www.GossamerSec.com

# Assurance Activity Report for Extreme Networks Fabric Engine Switches v9.1.100

Version 0.2
05/28/26


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


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

# Revision History

| Revision | Date | Authors | Summary |
|---|---|---|---|
| Version 0.1 | 05/06/26 | Gossamer | Initial draft |
| Version 0.2 | 05/28/26 | Gossamer | ECR Comments |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**The TOE Evaluation was Sponsored by**:

## Extreme Networks, Inc.

6480 Via Del Oro, San Jose, CA 95119

**Evaluation Personnel**:

- Ryan Hagedorn
- Will Micknick
- Reed Eberly

**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 Extreme Networks Fabric Engine Switches v9.1.100 NDcPP30e/SSH10 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

## 1.1 CAVP Certificates

The TOE evaluated configuration is one instance of each of the models listed in the table below running Fabric Engine 9.1.100.

| Platform | Series | Processor |
|---|---|---|
| Fabric Engine 9.1.100 | U5320 | BCM56175, BCM56274 |
| | U5420 | BCM56274, BCM56275 |
| | U5520 | BCM56375 |
| | U5720 | Intel Atom C3338, C3538 |
| | U7520 | Intel Atom C3758 |
| | U7720 | Intel Atom C3758 |

The processors highlighted in yellow were fully tested as part of the CC evaluation. The following operational environments were tested as part of the algorithm testing:

- Fabric Engine 9, 64-bit Shared Library on Broadcom BCM56175 with ARM Cortex A72
- Fabric Engine 9, 64-bit Shared Library on Broadcom BCM56274 with ARM Cortex A72
- Fabric Engine 9, 64-bit Shared Library on Broadcom BCM56275 with ARM Cortex A72
- Fabric Engine 9, 64-bit Shared Library on Broadcom BCM56375 with ARM Cortex A72
- Fabric Engine 9, 64-bit Shared Library on Intel C3338 with AES-NI
- Fabric Engine 9, 64-bit Shared Library on Intel C3538 with AES-NI
- Fabric Engine 9, 64-bit Shared Library on Intel C3758 with AES-NI

Each of these operational environments map to a single processor found in the table above.

All cryptography is implemented using the DigiCert TrustCore Cryptographic Library v7.1.0f and the DigiCert TrustCore Cryptographic Library with GCM-64k v7.1.0f.

The following functions have been CAVP certified in accordance with the identified standards.

| Functions | Requirement | Standard | Cert # |
|---|---|---|---|
| Encryption/Decryption | | | |
| AES CBC (128 and 256 bits) | NDcPP30:FCS_COP.1/DataEncryption | FIPS Pub 197<br>ISO 10116 | A5484 |
| AES CTR (128 and 256 bits) | NDcPP30:FCS_COP.1/DataEncryption | FIPS Pub 197<br>ISO 10116 | A5484 |
| AES GCM (128 and 256 bits) | NDcPP30:FCS_COP.1/DataEncryption | NIST SP 800-38A<br>ISO 19772 | A5492 |
| Cryptographic hashing | | | |
| SHA-1, SHA-256, SHA-384, SHA-512 (digest sizes 160, 256, 384,512) | NDcPP30:FCS_COP.1/Hash | FIPS Pub 180-4<br>ISO/IEC 10118-3:2004 | A5484 |
| Keyed-hash message authentication | | | |
| HMAC-SHA-1, HMAC-SHA-256, HMAC_SHA-384 (digest sizes 160, 256, 384) | NDcPP30:FCS_COP.1/KeyedHash | FIPS Pub 198-1<br><br>FIPS Pub 180-4<br>ISO/IEC 9797-2:2011 | A5484 |
| Cryptographic signature services | | | |
| RSA Digital Signature Algorithm (rDSA) (modulus 2048) | NDcPP30:FCS_COP.1/SigGen | FIPS Pub 186-5<br>ISO/IEC 9796-2 | A5484 |
| Random bit generation | | | |
| CTR_DRBG with SW based noise sources with a minimum of 256 bits of non-determinism | NDcPP30:FCS_RBG_EXT.1 | FIPS SP 800-90A<br>ISO/IEC 18031:2011 | A5484 |
| Key Generation | | | |
| RSA Key Generation (2048 bit) | NDcPP30:FCS_CKM.1 | FIPS Pub 186-5 | A5484 |

| | | | |
|---|---|---|---|
| ECDSA Key Generation<br><br>with Curves P-256, P-384 and P-521 | | FIPS Pub 186-5 | A5484 |
| FFC Scheme DSA (2048-bit) | | FIPS Pub 186-4 | A5484 |
| FFC Scheme using Safe Primes | | NIST SP 800-56A Rev 3 | Tested with known good implementation |
| Key Establishment | | | |
| ECC Key Establishment with Curves P-256, P-384 and P-521 | NDcPP30:FCS_CKM.2 | NIST SP 800-56A Rev 3 | A5484 |
| FFC Key Establishment (2048-bit) | | NIST SP 800-56A Rev 3 | A5484 |
| FFC Schemes using safe-prime groups Diffie-Hellman Group 14 | | NIST SP 800-56A Rev 3 | Tested with known good implementation |

## 1.2 Platform Equivalence

The TOE is the Extreme Networks Fabric Engine Switches v9.1.100.

The evaluation includes the 6 series (5300, 5400, 5500, 5700, 7500, and 7700) with the identified processors. The processors highlighted in yellow were the models performed in testing.

| Platform | Series | Processor |
|---|---|---|
| Fabric Engine 9.1.100 | U5320 | BCM56175, BCM56274 |
| | U5420 | BCM56274, BCM56275 |
| | U5520 | BCM56375 |
| | U5720 | Intel Atom C3338, C3538 |
| | U7520 | Intel Atom C3758 |
| | U7720 | Intel Atom C3758 |

The lab performed all testing on one model from each series. The processors highlighted in yellow in the table above identify the processor of the model used for testing. The different hardware models support different interfaces which effect network performance, but do not affect security functionality. Each series has its own unique image but all models of the same series, regardless of processor execute the same image. All Security Functionality) is implemented in the software image which leverages the DigiCert TrustCore Cryptographic Library v7.1.0f and the DigiCert TrustCore Cryptographic Library with GCM-64k v7.1.0f. All identified processors were covered by algorithm testing.

## 1.3 References

The following evidence was used to complete the Assurance Activities:

- Extreme Networks Fabric Engine Switches v9.1.100 Security Target, version 0.5, May 28. 2026 [**ST**]
- Extreme Fabric Engine Common Criteria Configuration Guide 9.1.100, May 2026 [**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 6.1 of [ST] explains that for cryptographic keys, the act of importing, exporting or deleting a key is audited the key is identified by name and the associated administrator account is identified.

The TOE is not distributed.

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

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

The section entitled "Audit Record Samples" in [CC-Guide] contains a table of sample audit records associated with each auditable event identified by the Security Target.

