorized administer sets the trust point and its name and the revocation-check method.

Checking is also done for the basicConstraints extension and the CA flag to determine whether they are present and set to TRUE. The local certificate that was imported must contain the basic constraints extension with the CA flag set to true, the check also ensure that the key usage extension is present, and the keyEncipherment bit or the keyAgreement bit or both are set. If they are not, the certificate is not accepted.

The certificate chain path validation is configured on the TOE by first setting crypto pki trustpoint name and then configuring the level to which a certificate chain is processed on all certificates including subordinate CA certificates using the chain-validation command. If the connection to determine the certificate validity cannot be established, the certificate is not accepted, and the connection will not be established.

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

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

a. Test 1: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

The evaluator established a syslog connection from the TOE to a remote test server protected by IPsec and obtained a packet capture of the activity. This was repeated when the CRL location was available and unavailable. When available the connection succeeded. When the CRL was unavailable, the connection failed.

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

2.3.9.1 NDcPP30e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.2 NDcPP30e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Not applicable, as 'device-specific information' is not selected.

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 4.6.4.2 of the Admin Guide provides instructions for generating a CSR including the required fields.

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

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

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

Test 1- The evaluator generated a certificate signing request by following the instructions in the [AGD] for generating the request. The request was then exported to an external CA through a trusted path. While the CSR was within the CA, the evaluator examined the CSR and found that it included the fields identified in the Security

Target. The CSR was also used to generate certificates on the CA which was returned to the TOE (also through a trusted path).

Test 2 - The evaluator tested that a certificate without a corresponding, valid, and trusted CA cannot be imported into the TOE manually.

2.4 Security management (FMT)

2.4.1 Management of Security Functions Behaviour (ndcpp30e:fmt_mof.1/Functions)

2.4.1.1 ndcpp30e:fmt_mof.1.1/Functions

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST], FMT_MOF.1/Functions states that the TOE provides the ability for Security Administrators to access TOE data, such as audit data.

Section 6.3 of the [ST], FAU_GEN.1 states that the audit records are transmitted using IPSec tunnel to the syslog server. If the communications to the syslog server is lost, the TOE generates an audit record and all permit traffic is denied until the communications is re-established.

The [ST] contains further details of how a Security Administrator can configure an IPsec endpoint for audit transmission in FCS_IPSEC_EXT.1.

The TOE is not distributed.

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

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

The TOE supports modifying the behavior or transmission of audit data to an external IT entity.

Section 3.3.6.1 "Syslog over IPsec" contains instructions for a Security Administrator to modify the transmission of audit data to the remote syslog server (protected by IPsec).

Section 3.3.4 also contains general instructions for configuring the logging location.

The TOE is not distributed.

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

The evaluator shall perform the following tests:

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

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

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

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

The evaluator shall perform the following tests:

a. Test 1: The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as Security Administrator (by authentication as a user with no administrator

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

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

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

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

The evaluator shall perform the following tests:

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

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

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

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

The evaluator shall perform the following tests:

a. Test 1: The evaluator shall try to determine the behaviour of all options chosen from the second selection without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This can be done in one test or in separate tests. The attempt(s) to

determine the behaviour of the selected functions without administrator authentication shall fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

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

The selection in the [ST] is [modify the behavior of] the functions [transmission of audit data to an external IT entity].

Test 1 (if 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator found in FIA_UIA_EXT.1 that prior to authentication as a security administrator, the only action a user could perform on the TOE was to view the login banner. Therefore, an unauthenticated user had no ability to modify the transmission of audit data.

Test 2 (if 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator logged in as a security administrator and found that they had the ability to configure the logging parameters. With these privileges, the evaluator added a syslog logging destination and found that the TOE accept the change.

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

2.4.2.1 NDCPP30E:FMT_MOF.1.1/MANUALUPDATE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see Section 2.4.1.1. There are no specific requirements for non-distributed TOEs.

Not applicable, as the TOE is not distributed.

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

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

Section 2 of the Admin Guide explains how to perform a manual update of the TOE. It explains the image on the Cisco web site and as part of the install, the signature will be validated. If the signature is not correct, the device will not boot. Section 2, steps 7 and 9 provide instructions for how to download and verify an image prior to running it on the TOE. The upgrade is not finalized until the TOE is rebooted.

The second part of the assurance activity is not applicable, as the TOE is not distributed.

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

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

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

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

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

The TOE is not distributed for the second part of the assurance activity.

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

2.4.3.1 NDcPP30e:FMT_MOF.1.1/Services

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST], FMT_MOF.1/Services, provides a reference to FMT_SMF.1 for a list of services that can be started and stopped. All the functions listed in that summary can be managed by the administrator.

The TOE is not distributed.

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

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

Section 6 of the Admin Guide provides a list of services that can be started or stopped by the administrator. It also references a command reference for manipulating the services.

The TOE is not distributed.

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

a. Test 1: The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) without prior authentication as Security Administrator (either by authenticating as a user with no administrator privileges, if possible, or without prior authentication at all). The attempt to enable/disable this service/these services should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to enable/disable this service/these services can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

b. Test 2: The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) with prior authentication as Security Administrator. The attempt to enable/disable this service/these services should be successful.

The operations to start and stop TOE services are only available to authorized administrators who can login to the TOE. As specified by FIA_UIA_EXT.1, the set of functions available to a user prior to login do not include operations to start and stop TOE services.

Refer to the results of FIA_UIA_EXT.1 that demonstrate the limited functions available to users prior to login.

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

2.4.4.1 NDcPP30e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST], FMT_MOF.1/CoreData, states the TOE provides the ability for Security Administrators to access TOE data, such as audit data, configuration data, security attributes, routing tables, and session thresholds, to securely manage certificates in the TOE's trust store, and to perform manual updates to the TOE. Each of the predefined and administratively configured roles has create (set), query, modify, or delete access to the TOE data. The TOE performs role-based authorization, using TOE platform authorization mechanisms, to grant access to the semi-privileged and privileged roles. For the purposes of this evaluation, the privileged role is equivalent to full administrative access to the CLI, which is the default access for IOS-XE privilege level 15; and the semi-privileged role equates to any privilege level that has a subset of the privileges assigned to level 15. Privilege levels 0 and 1 are defined by default and are customizable, while levels 2-14 are undefined by default and are also customizable.

The term "Security Administrator" is used in this ST to refer to any user which has been assigned to a privilege level that is permitted to perform the relevant action; therefore, has the appropriate privileges to perform the requested functions. Therefore, semi-privileged administrators with only a subset of privileges can also modify TOE data based on if granted the privilege. No administrative functionality is available prior to administrative login.

The TOE contains a trust store of X.509v3 certificates. The trust store contains certificates for the local TOE and certificates for the remote syslog server. Access to trust store data on each component is restricted to authorized administrators only.

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

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

Section 4.6.4 of the Admin Guide explains how to use X.509 certificates with the TOE. It explains how to request a certificate, how to communicate with a certificate authority and how to load a certificate.

The guidance activities found throughout the AAR address the remaining TSF-data-manipulating functions in the relevant SFR.

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.

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

2.4.5.1 NDcPP30e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST] states the TOE provides the ability for Security Administrators to generate, import, modify or delete cryptographic keys and certificates used for VPN operation through the TOE CLI as described in Section 7 Key Zeroization and in the [ST].

The TOE is not distributed.

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

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

Section 4.6 of the Admin Guide provides the instructions for generating, import, modifying or deleting cryptographic keys and certificates used for VPN operation.

The TOE is not distributed.

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

a. Test 1: The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

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

As can be seen in the FIA_UIA_EXT.1 test evidence, no functions are offered to users prior to a successful login.