This information includes details about the audit records which the TOE generates including details encompassing the required content. During testing, the evaluator mapped the entries in the tables in this section to the TOE generated events, showing that the section provides examples/descriptions of all required audit events.

The evaluator verified the administrative commands when performing all other guidance AA. Specific references to commands can be found throughout this AAR.

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

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

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

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

### 2.1.2 User identity association (NDcPP30e:FAU_GEN.2)

#### 2.1.2.1 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 where the activities for FAU_GEN.2 are already covered.

**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.

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

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

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

See FAU_GEN.1.

### 2.1.3 Protected Audit Event Storage (NDcPP30e:FAU_STG_EXT.1)

#### 2.1.3.1 NDcPP30e:FAU_STG_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.3.2 NDcPP30e:FAU_STG_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.3.3 NDcPP30e:FAU_STG_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.3.4 NDcPP30e:FAU_STG_EXT.1.4

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.3.5 NDcPP30e:FAU_STG_EXT.1.5

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.3.6 NDcPP30e:FAU_STG_EXT.1.6

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

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

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

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

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

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

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

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

Section 6.1 of [ST] explains that the TOE protects communications with this external syslog server using an encrypted via TLS over TCP (RFC 5425) session. Once a syslog server has accepted the TLS connection from the TOE, the TOE pushes new audit logs to the syslog server over the TLS protected channel in real time. The audit records are transferred as they are generated.

The TOE is a standalone device that saves its local internal audit log files in non-volatile memory within log files, where it does not overwrite older records. The TOE stops generating new audit records when audit storage hits the threshold of 75% or 90% depending on the device. The 5320 and 5420 have a threshold of 90%, the other devices have a threshold of 75%; the minimum amount of data stored is 64KB. Only Security Administrators are able to clear the local logs using CLI commands. When an Security Administrator clears local log files to make space available, the TOE automatically resumes auditing. When the TOE is not writing to local internal audit log files it will still attempt to send audit records generated to a connected syslog server.

The TOE is not distributed.

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

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

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

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

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

The section entitled "Enable a TLS Connection to the Syslog Server" in [CC-Guide] provides the steps necessary to configure a TLS protected channel for use in transferring audit data to an external audit server (syslog server). This section provides the command to configure the IP address and TLS port where the TOE will attempt to find the syslog server. The section entitled "Certificate Management" and its subsections describe the commands to import, display and remove the trust anchor for the syslog server. The section entitled "Audit Logs and Syslog" explains that the transmission of audit logs to the external audit server occurs in real time, with each audit record transferred as it is generated.

The section entitled "Enable a TLS Connection to the Syslog Server" states the Fabric Engine switch communicate with an external syslog (audit) server by establishing a trusted channel between itself and the audit server. This statement defines a requirement on the audit server as being able to support 'syslog' communication. The next paragraph indicates that the trusted channel employs TLSv1.2, which is interpreted as another requirement on the audit server. These statements indicate that a Fabric Engine Switch can communicate with an audit server supporting the syslog and TLSv1.2 protocols.

The section entitled "Log Files" includes a discussion of how the audit files can fill up available flash storage. Files are stored locally, with maximum capacity based on current available hard-drive space. You should free disk space on the flash if the system generates an alarm based on system thresholds, which are between 75% and 90%:

• 5320 and 5420 devices have a threshold of 90%.

• Other devices have a threshold of 75%.

After disk space usage returns below the device threshold, the system clears the alarm, and then starts logging to a file again.

The section entitled "Clear Log Messages and Files" contains a command to clear the log memory associated with logging. Individual log files must be removed using commands to delete the actual log file(s).

The storage size of the local audit data files are not configurable and no selections are made for FAU_STG_EXT.1.5. Therefore, neither apply.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Test 1: The evaluator configured the system (per guidance) to securely transfer audit data. 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. The evaluator also verified that the data was successfully transferred to the audit server and recorded the software (name and version) used on the audit server during testing. Once configured no further administrative action was required to cause the TOE to transfer audit data.

Test 2: Not Applicable. The TOE is not distributed

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

Test 4: The evaluator logged onto the TOE and showed the initial state of the TOE's local audit storage including total disk space consumed and a sample of the current log file verifying that logging was occurring. The evaluator then used scp to upload a large .txt file onto the TOE to exceed its local audit storage. The evaluator then logged back onto the TOE, repeating the initial commands verifying that logging has stopped, the number of lost logs was recorded, and no new logs beyond the "stop logging to file" audit has been dedicated to the current log file. The evaluator then removed the large file from the TOE and after a reboot verified that logging has resumed using the same commands.

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

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

## 2.2 Cryptographic support (FCS)

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

#### 2.2.1.1 NDcPP30e:FCS_CKM.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

In section 6.2 in [ST], The TSF supports RSA key generation scheme using cryptographic key size of 2048-bit that meet the FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.3; standard. The RSA algorithm implementation is provided by the included Mocana cryptographic library. The TSF also supports ECDSA (appendix B.4) and FFC key generation (appendix B.1). RSA key pairs can be generated during the creation of a Certificate Signing Request (CSR). The TOE follows NIST SP 800-56A Rev 3 ‘Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography’ key agreement scheme. The TOE acts as a sender of secret keying material for RSA key establishment.

The usage for each scheme is outlined in Table 6-2 Fabric Engine Key Establishment Schemes of the ST.

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

The TOE supports RSA key generation to create SSH Host keys and a key-pair for an x509 certificate.

The TOE supports RSA key generation as described by the section entitled "Enable RSA Authentication and Generate the Host Key". This section includes commands that create an RSA key to be used as the TOE host key. The section entitled "Generate the Key Pair" includes instructions to create a key-pair for an x509v3 Certificate that can be used as the TOE ssh Certificate.

The TOE supports ECC and FFC key generation only in the context of TLS key exchanges. The generation of keys for TLS key exchange is not configurable and is indirectly defined by the set of ciphersuites supported by the TOE. This set of ciphersuites and key exchanges is shown in the section entitled "Supported Cryptographic Methods."

**Component Testing Assurance Activities**: Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products. Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 or FIPS PUB 186-5 RSA Schemes

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

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

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

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

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

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

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

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

To test the key generation method for the Random Provable primes method or Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. If the

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

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

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

(TD0921 applied)

Key Generation for Elliptic Curve Cryptography (ECC)

Key Generation for ECDSA Schemes

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

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

- ECC Key Pair Generation

- ECC Public Key Validation (PKV)

ECC Key Pair Generation Test

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

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

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

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

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

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

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

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

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

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

(TD0921 applied)

ECC Public Key Verification (PKV) Test

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

(TD0921 applied)

Key Generation for FIPS PUB 186-5

FIPS 186-5 Key Generation Test

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

FIPS 186-5 Key Verification Test

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

Key Generation for Finite-Field Cryptography (FFC)

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

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

- Primes q and p shall both be provable primes

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

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

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

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

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

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

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

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

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

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

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

### 2.2.2 Cryptographic Key Establishment (NDcPP30e:FCS_CKM.2)

#### 2.2.2.1 NDcPP30e:FCS_CKM.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

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

The evaluator verified that the supported key establishment schemes identified by FCS_CKM.2 correspond to the key generation schemes identified in FCS_CKM.1.1.

| Scheme | SFRs | Service |
|---|---|---|
| ECC key establishment | FCS_TLSC_EXT.1 | Remote syslog Server<br>Remote Administration |
| FFC key establishment | FCS_TLSC_EXT.1 | Remote syslog Server |
| FFC Safe-primes key establishment | FCS_SSHS_EXT.1 (DH 14, 16, and 18) | Remote Administration |

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

Key establishment schemes are configured through the definition of SSH key exchanges and TLS ciphersuites.

The section entitled "Supported Cryptographic Methods" in [CC-Guide] identifies the ciphersuites and key exchanges supported by the TOE. This section also indicates that the TOE does not offer a management operation to allow administrators to change this supported set of ciphersuites and key exchange methods.

The section entitled "Secure Shell Configuration" explains that only Diffie-Hellman-Group14-SHA1 method is approved and it is enabled by default. No extra configuration is needed or allowed.

**Component Testing Assurance Activities**: Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests for ECC and FIPS186-type. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if

supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

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

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

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

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

Validity Test

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

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

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

RSA-based key establishment

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

FFC Schemes using 'safe-prime' groups

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

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

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

### 2.2.3 Cryptographic Key Destruction (NDcPP30e:FCS_CKM.4)

#### 2.2.3.1 NDcPP30e:FCS_CKM.4.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

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

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

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

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

Section 6.2 of [ST] provides a list of the Critical Security Parameters and their storage location. It identifies SSH keys, TLS keys, and account passwords. This section includes Table 6-3 which outlines the storage location and clearing method for each key.

Each plaintext key stored in volatile memory is associated with a protocol session (SSH or TLS). In each instance of a key, after the session closes, the key is overwritten with the value “00” After the overwrite operation is complete, the TOE performs a specific "read-verify" operation to confirm that the storage space no longer contains the key. For non-volatile storage (i.e., flash), the TOE does not store any keys in plaintext form within user-accessible, non-volatile storage. When deleted from FLASH, the previous value is overwritten with random data from the TSF RBG followed by a one pass of zeros.

**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 [CC-Guide] does not indicate that the TOE has any conditions that involve delayed key destruction.

**Component Testing Assurance Activities**: None Defined

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

#### 2.2.4.1 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 6.2 of [ST] indicates that the TOE provides symmetric encryption and decryption capabilities using AES in CBC mode (128 and 256 bit key sizes), AES in CTR mode (128 and 256 bit key sizes) as well as using AES in GCM mode (128 and 256 bit key sizes). AES is implemented in support of TLS and SSH protocols.

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

The sections entitled "Enable a TLS Connection to the Syslog Server" and "Secure Shell Configuration" in [CC-Guide] includes the instructions for administrators to place the TOE into a CC compliant configuration.

**Component Testing Assurance Activities**: AES-CBC Known Answer Tests

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

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

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

# Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

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

PT = IV

else:

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

PT = CT[i-1]

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

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

AES-GCM Test

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

128 bit and 256 bit keys

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

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

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

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

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

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

AES-CTR Known Answer Tests

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

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

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

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

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

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384

ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128].

AES-CTR Multi-Block Message Test

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

AES-CTR Monte-Carlo Test

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

# Input: PT, Key

for i = 1 to 1000:

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

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

There is no need to test the decryption engine.

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

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

#### 2.2.5.1 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.

Table 6-1 in Section 6.2 of [ST] indicates that The TOE supports hashing using SHA-1, SHA-256, SHA384 and SHA-512 conforming to FIPS 180-4, Secure Hash Standard (SHS). SHS hashing is used within several services including, NTP hashing, TLS and SSH. SHA-256 is used in conjunction with RSA signatures for verification of software image integrity. The TOE also uses SHA-1, SHA-256, SHA-384 and SHA-512 hashing as part of RSA signature generation and verification services.

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

The TOE does not offer administrative commands to modify the hash sizes used by TLS and SSH protocols. The hash sizes used are a result of the TLS ciphersuites, and SSH keyed hash negotiated with the network peer by the protocol.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

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

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

#### 2.2.6.1 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 6.2 of [ST] The TOE supports keyed hash HMAC-SHA1, HMAC-SHA256, and HMAC-SHA384 conforming to ISO/IEC 9797-2:2011. Supported cryptographic key sizes: 160, 256, and 384 bits and message digest sizes: 160, 256, and 384 bits.

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

The section entitled "Secure Shell Configuration" in [CC-Guide] indicates that the TOE supports MAC ciphers HMAC-SHA1 and HMAC-SHA2-256. It explains that other algorithms must be disabled in an evaluated configuration.

The section entitled “Supported Cryptographic methods” in [CC-Guide] indicates that the TOE supports the following HMAC ciphers for TLS:

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA

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

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

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

### 2.2.7.1 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.

Table 6-1 in Section 6.2 of [ST] indicates that the TOE supports generation and verification of RSA Digital Signature Algorithm with modulus of 2048 for cryptographic signature services.

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

The section entitled "Enable RSA Authentication and Generate the Host Key" in [CC-Guide] indicates that the SSH host key is generated using a 2048-bit RSA key. The section entitled "Generate the Key Pair" indicates that the generation of key-pairs for use in an x509 certificate must also use a 2048-bit RSA key-pair.

**Component Testing Assurance Activities**: ECDSA Signature Algorithm Tests

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

Signature Generation Test

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

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

Signature Verification Test

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

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

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

RSA Signature Algorithm Tests

Signature Generation Test

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

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

a. RSASSA-PKCS1-v1.5

b. RSASSA-PSS

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

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

Signature Verification Test

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

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

a. RSASSA-PKCS1-v1.5

b. RSASSA-PSS

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

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

(TD0921 applied)

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

### 2.2.8 NTP Protocol (NDcPP30e:FCS_NTP_EXT.1)

#### 2.2.8.1 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 6.2 of [ST] indicates that the TOE implements NTPv4 protocol to synchronize with an external time servers. The TOE authenticates updates using an administrator-configured SHA1 -based message authentication. The TOE does not synchronize based on broadcast and multicast time updates. The TOE supports configuration of multiple simultaneous time servers and follows RFC 5905 algorithm to prioritize them.

**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.

The section entitled "Specify and Enable the NTP Server" in [CC-Guide] states that an administrator can configure a maximum of 10 IPv4 NTP servers and 10 IPv6 NTP servers. The section entitled "Manage NTP Authentication" in [CC-Guide] provides instructions to specify a SHA1 authentication key for each configured server.

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

The evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server and confirmed via packet capture that the TOE establishes a connection to the external NTP server using NTPv4 as claimed in the ST.