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

The evaluator also attempted to log in as a non-administrative user with privilege level 1. The evaluator found that this user did not have any method available to modify, delete, or generate/import cryptographic keys as an administrative user would.

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

2.4.6.1 NDcPP30e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

(containing also requirements on Guidance Documentation and Tests)

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

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

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

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

Section 6.3 of the [ST], FMT_SMF.1, provides a description of the management functions of the TOE. The TOE provides all the capabilities necessary to securely manage the TOE and the services provided by the TOE. The

management functionality of the TOE is provided through the TOE CLI. The specific management capabilities available from the TOE include -

- Local and remote administration of the TOE and the services provided by the TOE via the TOE CLI, as described above; (NDcPP30e:FIA_UIA_EXT.1 and SSH10:FCS_SSH_EXT.1)
- The ability to manage the warning banner message and content - allows the Authorized Administrator the ability to define warning banner that is displayed prior to establishing a session (note this applies to the interactive (human) users; e.g. administrative users (NDcPP30e:FIA_UIA_EXT.1)
- The ability to manage the time limits of session inactivity which allows the Authorized Administrator the ability to set and modify the inactivity time threshold. (NDcPP30e:FTA_SSL.3)
- The ability to update the IOS-XE software. The validity of the image is provided using SHA-512 and digital signature prior to installing the update (NDcPP30e:FTP_TUD_EXT.1)
- The ability to manage audit behavior and the audit logs which allows the Authorized Administrator to configure the audit logs, view the audit logs, and to clear the audit logs (NDcPP230e:FAU_STG_EXT.1)
- The ability to manage the cryptographic functionality which allows the Authorized Administrator the ability to identify and configure the algorithms used to provide protection of the data, such as generating the RSA keys to enable SSHv2 (NDcPP30e:FCS_IPSEC_EXT.1, SSH10:FCS_SSH_EXT.1)
- The ability to manage cryptographic keys (NDcPP30e:FIA_X509_EXT.3)
- The ability to configure the authentication failure parameters for FIA_AFL.1. (NDcPP30e:FIA_AFL.1)
- The ability to import the X.509v3 certificates to the TOE's trusted store. (NDcPP30e:FCS_IPSEC_EXT.1 and NDcPP30e:FIA_X509_EXT.3)
- The ability to configure the reference identifier for the peer. (NDcPP30e:FCS_IPSEC_EXT.1.14)
- The ability to manually unlock a locked administrator account. (NDcPP30e:FIA_AFL.1)
- The ability to start and stop services. (NDcPP30e:FAU_GEN.1)
- The ability to modify the behaviour of the transmission of audit data to an external IT entity. (NDcPP30e:FAU_STG_EXT.1)
- The ability to configure NTP. (NDcPP30e:FCS_NTP_EXT.1)
- The ability to configure thresholds for SSH rekeying. (NDcPP30e:FCS_SSHS_EXT.1.8)
- The ability to configure the lifetime for IPsec SAs. (NDcPP30e:FCS_IPSEC_EXT.1.7)

- The ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors. (NDcPP30e:FIA_X509_EXT.3)
- The ability to manage the trusted public keys database. (NDcPP30e: FIA_UIA_EXT.1)
- The ability to define packet filtering rules; (VPNGW13:FPF_RUL_EXT.1)
- The ability to associate packet filtering rules to network interfaces; (VPNGW13:FPF_RUL_EXT.1)
- The ability to order packet filtering rules by priority. (VPNGW13:FPF_RUL_EXT.1)

Component Guidance Assurance Activities: See TSS Assurance Activities

See TSS Assurance Activities.

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

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

For a mapping of these functions to the SFRs where they were tested, see the TSS Assurance Activity response above.

2.4.7 Specification of Management Functions (VPNGW13:FMT_SMF.1/VPN)

2.4.7.1 VPNGW13:FMT_SMF.1.1/VPN

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 confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the TSS identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

Section 6.3 of the [ST], FMT_SMF.1, provides a description of the management functions of the TOE. The required VPN functions are included in the list of functions provided by the TOE.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the operational guidance identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

See the NDcPP30e:FMT_SMF.1 TSS Assurance Activity. All of the Module specified management functions are enumerated there.

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

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

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

2.4.8.1 NDcPP30e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.8.2 NDcPP30e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.8.3 NDcPP30e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FMT_SMR.2, states the TOE platform maintains privileged and semi-privileged administrator roles. The TOE performs role-based authorization, using TOE platform authorization mechanisms, to grant access to the semi-privileged and privileged roles. For the purposes of this evaluation, the privileged role is equivalent to full administrative access to the CLI, which is the default access for IOS-XE privilege level 15; and the semi-privileged role equates to any privilege level that has a subset of the privileges assigned to level 15. Privilege levels 0 and 1 are defined by default and are customizable, while levels 2-14 are undefined by default and are also customizable. Note: the levels are not hierarchical.

The term “Security Administrator” is used in this ST to refer to any user which has been assigned to a privilege level that is permitted to perform the relevant action; therefore, has the appropriate privileges to perform the requested functions.

The privilege level determines the functions the user can perform; hence the Security Administrator with the appropriate privileges. Refer to the Guidance documentation and IOS-XE Command Reference Guide for available commands and associated roles and privilege levels.

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

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

Section 3.2 of the Admin Guide discusses local administration. It explains how to create user accounts and set passwords. Section 3.3.1 explains how to setup remote administration and explains how to set parameters to meet the [ST] claims.

Component Testing Assurance Activities: In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH, if the TSF shall be validated against the Functional Package for Secure Shell referenced in Section 2.2 of the cPP; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

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

2.5 Packet Filtering (FPF)

2.5.1 Packet Filtering Rules (VPNGW13:FPF_RUL_EXT.1)

2.5.1.1 VPNGW13:FPF_RUL_EXT.1.1

TSS Assurance Activities: The evaluator shall verify that the TSS provide a description of the TOE's initialization/startup process, which clearly indicates where processing of network packets begins to take place, and provides a discussion that supports the assertion that packets cannot flow during this process.

The evaluator shall verify that the TSS also includes a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describes the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets.

Section 6.3 of the [ST], FPF_RUL_EXT.1, states the packet filtering rules are operational as soon as interfaces are operational following startup of the TOE. There is no state during initialization/startup that the access lists are not enforced on an interface. The initialization process first initializes the operating system, and then the networking daemons including the access list enforcement, prior to any daemons or user applications that potentially send network traffic. No incoming network traffic can be received before the access list functionality is operational. Whenever a failure occurs within the TOE that results in the TOE ceasing operation, the TOE securely disables its interfaces to prevent the unintentional flow of any information to or from the TOE.

Guidance Assurance Activities: The operational guidance associated with this requirement is assessed in the subsequent test EAs.

See the test evaluation activities.

Testing Assurance Activities: Test 1: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would otherwise be denied by the ruleset should be sourced and directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization.

Test 2: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would be permitted by the ruleset should be sourced and directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization and is only permitted once initialization is complete.

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test Evaluation Activities.

The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. Using guidance, the evaluator configured the TOE to block network traffic directed at a specific destination. The evaluator started transmitting packets from a test server. The packets being sent were constructed such that they should be blocked by the configure rule. The evaluator rebooted the TOE and monitored traffic on both sides of the TOE during the reboot process. The evaluator observed that while packets were being delivered to the TOE ingress side, no matching packets were being passed from the egress side.

2.5.1.2 VPNGW13:FPF_RUL_EXT.1.2

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet piltering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

There are no EAs specified for this element. See VPNGW13:FPF_RUL_EXT.1.4.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.3 VPNGW13:FPF_RUL_EXT.1.3

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

There are no EAs specified for this element. See VPNGW13:FPF_RUL_EXT.1.4.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.4 VPNGW13:FPF_RUL_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes a packet filtering policy that can use the following fields for each identified protocol, and that the RFCs identified for each protocol are supported:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

The evaluator shall verify that the TSS describes how conformance with the identified RFCs has been determined by the TOE developer (e.g., third party interoperability testing, protocol compliance testing).