#### 2.2.8.2 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).

The section entitled "Manage NTP Authentication" in [CC-Guide] provides instructions to specify a SHA1 authentication key for each configured server.

**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 evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server in NDcPP30e:FCS_NTP_EXT.1.1-t1 using the correct authentication algorithm. For this test, the evaluator changed the NTP server configuration to demonstrate that the TOE would not synchronize if the wrong message digest algorithm was used. The evaluator observed that the TOE rejected the connection and there was no time synchronization between the TOE and the NTP server.

#### 2.2.8.3 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.

The section entitled "Limitations and requirements" in [CC-Guide] states that NTP multicast and broadcast packets are not supported.

**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 configured the TOE to get NTP time updates from the evaluator's NTP Server. The evaluator also configured the NTP server to send broadcast and multi-cast time updates such that they would be visible to the TOE. The evaluator observed that the TOE did not accept the time updates from the NTP Server.

#### 2.2.8.4 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 the TOE with 3 valid NTP connections. The evaluator changed the time on the NTP servers and observed that the TOE updated its time and synced with the three valid NTP servers.

Test 2: The evaluator kept the TOE configured with the same 3 NTP servers as in test 1. The evaluator collected network traffic while monitoring the time on the TOE while an untrusted NTP server was configured to broadcast

to the TOE. The evaluator confirmed via packet capture that the TOE ignored the NTP packets and could not update time using traditional authenticated updates with the invalid NTP server.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.2.9 Random Bit Generation (NDcPP30e:FCS_RBG_EXT.1)

#### 2.2.9.1 NDcPP30e:FCS_RBG_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.9.2 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.

Table 6-1 in Section 6.2 of [ST] identifies the TOE as a NIST-approved AES-CTR DRBG with software based noise source with a minimum of 256 bits of non-determinism in accordance with ISO/IEC 18031:2011.

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

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

The TOE does not offer any configuration for the RNG functionality.

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

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

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

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

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

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

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

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

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

### 2.2.10 SSH Protocol - per TD0909 (SSH10:FCS_SSH_EXT.1)

#### 2.2.10.1 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

In section 6.2 of [ST], The TOE uses SSH for to facilitate secure remote administrative sessions (CLI). The TOE's SSH implementation supports the strict compliance with RFCs 4251, 4252, 4253, 4254, and 4256. The evaluator confirmed the implementation is consistent with prior selections.

**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.

#### 2.2.10.2 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.

In section 6.2 of [ST] states The TOE uses SSH for to facilitate secure remote administrative sessions (CLI). The TOE's SSH implementation supports the following:

- Use of 2048-bit RSA keys in support of ssh-rsa, rsa-sha2-256, rsa-sha2-512, and x509v3-rsa2048-sha256 for public key-based authentication;

•For x509v3-rsa2048-sha256 for public key-based authentication the identity of the user must be specified in the certificate's SubjectAltName: PrincipalName field;

•For ssh-rsa public key authentication, the user must pre-load their public key into the TOE, before attempting to use their private key during an SSH authentication;

•Password based authentication

Keyboard-interactive is not selected in the ST

**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.

In the section "Secure Shell Configuration" in [CC-Guide] states support for Public Key, Password, and X.509 Digital Certificate authentication. Configuration options for these mechanisms are described in this section with references to relevant sections.

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

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

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

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

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

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

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

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

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

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

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

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

b. The evaluator shall verify that the connection fails.

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

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

Test 1b.: The evaluator referred to FIA_UIA_EXT.1_t1 and verified that an SSH session was successfully established from a client when a username associated with an X509 certificate was used for authentication. The evaluator also verified that when using the same certificate with a non-associated username the session did not establish as well as failed to establish when the original username was used with a certificate that was not associated with it.

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

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

#### 2.2.10.3 SSH10:FCS_SSH_EXT.1.3

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

In section 6.1 of [ST] states the TOE drops SSH packets greater than 35840 bytes. This is accomplished by buffering all data for a particular SSH packet transmission until the buffer limit is reached and then dropping the packet.

**Guidance Assurance Activities**: None Defined

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

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

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

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

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

(TD0732 applied)

Test 1: The evaluator established an ssh session with the TOE, then created and sent a packet to the TOE that was exactly the maximum packet size of 35840 bytes verifying the TOE accepted the packet by confirming the connection was not terminated when transmitted by the ssh client.

Test 2: The evaluator established an ssh session with the TOE, then created and sent a packet to the TOE that exceeded the maximum packet size by 1 (35841). The evaluator then confirmed the TOE dropped the packet by terminating the connection with the ssh client.

#### 2.2.10.4 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.

In section 6.2of [ST], the TOE supports Encryption algorithms aes128-ctr, aes256-ctr, aes128-cbc, aes256-cbc, AEAD_AES_128_GCM, AEAD_AES_256_GCM, aes128-gcm@openssh.com, and aes256-gcm@openssh.com to ensure confidentiality of the session.

**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.

In the section "Secure Shell Configuration" in [CC-Guide], the list of authorized mechanisms are mentioned. The section provides instructions on how to disable mechanisms not permitted by Common Criteria.

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

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

Test 1: The evaluator must ensure that only claimed algorithms and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall establish an SSH connection with a remote endpoint. The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers only the algorithms defined in the ST for the TOE for SSH connections. The evaluator shall perform one successful negotiation of an SSH connection and verify that the negotiated algorithms were included in the advertised set. If the evaluator detects that not all algorithms defined in the ST for SSH are advertised by the TOE or the TOE advertises additional algorithms not defined in the ST for SSH, the test shall be regarded as failed.

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

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

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

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

aes128-ctr

aes256-ctr

aes128-cbc

aes256-cbc

AEAD_AES_128_GCM

AEAD_AES_256_GCM

aes128-gcm@openssh.com

aes256-gcm@openssh.com

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

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

#### 2.2.10.5 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.

In section 6.1 of [ST], The TOE supports Hashing algorithm hmac-sha2-256, AEAD_AES_128_GCM, AEAD_AES_256_GCM to ensure the integrity of the session (integrity is also provided implicitly by GCM when using aes128-gcm@openssh.com and aes256-gcm@openssh.com)

**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.

In the section "Secure Shell Configuration" in [CC-Guide], the list of authorized mechanisms are mentioned. The section provides instructions on how to disable mechanisms not permitted by Common Criteria.

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

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

Test 1: The evaluator referred to FCS_SSH_EXT.1.4 and verified that hmac-sha2-256, AEAD_AES_128_GCM, AEAD_AES_256_GCM, and implicit were offered by the TOE .

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

#### 2.2.10.6 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.

In section 6.2 of [ST], The TOE uses the enforcement of diffie-hellman-group14-sha256, diffie-hellman-group16-sha512, and diffie-hellman-group18-sha512 as the only allowed key exchange methods in the implementation of SSH. The evaluator verified that these methods matched with the mechanisms selected in 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.

In the section "Secure Shell Configuration" in [CC-Guide], the list of authorized mechanisms are mentioned. The section provides instructions on how to disable mechanisms not permitted by Common Criteria.

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

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

Test 1: The evaluator referred to test FCS_SSH_EXT.1.4 and verified the TOE advertised diffie-hellman-group14-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512.

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

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

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

#### 2.2.10.7 SSH10:FCS_SSH_EXT.1.7

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

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

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.10.8 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.

In section 6.2 in [ST], The TOE initiates a rekey before 1 hour or before 1GB whichever comes first.

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

In the section "Configure the SSH Rekeying Interval" in [CC Guide] it provides instructions on configuring the data and time limits required to trigger a rekey.

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

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

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

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

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

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

Test 3: The evaluator attempted to connect to the TOE using a SSH client while transmitting two .5gb files to the TOE. Before the second file completed its transfer, the connection was rekeyed.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

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

#### 2.2.11.1 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.

In the section "Secure Shell Configuration" in [CC-Guide], the list of authorized mechanisms are mentioned. The section provides instructions on how to disable mechanisms not permitted by Common Criteria.

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

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

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

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

(TD0682 applied)

Test 1: The evaluator referred to test FCS_SSH_EXT.1.4-t1 where the evaluator verified that the TOE only advertised ssh-rsa, rsa-sha2-256, rsa-sha2-512, x509v3-rsa2048-sha256, and no other algorithms.

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

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

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

### 2.2.12.1 NDcPP30e:FCS_TLSC_EXT.1.1

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

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

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ ECDHE_RSA_WITH_AES_256_CBC_SHA384

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

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

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

As stated in the section entitled "Supported Cryptographic Methods", the TOE does not allow TLS ciphersuites to be configured. The only configuration necessary is described in the section entitled "Enable a TLS Connection to the Syslog Server" which explains how to identify the syslog server and network port used by the TOE.

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

a. Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as

part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

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

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

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

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

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

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

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

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

i. [conditional]: Perform this test only if support of TLS 1.2 is claimed. If using DHE or ECDH ciphersuites, modify the signature block in the Server's Key Exchange handshake message, and verify that the client denies the connection and no application data flows. The handshake shall be valid (e.g. the Finished message is calculated using the

modified signature), with the exception of the invalid signature. This test does not apply to ciphersuites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

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

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

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

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

iii. [conditional]: Perform this test only if support of TLS 1.3 is claimed. Send a plaintext EncryptedExtensions message from the server and verify that the handshake is not finished successfully and no application data flows. (Note: Under TLS 1.3, the EncryptedExtensions message is the first message to be encrypted with the handshake traffic secret.)

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

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

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

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

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

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

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

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

Test 5ii: Not Applicable. TLS 1.3 is not claimed.

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

Test 6iii: Not Applicable. TLS 1.3 is not claimed.

Test 6iv: The evaluator configured the TOE to connect to a test server using TLS with a TOE supported ECDHE key exchange method. The evaluator also configured the test server to accept that same ECHDE key exchange method, but to require a curve that was not supported by the TOE (i.e., P-192). The evaluator then observed the TOE reject negotiation.

#### 2.2.12.2 NDcPP30e:FCS_TLSC_EXT.1.2

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

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

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

Section 6.2 of [ST] indicates that the TOE supports X509v3 certificates following format defined by RFC 5280 during TLS negotiations. The reference identifier configured on the TOE must be either a hostname/FQDN or an IPv4 address. The following identifiers are supported in CN: IPv4 address or a hostname. The following identifiers are supported in SAN: FQDN, IPv4 address. Wildcards are supported in the CN with a hostname or in the SAN with a FQDN identifier.

Section 6.2 of [ST] indicates that the TOE does support IP addresses in the CN, and does describe conversion of IP addresses, it further indicates that canonical format is enforced per RFC 3986. This section states that when the presented identifier in the CN is an IPv4 address, the TOE converts the string to a binary representation of an IPv4 address in network byte order. If there is not an exact binary match, then the verification fails. The TOE expects IPv4 identifier to follow the RFC 3986 defined canonical format, if any unexpected special characters or extra numbers are encountered, the verification fails.

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

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

The section entitled "Enable a TLS Connection to the Syslog Server" in [CC-Guide] provides instructions to configure a syslog server to accept audit data from the TOE. These instructions include commands to identify the syslog

server by IPv4 address. This section also states that Fabric Engine will accept certificates from the syslog server which identify the server either by IPv4 address in the SAN or CN. It will also accept certificates from a syslog server where a DNS name in the SAN or CN can be resolved to the IPv4 address that is configured.

The TOE is not distributed.

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

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

or

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

or

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

Note that for some tests additional conditions apply.

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

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

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

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

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

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

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

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

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

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

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

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

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

[conditional]: If IP addresses identifiers are supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.0.1, CN=2001:0DB8:0000:0000:0008:0800:200C:*when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A).

The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

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

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

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

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

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

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

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

Test 3: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. Using a network

sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

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

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

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

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

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

#### 2.2.12.3 NDcPP30e:FCS_TLSC_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

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

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

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

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

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

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

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

#### 2.2.12.4 NDcPP30e:FCS_TLSC_EXT.1.4

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

In section 6.2 of [ST], The TOE offers secp256r1, secp384r1, secp521r1 as supported groups for ECDHE cipher suites when acting as a TLS client. All TLS parameters are enabled by default and no configuration is required.

This section also explains that the TOE supports DHE ciphers but does not support FFDHE groups.

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

The Supported Elliptic Curves/Supported Groups Extension cannot be configured and thus the [CC-Guide] does not contain any instructions for configuration of these values.

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

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

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

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

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

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

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

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

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

Test 4a: Not applicable, the TOE does not support TLS 1.3.

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

#### 2.2.12.5 NDcPP30e:FCS_TLSC_EXT.1.5

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

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

Section 6.2 of [ST] states that the TOE offers rsa_pkcs1 with sha256, rsa_pkcs1 with sha384, and rsa_pkcs1 with sha512 as supported algorithms for the Signature Algorithms Extension when acting as a TLS client

The selection to present the signature_algorithms_cert extension is not present

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

The TOE does not need to be configured to meet the requirement.

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

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

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

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

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

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

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

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

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

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

#### 2.2.12.6 NDcPP30e:FCS_TLSC_EXT.1.6

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

In section 6.2 of [ST] all TLS parameters are enabled by default and no configuration is required.

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

The TSF does not provide the ability to configure the list of supported ciphersuites.