The evaluator shall verify that each rule can identify the following actions: permit, discard, and log.

The evaluator shall verify that the TSS identifies all interface types subject to the Packet Filtering policy and explains how rules are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used), they can be treated collectively as a distinct network interface.

Section 6.3 of the [ST], FPF_RUL_EXT.1, explains an authorized administrator can define the traffic that needs to be protected by configuring access lists (permit, deny, log) and applying these access lists to interfaces using the 'ip access-group' command. Therefore, traffic may be selected on the basis of the source and destination address, and optionally the Layer 4 protocol and port. The access lists can be applied to all the network interfaces.

The TOE enforces information flow policies on network packets that are received by TOE interfaces and leave the TOE through other TOE interfaces. When network packets are received on a TOE interface, the TOE verifies whether the network traffic is allowed or not and performs one of the following actions, pass/not pass information, as well as optional logging.

By implementing rules that defines the permitted flow of traffic between interfaces of the TOE for unauthenticated traffic. These rules control whether a packet is transferred from one interface to another based on:

1. presumed address of source
2. presumed address of destination
3. transport layer protocol (or next header in IPv6)
4. Service used (UDP or TCP ports, both source and destination)
5. Network interface on which the connection request occurs

These rules are supported for the following protocols: RFC 791(IPv4); RFC 8200 (IPv6); RFC 793 (TCP); RFC 768 (UDP). TOE compliance with these protocols is verified via regular quality assurance, regression, and interoperability testing.

Guidance Assurance Activities: The evaluators shall verify that the operational guidance identifies the following protocols as being supported and the following attributes as being configurable within packet filtering rules for the associated protocols:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

The evaluator shall verify that the operational guidance indicates that each rule can identify the following actions: permit, discard, and log.

The evaluator shall verify that the operational guidance explains how rules are associated with distinct network interfaces.

The guidance may describe the other protocols contained within the ST (e.g., IPsec, IKE, potentially HTTPS, SSH, and TLS) that are processed by the TOE. The evaluator shall ensure that it is made clear what protocols were not considered as part of the TOE evaluation.

Section 3.3.7 of the Admin Guide identifies the protocols that are addressed by the evaluation. Packet filtering is able to be done on many protocols by the TOE, including but not limited to (although the evaluation only covers IPv4, IPv6, TCP and UDP):

- IPv4 (RFC 791)
- IPv6 (RFC 2460)
- TCP (RFC 793)
- UDP (RFC 768)
- IKEv1(RFCs 2407, 2408, 2409, RFC 4109)
- IKEv2 (RFC 5996)
- IPsec ESP (RFCs 4301, 4303)

- SSH(RFCs4251,4252,4253,and4254)

The following attributes, at a minimum, are configurable within Packet filtering rules for the associated protocols:

- IPv4
  - o Source address
  - o Destination Address
  - o Protocol
- IPv6
  - o Source address
  - o Destination Address
  - o Next Header (Protocol)
- TCP
  - o Source Port
  - o Destination Port
- UDP
  - o Source Port
  - o Destination Port

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

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet filter rules can be created that permit, discard, and log packets for each of the following attributes:

- IPv4

o Source address

o Destination Address

o Protocol

- IPv6

o Source Address

o Destination Address

o Next Header (Protocol)

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

Test 2: The evaluator shall repeat Test 1 above for each distinct network interface type supported by the TOE to ensure that Packet filtering rules can be defined for each all supported types.

Note that these test activities should be performed in conjunction with those of FPF_RUL_EXT.1.6 where the effectiveness of the rules is tested; here the evaluator is just ensuring the guidance is sufficient and the TOE supports the administrator creating a ruleset based on the above attributes. The test activities for FPF_RUL_EXT.1.6 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfill the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

Test 1 & 2: These tests are performed in conjunction with FPF_RUL_EXT.1.6, which incorporates numerous variations of packet filtering rules that demonstrate proper enforcement of the packet filtering ruleset (e.g., permit, deny, and log rules, ICMPv*, IPv4 and IPv6, TCP and UDP, numerous ports, source & destination differences, and transport protocols).

2.5.1.5 VPNGW13:FPF_RUL_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset.

Section 6.3 of the [ST], FPF_RUL_EXT.1, explains that the TOE is capable of inspecting network packet header fields to determine if a packet is part of an established session or not. ACL rules still apply to packets that are part of an ongoing session.

Packets will be dropped unless a specific rule has been set up to allow the packet to pass (where the attributes of the packet match the attributes in the rule and the action associated with the rule is to pass traffic). This is the default action that occurs on an interface if no ACL rule is found. If a packet arrives that does not meet any rule, it is expected to be dropped. Rules are enforced on a first match basis from the top down. As soon as a match is found the action associated with the rule is applied.

These rules are entered in the form of access lists at the CLI (via 'access list' and 'access group' commands). These interfaces reject traffic when the traffic arrives on an external TOE interface, and the source address is an external IT entity on an internal network;

These interfaces reject traffic when the traffic arrives on an internal TOE interface, and the source address is an external IT entity on the external network;

These interfaces reject traffic when the traffic arrives on either an internal or external TOE interface, and the source address is an external IT entity on a broadcast network;

These interfaces reject traffic when the traffic arrives on either an internal or external TOE interface, and the source address is an external IT entity on the loopback network;

These interfaces reject requests in which the subject specifies the route for information to flow when it is in route to its destination.

Otherwise, these interfaces pass traffic only when its source address matches the network interface originating the traffic through another network interface corresponding to the traffic's destination address.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes how the order of Packet filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing.

Section 3.3.7 of the Admin Guide states the traffic matching is done based on a top-down approach in the access list. The first entry that a packet matches will be the one applied to it. The VPNGW Module requires that the TOE Access control lists (ACLs) are to be configured to drop all packet flows as the default rule and that traffic matching the ACL be able to be logged. The drop all default rule can be achieved by including an ACL rule to drop all packets as the last rule in the ACL configuration. The logging of matching traffic is done by appending the key word "log-input" per the command reference at the end of the ACL statements.

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

Test 1: The evaluator shall devise two equal Packet Filtering rules with alternate operations â€“ permit and discard. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.

Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

Test 1: The evaluator configured the TOE (according to the admin guide) with two packet filtering rules using the same matching criteria, where one rule would permit while the other deny traffic. Packets matching the ACL entry rule were sent through the TOE and the evaluator observed that the action taken by the TOE matched the action specified by the first ACL entry.

Test 2: Continuing test 1, the evaluator repeated the procedure above, except the evaluator changed the rules to make one a subset of the other, and then tested both orders to ensure that the first is enforced regardless of the specificity of the rule.

2.5.1.6 VPNGW13:FPF_RUL_EXT.1.6

TSS Assurance Activities: The evaluator shall verify that the TSS describes the process for applying Packet Filtering rules and also that the behavior (either by default, or as configured by the administrator) is to discard packets when there is no rule match. The evaluator shall verify the TSS describes when the IPv4/IPv6 protocols supported by the TOE differ from the full list provided in the RFC Values for IPv4 and IPv6 table.

For the first part of the AA, See FPF_RUL_EXT.1.5.

Section 6.3, FPF_RUL_EXT.1 of the [ST] also contains a description of the IPv4 protocol (2) and IPv6 protocols (43, 44, 51, 60, & 135) that are not supported by the TOE.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the operational guidance provides the appropriate instructions to configure the behavior to discard packets with no matching rules. The evaluator shall verify that the operational guidance describes the range of IPv4 and IPv6 protocols supported by the TOE.

Section 3.3.7 of the Admin Guide states the drop all default rule can be achieved by including an ACL rule to drop all packets as the last rule in the ACL configuration. This same section also lists the only unsupported IPv4 protocol (2) and IPv6 protocols (43, 44, 51, 60, & 135).

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

Test 1: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 2: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 3: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 4: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 5: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 6: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 7: The evaluator shall configure the TOE to permit and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

Test 8: The evaluator shall configure the TOE to discard and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 9: The evaluator shall configure the TOE to permit and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Here the evaluator ensures that the UDP port 500 (IKE) is included in the set of tests.

Test 10: The evaluator shall configure the TOE to discard and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are denied (i.e., by

capturing no applicable packets passing through the TOE) and logged. Again, the evaluator ensures that UDP port 500 is included in the set of tests.

The following table identifies the RFC defined values for the protocol fields for IPv4 and IPv6 to be used in configuring and otherwise testing Packet Filtering rule definition and enforcement:

Test 1: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured. The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv4 Protocol permitted and logged based on specific source and destination addresses.
- IPv4 Protocol permitted and logged based on specific destination addresses.
- IPv4 Protocol permitted and logged based on specific source addresses.
- IPv4 Protocol permitted and logged based on wildcard addresses.

Test 2: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv4 Protocol all permitted with some denied and logged based on specific source and destination addresses.
- IPv4 Protocol all permitted with some denied and logged based on specific destination addresses.
- IPv4 Protocol all permitted with some denied and logged based on specific source addresses.
- IPv4 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 3: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv4 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on specific destination addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on specific source addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on wildcard addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 4: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv6 Protocol permitted and logged based on specific source and destination addresses.
- IPv6 Protocol permitted and logged based on specific destination addresses.
- IPv6 Protocol permitted and logged based on specific source addresses.
- IPv6 Protocol permitted and logged based on wildcard addresses.

Test 5: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv6 Protocol all permitted with some denied and logged based on specific source and destination addresses.
- IPv6 Protocol all permitted with some denied and logged based on specific destination addresses.
- IPv6 Protocol all permitted with some denied and logged based on specific source addresses.
- IPv6 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 6: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv6 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on specific destination addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on specific source addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on wildcard addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 7: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- TCP (IPv4) permitted and logged based on source port.

- TCP (IPv4) permitted and logged based on destination port.
- TCP (IPv4) permitted and logged based on source and destination port.
- TCP (IPv6) permitted and logged based on source port.
- TCP (IPv6) permitted and logged based on destination port.
- TCP (IPv6) permitted and logged based on source and destination port.

Test 8: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- TCP (IPv4) denied and logged based on source port.
- TCP (IPv4) denied and logged based on destination port.
- TCP (IPv4) denied and logged based on source and destination port.
- TCP (IPv6) denied and logged based on source port.
- TCP (IPv6) denied and logged based on destination port.
- TCP (IPv6) denied and logged based on source and destination port.

Test 9: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- UDP (IPv4) permitted and logged based on source port.
- UDP (IPv4) permitted and logged based on destination port.
- UDP (IPv4) permitted and logged based on source and destination port.
- UDP (IPv6) permitted and logged based on source port.
- UDP (IPv6) permitted and logged based on destination port.
- UDP (IPv6) permitted and logged based on source and destination port.

Test 10: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- UDP (IPv4) denied and logged based on source port.
- UDP (IPv4) denied and logged based on destination port.
- UDP (IPv4) denied and logged based on source and destination port.
- UDP (IPv6) denied and logged based on source port.
- UDP (IPv6) denied and logged based on destination port.
- UDP (IPv6) denied and logged based on source and destination port.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6 Protection of the TSF (FPT)

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

2.6.1.1 NDcPP30e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.1.2 NDcPP30e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password

data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Section 6.3 of the [ST], FPT_APW_EXT.1, states the TOE includes CLI command features that can be used to configure the TOE to encrypt all locally defined user passwords. In this manner, the TOE ensures that plaintext user passwords will not be disclosed even to administrators. The password is encrypted by using the command "password encryption aes" used in global configuration mode. The “password encryption aes” command enables the functionality and the “key config-key password-encrypt” command is used to set the master password to be used to encrypt the preshared keys.

The command *service password-encryption* applies encryption to all passwords, including username passwords, authentication key passwords, the privileged command password, console and virtual terminal line access passwords.

Additionally, enabling the ‘hidekeys’ command in the logging configuration ensures that and passwords are not displayed in plaintext.

The TOE includes a Master Passphrase feature that can be used to configure the TOE to encrypt all locally defined user passwords using AES. In this manner, the TOE ensures that plaintext user passwords will not be disclosed even to administrators.

Password encryption is configured using the ‘service password-encryption’ command. There are no administrative interfaces available that allow passwords to be viewed as they are encrypted via the password-encryption service.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.2 Failure with Preservation of Secure State (Self-Test Failures) (VPNGW13:FPT_FLS.1/SelfTest)

2.6.2.1 VPNGW13:FPT_FLS.1.1/SelfTest

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes how the TOE ensures a shutdown upon a self-test failure, a failed integrity check of the TSF executable image, or a failed health test of the noise source. If there are instances when a shut-down does not occur, (e.g., a failure is deemed non- security relevant), the evaluator shall ensure that those cases are identified and a rationale is provided that supports the classification and justifies why the TOE's ability to enforce its security policies is not affected in any such instance.

Section 6.3 of the [ST], FPT_FLS.1, states that whenever a failure occurs within the TOE that results in the TOE ceasing operation, the TOE securely disables its interfaces to prevent the unintentional flow of any information to or from the TOE. The TOE shuts down by reloading and will continue to reload as long as the failures persist. This functionally prevents any failure of power-on self-tests, failure of integrity check of the TSF executable image, failure of noise source health tests from causing an unauthorized information flow. There are no failures that circumvent this protection.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides information on the self-test failures that can cause the TOE to shut down and how to diagnose the specific failure that has occurred, including possible remediation steps if available.

Section 7 of the Admin Guide provides a summary of the self-tests. This list matches the [ST]. After the list of tests, the Admin Guide explains what to do if any of the tests fail.

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

Component Testing Assurance Activities: None Defined

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

2.6.3.1 NDcPP30e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FPT_SKP_EXT.1, states the TOE stores all private keys in a secure directory protected from access as there is no interface in which the keys can be accessed.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.4 Reliable Time Stamps (NDcPP30e:FPT_STM_EXT.1)

2.6.4.1 NDcPP30e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.4.2 NDcPP30e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the [ST], FPT_STM_EXT.1, states that the TOE provides a source of date and time information used in audit event timestamps, and for certificate validity checking. This date and time are used as the time stamp that is applied to TOE generated audit records and used to track inactivity of administrative sessions. The time information is also used in various routing protocols such as, OSPF and BGP; Calculate IKE stats (including limiting SAs based on times); determining AAA timeout, administrative session timeout, and SSH rekey.

The TOE synchronizes with an NTP server for its reliable and accurate timestamp. The TOE can be configured to support at least three (3) NTP servers. The TOE supports NTPv4 and validates the integrity of the time-source using IPsec to provide trusted communication between itself and an NTP time source. In addition, the TOE does not allow the timestamp to be updated from broadcast addresses. NTP use UTC to synchronize computer clock times to a millisecond, and sometimes to a fraction of a millisecond.

The TOE does not obtain its time from an underlying VS.

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

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

Section 4.3 of the Admin Guide explains how to set the time on the TOE and also specifies how to configure an NTP server. This section also explains that NTP is protected by IPsec, the configuration of which is described earlier in the Admin Guide.