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

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

#### 2.2.12.7 NDcPP30e:FCS_TLSC_EXT.1.7

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

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

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

#### 2.2.12.8 NDcPP30e:FCS_TLSC_EXT.1.8

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

Not applicable as the TOE does not support TLS 1.3

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

Not applicable as the TOE does not support TLS 1.3

**Testing Assurance Activities**: None Defined

#### 2.2.12.9 NDcPP30e:FCS_TLSC_EXT.1.9

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

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

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

b. Test 2 [conditional]: If 'support TLS 1.2 secure renegotiationâ€¦â€' is selected, the evaluator shall perform a TLS 1.2 handshake and verify the TOE TLS Client's handling of ServerHello messages received during the initial handshake that include the 'renegotiation_info' extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the TOE TLS client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

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

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

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

Test 4b [conditional, if the TOE supports TLS 1.2 and rejects TLS 1.2 renegotiation attempts]: The evaluator shall initiate a TLS session between the so-configured TSF and a test TLS 1.2 server that is configured to perform a

compliant handshake, followed by a hello request. The evaluator shall confirm that the TSF completes the initial handshake successfully but does not initiate renegotiation after receiving the hello request.

(TD0899 applied)

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

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

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

Test 4a: Not Applicable, the TOE does not support TLS 1.3.

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

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

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

The TOE does not support mutual authentication, therefore all tests are configured not to require mutual authentication.

## 2.3 Identification and Authentication (FIA)

### 2.3.1 Authentication Failure Management (NDcPP30e:FIA_AFL.1)

#### 2.3.1.1 NDcPP30e:FIA_AFL.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.1.2 NDcPP30e:FIA_AFL.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

Section 6.3 of [ST] explains that the Security Administrator can configure the maximum number of failed attempts using the CLI interface. The configurable range is between 1 and 255 attempts. When a user account has exceeded maximum number of unsuccessful authentication attempts it will be locked. The host that the user was connecting from, is also locked out, but that host is automatically unlocked base on a timer. The user account remains locked out till the admin unlocks the user's account using a CLI command.

Section 6.3 of [ST] also explains that the account lockout feature is not enforced on logins occurring at the local console for the "Privilege" account. This account is allowed to login only at the console, and can unlock other accounts, thus ensuring that a system cannot get into a situation where no administrator access is available.

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

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

The section entitled "Configure Global Password Settings" in [CC-Guide] describes the TOE mechanism that locks an account, identifies the commands to configure this mechanism. The section entitled "Enable a Locked-Out User Account" describes how a user in the "Privilege Role" can unlock an account, and indicates that the user in this role can login ONLY from the local console. This prevents the account from becoming locked as a result of remote authentication failures, thus ensuring that administrators will always have access to the TOE at the 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 authentication attempts (i.e., 3 failures). The evaluator then performed more login attempts using incorrect credentials than the configured limit. The evaluator observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator then unlocked the account (using procedures from guidance), observed that the user could login successfully with the correct password and that the count of failed login attempts was reset to zero.

### 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.

In Section 6.3 of [ST], 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 ')'). The minimum password length is configurable by the Security Administrator. When the TOE is in the evaluated configuration, the minimum password length configured by a Security Administrator to a value between 8 and 32 characters (default is 15).

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

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

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

The section entitled "Configure User Passwords" in [CC-Guide] identifies the characters that can be used in a password. The section entitled "Configure Global Password Settings" explains that minimum password length can be set to a value between 8 and 32 characters. The section entitled "Configure User Passwords" provides instructions to set and change a user's password.

**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.

Test 1 & Test 2: The evaluator attempted to set/change a password for a user's account using several attempts. Throughout those attempts, every upper case letter, lower case letter, digit, and special character (as specified by the SFR in [ST]) were used in a password. The evaluator also confirmed that a minimum 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).

### 2.3.3 Protected Authentication Feedback (NDcPP30e:FIA_UAU.7)

#### 2.3.3.1 NDcPP30e:FIA_UAU.7.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

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

There are no preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

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

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

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

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

#### 2.3.4.1 NDcPP30e:FIA_UIA_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.4.2 NDcPP30e:FIA_UIA_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.4.3 NDcPP30e:FIA_UIA_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.4.4 NDcPP30e:FIA_UIA_EXT.1.4

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

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

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

Administrators according to FIA_UIA_EXT.1 the TSS shall describe any unauthenticated services/services that are supported by the component.

Section 6.3 of [ST] indicates the TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed through the following administrative interfaces:

- Directly connecting to the TOE, and
- Remotely connecting via SSHv2.

This section explains that regardless of the interface at which the Security Administrator interacts, the TOE will enforce username and authentication credentials to be presented. Authentication credentials may be a password or public-key at either the local console or via an SSHv2 protected session. The TOE also accepts an X.509v3 certificate as a valid authentication credential over an SSHv2 protected session. Only after the administrative user presents the correct authentication credentials will access to the TOE administrative functionality be granted.

Section 6.3 also states that no access is allowed to the administrative functionality of the TOE until a Security Administrator is successfully identified and authenticated.

The TOE is not distributed.

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

The TOE supports login to a command line interface (CLI) via the local serial console or a remote SSHv2 session. The section entitled "Access to the Switch" in [CC-Guide] explains that the system can be accessed for management purposes through a serial connection and an SSHv2 session.

The section entitled "Establish a Serial Connection" provides instructions for the configuration and use of the local console.

The "Secure Shell Configuration" section describes setup instructions for configuration of SSH.

The section entitled "Enhanced Secure Mode" in [CC-Guide] explains that Enhanced secure mode enables role-based access control (RBAC) and requires strong password complexity. Enhanced secure mode is required in a Common Criteria configuration. Sub-sections describe the setup of the system's admin accounts, how those accounts are defined and created, and how these accounts can be authenticated at the various admin interfaces.

The section entitled "Create User Accounts", instructs administrators to use the "password create-user" command to create administrative user accounts using local authentication.

The section entitled "Enable Public Key Authentication" explains how to configure the TOE use SSH public key authentication for a user account.

The section entitled "Enable X.509 Authentication" describes the steps necessary to configure the TOE to use x509 certificates with SSH for user authentication.

The section entitled "Disable Unsupported Services" provides instruction to disable HTTP, HTTPS, and iqagent in order to operate in an evaluated configuration. No services are offered on the TOE management network interface prior to user authentication.

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

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

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

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

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

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

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

Test 2 - Using each interface the evaluator was able to observe the TOE displayed a banner to the user before login.

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

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

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

#### 2.3.5.1 NDcPP30e:FIA_X509_EXT.1.1/Rev

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is expected that either OCSP or CRL revocation checking is performed when a certificate is presented to the TOE (e.g. during authentication). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

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

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

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

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

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

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

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

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

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

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

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

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

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

l. Test 8b: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a chain of EC certificates (terminating in a trusted CA certificate), where the intermediate certificate in the certificate chain uses an explicit format version of the Elliptic Curve parameters in

the public key information field, and is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

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

The TOE validates certificates as part of the TLS Session establishment with a syslog server and as part of SSH user authentication. The evaluator performed each of the following tests on both of these interfaces.

Test 1 -- The evaluator configured the TOE and a peer with valid certificates. The evaluator then attempted to make a connection between the peer devices using TLS protected syslog and using a remote SSH client. A successful connection was made in each case.

The evaluator then configured a syslog server that presented a certificate chain with an invalid certification path by deleting an intermediate CA so that the certificate chain was invalid because of a missing certificate. The connection between the TOE and the syslog server was refused by the TOE.

The evaluator then attempted to connect the SSH Test client to the TOE. The expectation was that the TOE would accept the first SSH connection (where the test SSH client presents a complete chain) and reject the second SSH connection (where the SSH client presents a chain missing a CA certificate).

Test 2 -- The evaluator used the TOE's TLS client (syslog) to attempt connections to a test server. The test server then presented a certificate during the TLS negotiation where the certificate was expired. The TOE rejected the connection. For this test, the evaluator configured a PKIX SSH client on a test server to send an authentication certificate that is expired and observed that the TOE SSH server rejected the connection.

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

The evaluator performed this same test presenting a revoked certificate to the TOE from a remote PKIX SSH client and observed that the TOE rejected the connection.

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

TOE also rejected the connection attempt from a PKIX client that caused the TOE to receive an OCSP response signed by a certificate without the OCSP signing purpose.

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

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

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

Test 8 -- The TOE does not support ECDSA certificates, therefore test 8 is not applicable.

#### 2.3.5.2 NDcPP30e:FIA_X509_EXT.1.2/Rev

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

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

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

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

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

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

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

Test 1: The evaluator configured a test syslog server and a PKIX SSH client to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test server and the PKIX SSH client to connect to the TOE. In each case the evaluator observed that the TOE rejected the connections.

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

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

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

The section entitled "Certificate Management" in [CC-Guide] explains that Fabric Engine can authenticate SSH users with X.509 certificates and can authenticate a network service that uses TLS. This section also states that when certificates are loaded into the system, the imported certificates are validated.

This section also identifies a set of checks that occur as part of certificate validation, which includes the following:

- Certificate expiration date check;
- Certificate path (continuity of the certificate chain) validation up to the trusted CA;
- Certificate revocation check;
- Public key, key algorithm, and parameters check;
- Check of certificate issuer;
- Process certificate extensions.

The system requires the certificate presented by the syslog server to include the ServerAuth EKU, and requires CA certificates to include the BasicConstraints flag as true. Certificates presented by a Security Administrator to the system's SSH server must include the user identity (username@domain.com) as a PrincipalName in the SubjectAltName (SAN) extension. The Fabric Engine Switch ignores all other EKU within certificates.

**Component Guidance Assurance Activities**: The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

The section entitled "Certificate Management" in [CC-Guide] explains that Fabric Engine can authenticate SSH users with X.509 certificates and can authenticate a network service that uses TLS. This section also states that when certificates are loaded into the system, the imported certificates are validated

This section also identifies a set of checks that occur as part of certificate validation, which includes the following:

- Certificate expiration date check;
- Certificate path (continuity of the certificate chain) validation up to the trusted CA;
- Certificate revocation check;
- Public key, key algorithm, and parameters check;
- Check of certificate issuer;
- Process certificate extensions.

The system requires the certificate presented by the syslog server to include the ServerAuth EKU, and requires CA certificates to include the BasicConstraints flag as true. Certificates presented by a Security Administrator to the system's SSH server must include the user identity (username@domain.com) as a PrincipalName in the SubjectAltName (SAN) extension. The Fabric Engine Switch ignores all other EKU within certificates.

**Component Testing Assurance Activities**: None Defined

### 2.3.6 X.509 Certificate Authentication (NDcPP30e:FIA_X509_EXT.2)

#### 2.3.6.1 NDcPP30e:FIA_X509_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.6.2 NDcPP30e:FIA_X509_EXT.2.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

Section 6.3 of [ST] indicates that Security Administrators configure a certificate for each service (i.e., syslog, ssh x509v3 authentication) and those certificates are used by the TOE service for authentication.

The Security Administrator is expected to configure the operating environment such that devices in the operating environment and the TOE use accurate time (to support validity check and OCSP response validity periods). The Security Administrator must also ensure that the certificates loaded into the TOE as trusted roots are those that are also accepted by network peers.

Section 6.3 also states that when the TOE determines a certificate to be valid and the necessary OCSP server cannot be contacted for a revocation check, then that certificate is not accepted as part of an SSH session negotiation, but that certificate is accepted as part of a TLS session negotiation.

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

The section entitled "Certificate Provisioning Methods" describe the two methods of certificate provisioning (offline and online management), but only the offline method is supported for an evaluated configuration.

The section entitled "Certificate Validation with OCSP" indicates that Online Certificate Status Protocol (OCSP) is used to check the revocation status of X.509 v3 certificates.

This section also explains the TOE behavior when an OCSP server cannot be contacted by the SSH Server and the TLS client. It states that the SSH Server rejects a certificate and the TLS client accepts a certificate if the OCSP server for the certificate cannot be contacted.

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

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

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

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

Because the TOE also uses x509 certificates for authentication of users for SSH sessions, the evaluator also tested the TOE behavior for SSH connections. The evaluator demonstrated that when the revocation server for the certificate presented by the SSH user was available, the TOE was successful in establishing the SSH session. The evaluator then made the revocation server inaccessible and observed that the TOE was not able to successfully

establish connections with the syslog server. Since this was the behavior claimed in the SFR selection for an SSH connection, this test passed.

### 2.3.7 X.509 Certificate Requests (NDcPP30e:FIA_X509_EXT.3)

#### 2.3.7.1 NDcPP30e:FIA_X509_EXT.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.7.2 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.

The NDcPP30e:FIA_X509_EXT.3 requirement in [ST] does not include the selection for 'device-specific information'.

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

The section entitled "Generate the Certificate Signing Request" in [CC-Guide] provides a description of the process for the TOE to issue a CSR. The section entitled "Configure Subject Parameters" includes the commands to specify subject parameters. Subject parameters are the details needed for the certificate signing request (CSR).

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

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

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

Test 1- The evaluator generated a certificate signing request by following the instructions in the guidance documentation for generating the request. The request was then exported to an external CA where the evaluator verified the CSR could be read as a well-formed CSR by a non-TOE test server. While the CSR was within the CA, the evaluator examined the CSR and found that it included the fields identified in the Security Target.

Test 2 - The evaluator signed the CSR from test 1 using a CA certificate that did not chain to a trusted root installed on the TOE. The attempt to import this certificate into the TOE failed. Since the TOE was already configured with a valid root certificate, the evaluator signed the CSR from test 1 using the CA certificate that did chain to the trusted root that was already installed on the TOE. This import attempt was successful.

## 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 6.4 in [ST] states that only Security Administrators can determine/modify the behavior of the transmission of audit data to an external audit server. Only Security Administrators can determine/modify the handling of audit data by viewing and changing the logging levels.

**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 section entitled "Enable a TLS Connection to the Syslog Server" describes the TOE's functionality to establish a trusted channel between itself and the audit server as well as commands to configure a connection.