The TOE does not obtain its time from an underlying VS.

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

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

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

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

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

Test 1: The evaluator logged in as a Security Administrator and followed the [AGD] instructions to manually set the time. The evaluator observed that the time was set to the correct date/time value.

Test 2: In testing for FCS_NTP_EXT.1, the evaluator followed the [AGD] to configure the TOE's NTP client for communication with an external NTP server. The evaluator observed in this same test that the TOE was able to establish a connection to the NTP server and sync the time correctly. The TOE only supports NTPv4.

Test 3 is not applicable, as the TOE does not obtain its time from an underlying VS.

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

2.6.5.1 NDCPP30E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.5.2 NDCPP30E:FPT_TST_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. If more than one failure response is listed in FPT_TST_EXT.1.2, the evaluator shall examine the TSS to ensure it clarifies which response is associated with which type of failure.

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

(TD0836 applied)

Section 6.3 of the [ST], FPT_TST_EXT.1, explains the self-tests. The TOE runs a suite of self-tests during initial start-up to verify its correct operation. Refer to the FIPS Security Policy for available options and management of the cryptographic self-test. For testing of the TSF, the TOE automatically runs checks and tests at startup and during resets and periodically during normal operation to ensure the TOE is operating correctly, including checks of image integrity and all cryptographic functionality.

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

- Noise Source Health Tests –

The Noise Source Health Tests check the functioning of the Noise Source that supplies randomness to the Entropy Source. The tests are designed to detect failure of the Noise Source. These tests are run at startup and continuously during normal operation.

- AES Known Answer Test –

For the encrypt test, a known key is used to encrypt a known plain text value resulting in an encrypted value. This encrypted value is compared to a known encrypted value to ensure that the encrypt operation is working correctly. The decrypt test is just the opposite. In this test a known key is used to decrypt a known encrypted value. The resulting plaintext value is compared to a known plaintext value to ensure that the decrypt operation is working correctly.

- RSA Signature Known Answer Test (both signature/verification) –

This test takes a known plaintext value and Private/Public key pair and used the public key to encrypt the data. This value is compared to a known encrypted value to verify that encrypt operation is working properly. The encrypted data is then decrypted using the private key. This value is compared to the original plaintext value to ensure the decrypt operation is working properly.

- RNG/DRBG Known Answer Test –

For this test, known seed values are provided to the DRBG implementation. The DRBG uses these values to generate random bits. These random bits are compared to known random bits to ensure that the DRBG is operating correctly.

- HMAC Known Answer Test –

For each of the hash values listed, the HMAC implementation is fed known plaintext data and a known key. These values are used to generate a MAC. This MAC is compared to a known MAC to verify that the HMAC and hash operations are operating correctly.

- SHA-1/256/384/512 Known Answer Test –

For each of the values listed, the SHA implementation is fed known data and key. These values are used to generate a hash. This hash is compared to a known value to verify they match, and the hash operations are operating correctly.

- ECDSA self-test (both signature/verification) –

This test takes a known plaintext value and Private/Public key pair and used the public key to encrypt the data. This value is compared to a known encrypted value to verify that encrypt operation is working properly. The encrypted data is then decrypted using the private key. This value is compared to the original plaintext value to ensure the decrypt operation is working properly.

- Software Integrity Test –

The Software Integrity Test is run automatically whenever the IOS system images is loaded and confirms that the image file that's about to be loaded has maintained its integrity. The integrity of stored TSF executable code when it is loaded for execution can be verified through the use of RSA and Elliptic Curve Digital Signature algorithms.

If any component reports failure for the POST, the system crashes and appropriate information is displayed on the screen and saved in the crashinfo file. All ports are blocked from moving to forwarding state during the POST. If all components of all modules pass the POST, the system is placed in FIPS PASS state and ports are allowed to forward data traffic.

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

Example Error:

*Nov 26 2022 16:28:23.629: %CRYPTO-0-SELF_TEST_FAILURE: Encryption self-test failed

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

The integrity of stored TSF executable code when it is loaded for execution can be verified through the use of RSA Digital Signature algorithms.

The TOE is not distributed.

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

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

Section 7 of the Admin Guide provides a summary of the self-tests. This list matches the [ST]. After the list of tests, the Admin Guide explains what to do if any of the tests fail.

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

- If necessary, return the TOE to Cisco under guidance of Cisco Technical Assistance

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

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

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

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

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

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

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

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

(TD0836 applied)

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

2.6.6 Self-Test with Defined Methods (VPNGW13:FPT_TST_EXT.3)

2.6.6.1 VPNGW13:FPT_TST_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.6.2 VPNGW13:FPT_TST_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator verifies that the TSS describes the method used to perform self-testing on the TSF executable code, and that this method is consistent with what is described in the SFR.

Section 6.3 of the [ST], FPT_TST_EXT.3, explains the Software Integrity Test is run automatically whenever the IOS system images is loaded and confirms that the image file that's about to be loaded has maintained its integrity. The integrity of stored TSF executable code when it is loaded for execution can be verified through the use of RSA and Elliptic Curve Digital Signature algorithms.

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

All ports are blocked from moving to forwarding state during the POST. If all components of all modules pass the POST, the system is placed in FIPS PASS state and ports are allowed to forward data traffic.

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

Example Error:

*Nov 26 2023 16:28:23.629: %CRYPTO-0-SELF_TEST_FAILURE: Encryption self-test

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 behavior will be identified by the failure of a self-test.

The integrity of stored TSF executable code when it is loaded for execution can be verified through the use of RSA Signature algorithm.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.7 Trusted update (NDcPP30e:FPT_TUD_EXT.1)

2.6.7.1 NDcPP30E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.7.2 NDcPP30E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.7.3 NDcPP30E:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

For distributed TOEs, the evaluator shall examine the TSS to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update

(when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component. Alternatively, this description can be provided in the guidance documentation. In that case the evaluator should examine the guidance documentation instead.

Section 6.3 of the [ST], FPT_TUD_EXT.1 states that an Authorized Administrator can query the software version running on the TOE and can initiate updates to software images. The current active version can be verified by executing the “show version” command from the TOE’s CLI. When software updates are made available by Cisco, an administrator can obtain, verify the integrity of, and install those updates. The updates can be downloaded from the software.cisco.com.

The cryptographic hashes (i.e., SHA-512) are used to verify software update files (to ensure they have not been modified from the originals distributed by Cisco) before they are used to actually update the applicable TOE components. Authorized Administrators can download the approved image file from Cisco.com onto a trusted computer system for usage in the trusted update functionality. The hash value can be displayed by hovering over the software image name under details on the Cisco.com web site. The verification should not be performed on the TOE during the update process. If the hashes do not match, contact Cisco Technical Assistance Center (TAC).

Digital signatures and published hash mechanisms are used to verify software/firmware update files (to ensure they have not been modified from the originals distributed by Cisco) before they are used to actually update the applicable TOE components. The TOE image files are digitally signed so their integrity can be verified during the boot process, and an image that fails an integrity check will not be loaded.

To verify the digital signature prior to installation, the “show software authenticity file” command allows you to display software authentication related information that includes image credential information, key type used for verification, signing information, and other attributes in the signature envelope, for a specific image file. If the output from the “show software authenticity file” command does not provide the expected output, contact Cisco Technical Assistance Center (TAC) https://tools.cisco.com/ServiceRequestTool/create/launch.do.

Further instructions for how to do this verification are provided in the administrator guidance for this evaluation.

Software images are available from Cisco.com at the following:

<http://www.cisco.com/cisco/software/navigator.html>

The options 'support automatic checking for updates' or 'support automatic updates' are not selected.

The TOE is not distributed.

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

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

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

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

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

Section 2 of the Admin Guide, steps 8 and 9 provide instructions for how to download and verify an image prior to running it on the TOE. The [AGD] provides detailed instructions for each step in the process. It also says if the verification fails, the image will not be loaded.

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

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

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

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

ii. An image that has not been signed.

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

iv. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

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

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

Test 1: Prior to performing an update, the evaluator verified the TOE version using TOE commands. The evaluator then followed guidance to install a valid update to the TOE. Upon successful installation, the evaluator verified the TOE version once again and confirmed that the version after the successful update was changed as expected.

Test 2: The evaluator attempted to perform a TOE update using a legitimate update that was modified using a hex editor. The TOE rejected the modified update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with the digital signature removed. The TOE rejected the modified update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with the digital signature manually modified. The TOE rejected the modified update and the product version did not change.

2.7 TOE ACCESS (FTA)

2.7.1 TSF-initiated Termination (NDcPP30e:FTA_SSL.3)

2.7.1.1 NDcPP30e:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FTA_SSL.3, states the allowable inactivity timeout range is from 1 to 65535 seconds. Administratively configurable timeouts are also available for the EXEC level access (access above level 1) through use of the “exec-timeout” setting.

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

Section 3.2.5 of the Admin Guide discusses remote session termination. It provides instructions for setting termination values for inactive sessions.

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

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

The evaluator followed the guidance to configure the session timeout periods of 1 and 2 minutes for SSH CLI remote sessions. The evaluator confirmed that the session was terminated after the configured time period.

2.7.2 User-initiated Termination (NDcPP30e:FTA_SSL.4)

2.7.2.1 NDcPP30e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FTA_SSL.4, states an administrator is able to exit out of both local and remote administrative sessions. Each administrator logged onto the TOE can manually terminate their session using the "exit" or "logout" command.

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

Section 3.3.1 of the Admin Guide states that to terminate a remote or local session to the router, use the "exit" or "logout" command.

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

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

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

Test 1 & 2: Refer to the results of FIA_UIA_EXT.1-t1 where the evaluator followed the guidance instructions to exit the session at both administrative interface, local console and SSH connection.

2.7.3 TSF-initiated Session Locking (NDcPP30e:FTA_SSL_EXT.1)

2.7.3.1 NDcPP30e:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FTA_SSL.EXT.1, states an administrator can configure maximum inactivity times individually for both local and remote administrative sessions through the use of the "session-timeout" setting applied to the console. The allowable inactivity timeout range is from 1 to 65535 seconds. When a session is inactive (i.e., no session input from the administrator) for the configured period of time the TOE will terminate the session, and no further activity is allowed requiring the administrator to log in (be successfully identified and authenticated) again to establish a new session. If a remote user session is inactive for a configured period of time, the session will be terminated and will require 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 3.2.5 of the Admin Guide discusses remote session termination. It provides instructions for setting termination values for inactive sessions.

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

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

Test 1: The evaluator set the console timeout period to 1 minute and observed that the session terminated after the defined time period. This test was then repeated using a time value of 2 minutes and the same behavior was observed after each defined value.

2.7.4 Default TOE Access Banners (NDcPP30e:FTA_TAB.1)

2.7.4.1 NDcPP30e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FTA_TAB.1, states the TOE displays a privileged Administrator specified banner on the CLI management interface prior to allowing any administrative access to the TOE. This interface is applicable for both local (via console) and remote (via SSH) TOE administration.

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

Section 4.5 of the Admin Guide explains how to configure the login banner.

Component Testing Assurance Activities: The evaluator shall also perform the following test:

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

The evaluator configured a banner and verified that the banner was displayed appropriately for console and SSH CLI logins.

2.8 Trusted path/channels (FTP)

2.8.1 Inter-TSF trusted channel (NDcPP30e:FTP_ITC.1)

2.8.1.1 NDcPP30e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 NDcPP30e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.3 NDcPP30e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the [ST], FTP_ITC.1 states that the TOE protects communications with peer or neighbor routers using keyed hash as defined in FCS_COP.1/KeyedHash and cryptographic hashing functions FCS_COP.1/Hash. This protects the data from modification of data by hashing that verify that data has not been modified in transit. In addition, encryption of the data as defined in FCS_COP.1/DataEncryption is provided to ensure the data is not disclosed in transit. The TSF allows the TSF, or the authorized IT entities to initiate communication via the trusted channel.

The TOE also requires that peers and other TOE instances establish an IKE/IPSec connection in order to forward routing tables used by the TOE.

The TOE protects communications between the TOE and the remote audit server using IPsec. This provides a secure channel to transmit the log events. Likewise, communications between the TOE and AAA servers are secured using IPsec.

The TOE protects communications between itself and the NTP server using NTPv4, IPsec.

The distinction between “remote VPN peer” and “another instance of the TOE” is that “another instance of the TOE” would be installed in the evaluated configuration, and likely administered by the same personnel, whereas a “remote VPN peer” could be any interoperable IPsec gateway/peer that is expected to be administered by personnel who are not administrators of the TOE, and who share necessary IPsec tunnel configuration and authentication credentials with the TOE administrators. For example, the exchange of X.509 certificates for certificate-based authentication.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

Section 4.6 of the Admin Guide explains how to establish an IPsec connection. See FCS_IPSEC_EXT.1 for specific configuration details. Section 4.6.6 explains that if an IPsec session with a peer is unexpectedly interrupted, the connection will be broken. In these cases, no administrative interaction is required. The IPsec session will be reestablished (a new SA set up) once the peer is back online.

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

The evaluator shall perform the following tests:

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

b. Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

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

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

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a

duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

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

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

Further assurance activities are associated with the specific protocols.

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

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

The TOE utilizes IPsec for trusted channels protecting communication with an external audit server (syslog server) and an external authentication server (RADIUS).

A successful TOE IPsec connection supporting communication to an external audit server was established. Examining the packet capture from that test we see that the IPsec connection between the TOE component and the external syslog server was established; the TOE initiated the connection; and Application data transferred is encrypted (i.e., not plaintext).

The evaluator began a packet capture off traffic between the TOE and external audit sever. With the connection established, the evaluator physically disconnected the network between the TOE and the remote audit server. The evaluator left the network disconnected several minutes, and reconnected the wiring. Because the TOE automatically reconnects broken IPsec connections, the evaluator waited for the syslog server to begin receiving audit data again and stopped the packet capture shortly after traffic began flowing after the disruption. The evaluator observed that no data was transmitted unprotected.

The evaluator also used the TOE to initiate an IPsec protected communication pathway to an external authentication server. Examination of the packet capture obtained during this activity showed that the connection was protected by IPsec, the TOE initiated the connection, and all application data was transferred encrypted (i.e., not plaintext). The evaluator also performed the same physical disruption test during this test and observed that no data was transmitted unprotected.

Upon completion of these activities, the resulting transcripts and packet captures were inspected. This data showed the following:

Test 1: The TOE support for IPsec protected syslog and RADIUS was demonstrated.

Test 2: The TOE initiated the IPsec connection for both syslog and RADIUS trusted channels.

Test 3: Syslog and RADIUS communication were not sent in plaintext.

Test 4: A physical disruption in the network resulted in an IPsec session interruption and no data was transmitted unprotected upon resumption.

2.8.2 Inter-TSF Trusted Channel (VPN Communications) (VPNGW13:FTP_ITC.1/VPN)

2.8.2.1 VPNGW13:FTP_ITC.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.2 VPNGW13:FTP_ITC.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.3 VPNGW13:FTP_ITC.1.3/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

See NDcPP30e:FTP_ITC.1

Component Guidance Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

See NDcPP30e:FTP_ITC.1

Component Testing Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications. Additional evaluation testing for IPsec is covered in FCS_IPSEC_EXT.1.

See NDcPP30e:FTP_ITC.1

2.8.3 Trusted Path (NDcPP30e:FTP_TRP.1/Admin)

2.8.3.1 NDcPP30e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.2 NDcPP30e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.3 NDcPP30e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall

also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Section 6.3 of the [ST], FTP_TRP.1/Admin, states all remote administrative communications take place over a secure encrypted SSHv2 session which has the ability to be encrypted further using IPsec. The SSHv2 session is encrypted using AES encryption. The remote users are able to initiate SSHv2 communications with the TOE.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Section 3.3.1 of the Admin Guide explains how to establish an SSH connection for remote administration. See FCS_SSH_EXT.1 and FCS_SSHS_EXT.1 for specific configuration details.

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

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

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

Further assurance activities are associated with the specific protocols.

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

The TOE offers remote administration via SSHv2 or SSH tunneled through IPsec to provide the trusted path (with protection from disclosure and modification) for all remote administration sessions.

The evaluator found in FCS_SSHS_EXT1.2-t1 that the SSH connection is not sent in plaintext

The evaluator found in FTP_ITC.1-t4 that the IPsec connection is not sent in plaintext

The TOE is not distributed.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic functional specification (ADV_FSP.1)

Assurance Activities: The EAs for this assurance component focus on understanding the interfaces (e.g., application programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Explicitly labeling TSFI as security

relevant or non-security relevant is not necessary. A TSFI is implicitly security relevant if it is used to satisfy an evaluation activity, or if it is identified in the ST or guidance documentation as adhering to the security policies (as presented in the SFRs). The intent is that these interfaces will be adequately tested and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied. According to the description above 'security relevant' corresponds to the combination of 'SFR-enforcing' and 'SFR-supporting' as defined in CC Part 3, paragraph 224 and 225.

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

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

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

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

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

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

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

AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to Security Administrators and users (as appropriate) as part of the TOE.

Note that additional Evaluation Activities for the guidance documentation in the case of a distributed TOE are defined in Appendix B.4.2.1.

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR. For the related evaluation activities, the evaluation evidence documents Security Target, AGD documentation (user guidance) and functional specification documentation (if provided) shall be used as input documents. Each input document is subject to ALC_CMS.1-2 requirements.

In addition, the evaluator performs the EAs specified below.

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR. For the related evaluation activities, the evaluation evidence documents Security Target, AGD documentation (user guidance) and functional specification documentation (if provided) shall be used as input documents. Each input document is subject to ALC_CMS.1-2 requirements.

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

b. The evaluator shall verify that this process includes instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

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

Section 3.2.3 of the [AGD] states the TOE must be run in the FIPS mode of operation. The use of the cryptographic engine in any other mode was not evaluated nor tested during the CC evaluation of the TOE. Instructions for configuring algorithms are provided with each protocol.

Section 1.4 of the [AGD] provides information for the Operational Environment to be CC compliant, while Section 1.5 provides information on what functionality is excluded from the evaluation.

Section 4.7 of the [AGD] describes the verification and update process. It references Section 2 which provides step by steps instructions for installing and verifying an image.

3.2.2 Preparative procedures (AGD_PRE.1)

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

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

In addition, the evaluator performs the EAs specified below.

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

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

The evaluator shall examine the preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

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

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

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

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

The preparative procedures must

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

b. identify TOE passwords that have default values associated with them and mandate that they shall be changed.

The evaluator had the Admin Guide to use when configuring the TOE. The completeness of the Admin Guide is addressed by its use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

ALC_CMC.1-1 The evaluator shall check that the TOE provided for evaluation is labelled with its reference.

989 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

Alternatively, the unique reference provided for the TOE may be the combination of the unique reference of each component from which the TOE is comprised (e.g. in the case of a composed TOE).

ALC_CMC.1-2 The evaluator shall check that the TOE references used are consistent.

If the TOE is labelled more than once then the labels have to be consistent. For example, it should be possible to relate any labelled guidance documentation supplied as part of the TOE to the evaluated operational TOE. This ensures that consumers can be confident that they have purchased the evaluated version of the TOE, that they have installed this version, and that they have the correct version of the guidance to operate the TOE in accordance with its ST.

The evaluator also verifies that the TOE reference is consistent with the ST.

If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumers will be able to determine whether they have the appropriate items.

The evaluator verified that the [ST], TOE and Guidance are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software

versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM COVERAGE (ALC_CMS.1)

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

ALC_CMS.1-1 The evaluator shall check that the configuration list includes the following set of items:

a) the TOE itself;

b) the evaluation evidence required by the SARs in the ST.

ALC_CMS.1-2 The evaluator shall examine the configuration list to determine that it uniquely identifies each configuration item.

1103 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

See section 3.3.1 for an explanation of how all CM items are addressed.

3.4 TESTS (ATE)

3.4.1 INDEPENDENT TESTING - CONFORMANCE (ATE_IND.1)

Assurance Activities: The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2, 3 and 4.

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

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

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The test configuration consisted of the following TOE platforms along with supporting products:

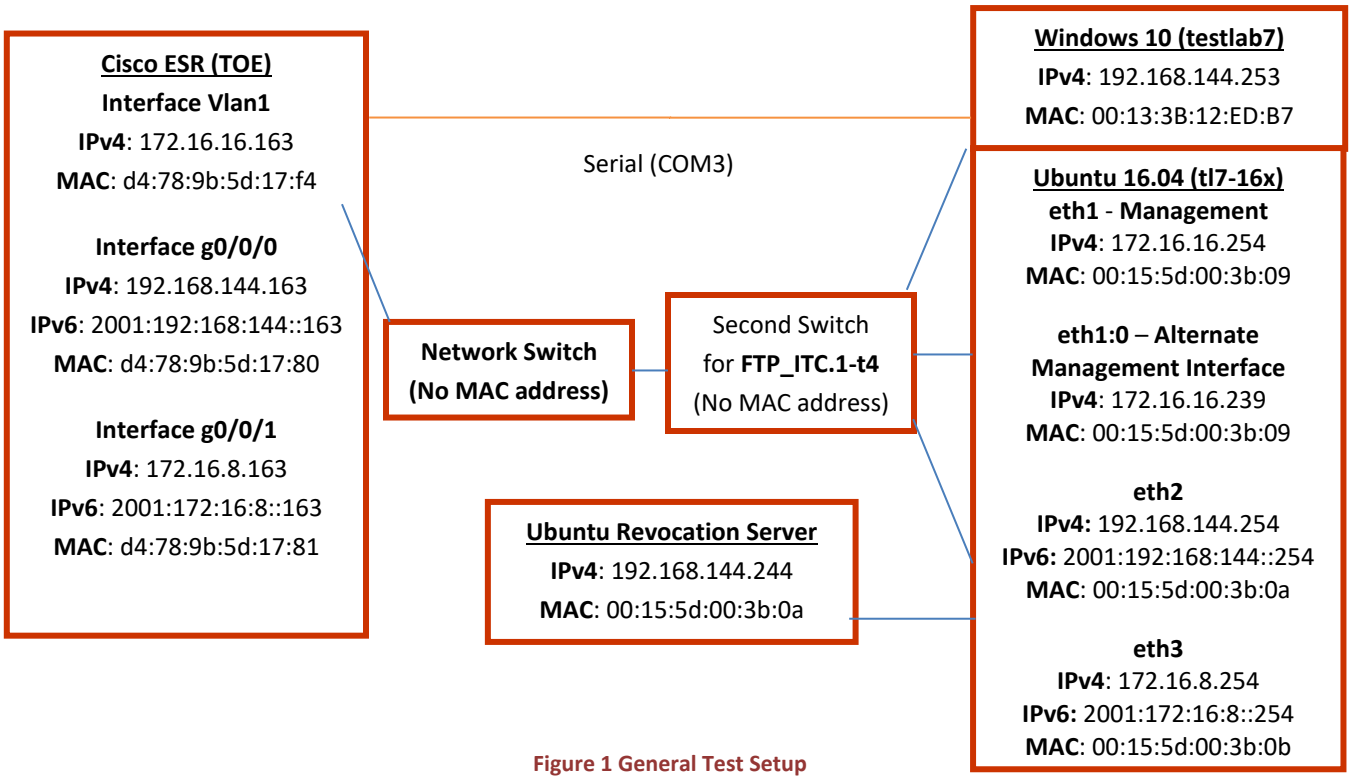

Figure 1 General Test Setup

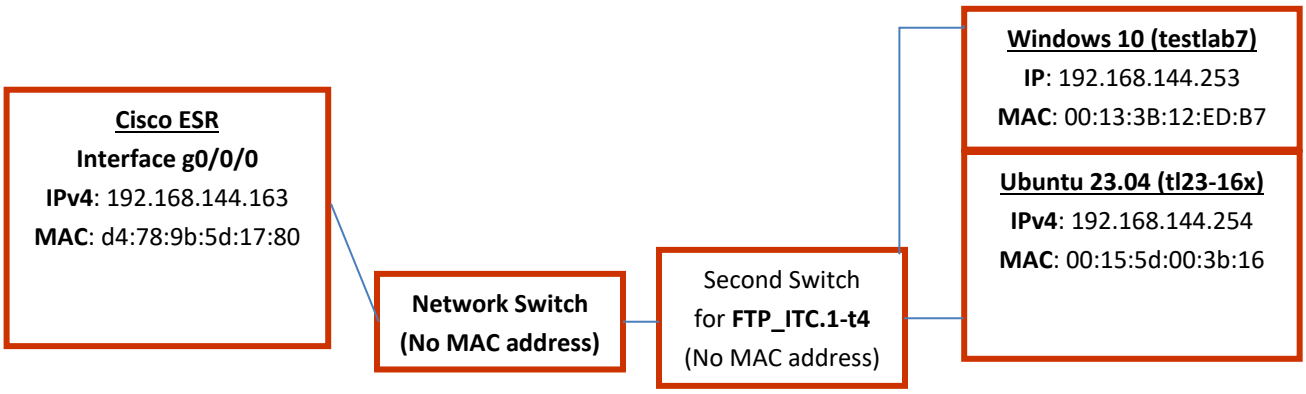

Figure 2 Alternative Ubuntu 23 Test Setup for PPKs

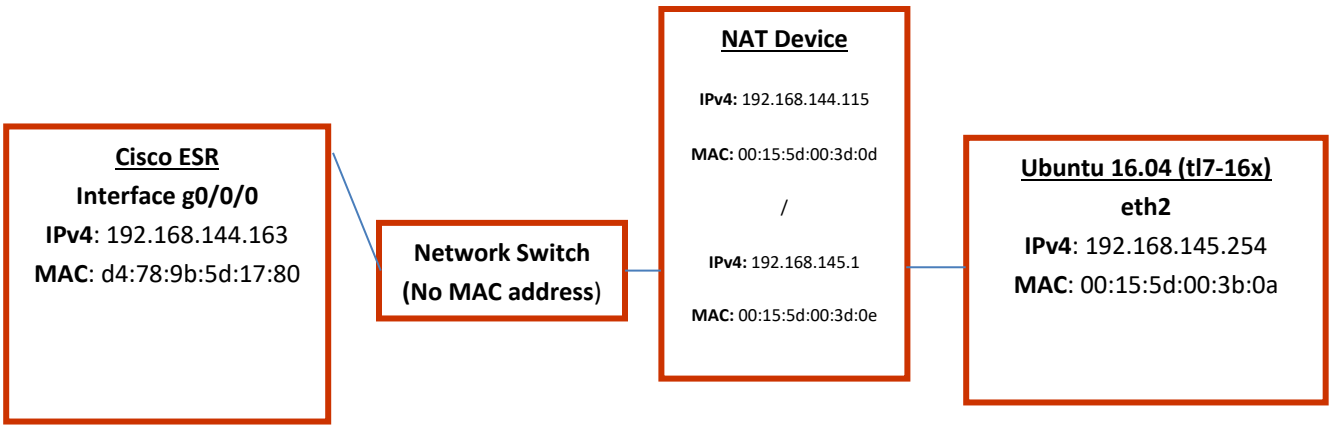

Figure 3 IPsec NAT Test Setup

TOE Platforms:

- ESR-6300-NCP-K9
- ESR-6300-CON-K9

Supporting Software:

The Gossamer Test server utilized both a Windows and Ubuntu environment.

The Windows supporting environment included the following:

- Wireshark version 4.0.5
- Putty version 0.76 (used to connect to device console and Ubuntu environment)

The Ubuntu 16 supporting environment included the following:

- Openssh-client version 7.2p2
- Big Packet Putty version 6.2
- Nmap version 7.01
- tcpdump 4.9.3
- libpcap version 1.7.4
- OpenSSL 1.0.2g
- Rsyslog 8.16.0
- strongSwan U5.3.5

The Evaluators also utilized a second Ubuntu 23 setup solely for Postquantum Pre-shared Key testing. This environment included the following:

- strongSwan U5.9.11
- libcap version 1.10.3
- tcpdump 4.99.3

As the TOE supports syncing to an NTP server, the administrator ensured the TOE and non-TOE devices were all synced to the same NTP server.

The TOE was tested on site at the Gossamer evaluation lab. The evaluation team followed the procedures in the [AGD].

Each test case in the DTR provides a Test Summary repeating the test assurance activity, Test Procedures, which are provided in abstract form and are simply a high-level summary of the actual test steps, and the Expected and

Actual results. For SSH, IPsec and X509 testing, tests are performed and actual results are generated using Gossamer's standard testing scripts which are run from the test server. These scripts perform the necessary setup procedures and generate packet captures and text files which demonstrate the actual steps performed and the output. The text files also include information regarding how to interpret the results based on the output (e.g. text strings which indicate whether the connection was successful or unsuccessful).

These text files and packet captures are provided in the Actual Results section (along with supporting text) for each relevant test case and serve as evidence of the test steps that were performed and of the actual results of the test. Each text file and packet capture is manually examined and verified by the evaluators to match the expected results.

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator shall follow a set of well-defined activities and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

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

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

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

The developer shall provide documentation identifying the list of software and hardware components[7] that compose the TOE. Hardware components should identify compute-capable hardware components, at a minimum that must include the processor, and where applicable, discrete crypto ASICs, TPMs, etc. used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic implementations, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis.

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

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

b. a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, Table 2]

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

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

The evaluator searched the 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/) on 6/26/2025 (from 6/1/2024) with the following search terms: "Cisco IOS XE", "ESR-6300-NCP-K9", "ESR-6300-CON-K9", "Marvell Armada ARMv8 Cortex A72", "IC2M", "IOS Common Cryptographic Module".

For each resulting vulnerability matching the search terms, the evaluator examined the vulnerability and determined that the vulnerability was either not applicable to the TOE, or otherwise not exploitable.

The TOE is not affected by Bleichenbacher and Klima et al. style attacks as the TOE does not support TLS.

To determine if any additional penetration testing was required, the evaluator observed the nmap scan performed in FIA_UIA_EXT.1-t2 and observed that no ports were open, beyond the ones used for SSH, which has already been tested fully. As there are no other potential ports to exploit, and all authentication mechanisms have been properly tested, it was determine that no additional penetration testing was necessary.