The section entitled "Log Files" describes that Files are stored locally, with maximum capacity based on current available hard-drive space. You should free disk space on the flash if the system generates an alarm based on system thresholds, which are between 75% and 90%:

• 5320 and 5420 devices have a threshold of 90%.

• Other devices have a threshold of 75%.

After disk space usage returns below the device threshold, the system clears the alarm, and then starts logging to a file again.

The section entitled "Log Message Severity" describes the TOE can be configured to generate audit data to local log file or remote host based on the severity of the TOE's behavior.

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

The evaluator shall perform the following tests:

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

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

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

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

The evaluator shall perform the following tests:

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

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

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

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

The evaluator shall perform the following tests:

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

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

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

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

The evaluator shall perform the following tests:

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

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

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

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

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

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

#### 2.4.2.1 NDCPP30E:FMT_MOF.1.1/MANUALUPDATE

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

The TOE is non-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).

The section entitled "Software Upgrade" of [CC-Guide] provides instructions on how an administrator can initiate a product update.

The TOE is not distributed, and thus the guidance only discusses updates to the one component that is the TOE.

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

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

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

The evaluator tested to determine that 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.

The TOE is not distributed.

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

#### 2.4.3.1 NDcPP30e:FMT_MTD.1.1/CoreData

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

In section 6.4 in [ST], 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 Security Administrator is successfully identified and authenticated. Security management is restricted to Security Administrators. The trust store is accessed when Security Administrators import/remove certificates as described in the Admin Guide. The trust store is protected by default and is restricted such that only Security Administrators have access.

**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.

Specific sections of the [CC-guide] and commands are identified or referenced throughout this AAR with the requirement to which they apply. The section entitled "Overview" explains that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on the Common Criteria standards. Network management communication paths are protected against modification and disclosure by SSHv2. This section also explains that the TOE supports only a trusted channel to an external audit server and that this trusted channel must be configured to be protected by TLS.

The set of subsections within the section entitled "Certificate Management" describe the various administrative actions that administrators can perform to generate key-pairs, generate Certificate-signing requests, and manage certificates.

The section entitled "Specify and Enable the NTP Server" and "Manage NTP Authentication" explain how to configure the NTP client within the TOE to ensure the TOE time is accurate. It also indicates that an authentication key must be provided for each configure NTP server.

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

No separate testing for FMT_MTD.1/CoreData is required.

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

#### 2.4.4.1 NDcPP30e:FMT_MTD.1.1/CryptoKeys

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

In section 6.4 of [ST], The TOE only allows Security Administrators to perform management operations including the command to generate and delete cryptographic keys. Security Administrators can also import and delete CA certificates and their keys into the trust store.

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

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

The section entitled "Secure Shell Configuration" lists the keys which an administrator is able to manage as SSH host keys and SSH x509 Server Certificate keys. The section entitled "Enable RSA Authentication and Generate the Host Key" describes how to configure and generate an SSH Host Key to be used by the TOE to authenticate itself to the SSH client. The section entitled "Enable Public Key Authentication" explains how to configure user accounts w/ a public key, so that they can login w/o a password and w/o an x509 certificate. Finally, the section entitled "Enable RSA Authentication and Generate the Host Key" explains how to configure the TOE to use an X509 certificate as its host key, while the section entitled "Enable X.509 Authentication" explains how to configure per-user x509 certificate.

These include the public and private SSH host key generated using instructions in section entitled "Enable RSA Authentication and Generate the Host Key".

The section entitled "Enable RSA Authentication and Generate the Host Key" explains how to generate and delete an RSA host key. It also explains that the generation of a new host key will overwrite the previous key.

The section entitled "Remove a Key" provides instructions to delete a key associated with a CSR and its certificate from the certificate store.

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

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

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

The evaluator attempted to modify, delete, generate or import a cryptographic key before being authenticated as an administrator. The attempt was observed to fail. The evaluator then completed a login and attempted the same command. The attempt after a successful login was observed to be successful.

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

#### 2.4.5.1 NDcPP30e:FMT_SMF.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

(containing also requirements on Guidance Documentation and Tests)

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

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

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

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

Section 6.4 of [ST] indicates that the TOE is securely managed via the CLI which is available through a local console or over an SSHv2 protected session. The CLI offers command line functions which allow Security Administrators to configure the TOE. These command line functions can be used to effectively manage every security feature (supporting all requirements), as well as the non-security relevant aspects of the TOE.

Section 6.4 of [ST] also lists the management functions offered by the TOE. These functions correspond to those required by FMT_SMF.1 and were observed by the evaluator during testing. The specific management capabilities defined in the ST include:

Ability to administer the TOE remotely;

Ability to configure the access banner;

Ability to configure the remote session inactivity time before session termination;

Ability to update the TOE, and to verify the updates using digital signature capability prior to installing those updates;

Ability to modify the behavior of the transmission of audit data to an external IT entity,

Ability to manage the cryptographic keys,

Ability to configure the cryptographic functionality,

Ability to re-enable an Administrator account,

Ability to set the time which is used for time-stamps,

Ability to configure NTP,

Ability to configure the reference identifier for the peer,

Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors,

Ability to generate Certificate Signing Request (CSR) and process CA certificate response,

Ability to administer the TOE locally,

Ability to configure the local session inactivity time before session termination or locking,

Ability to configure the authentication failure parameters for FIA_AFL.1,

Ability to manage the trusted public keys database

Ability to configure local audit behavior (e.g. changes to storage locations for audit; changes to behaviour when local audit storage space is full; changes to local audit storage size),

The TOE is not distributed

**Component Guidance Assurance Activities**: See TSS Assurance Activities

The TOE is compliant with all requirements in the ST as identified in this report.

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

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

### 2.4.6 Restrictions on Security Roles (NDcPP30e:FMT_SMR.2)

#### 2.4.6.1 NDcPP30e:FMT_SMR.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.6.2 NDcPP30E:FMT_SMR.2.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.6.3 NDcPP30E:FMT_SMR.2.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

Section 6.4 of [ST] indicates that management functions are exclusively restricted to Security Administrators with corresponding privileges. The term "Security Administrator" used in the ST refers to any user that has a role that has been assigned any of the privileges allowing the user to perform any of the management functions. Not every administrator would necessarily have sufficient privileges to access each administrative function.

The TOE supports multiple administrative roles when accessing the administrative interface through the local or remote CLI. These roles define the access that is allowed per role. The following list identifies the configuration capabilities assigned to each role.

- User EXEC Mode: Initial mode of access.
- Privileged EXEC Mode: User mode and password combination determines access level.
- Global Configuration Mode: Use this mode to make changes to the running configuration.
- Interface Configuration Mode: Use this mode to modify or configure logical interface, VLAN or a physical interface.
- Router Configuration Mode: Use this mode to modify a protocol.

- Application Configuration Mode: Use this mode to access the applications.

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

The section entitled " Access to the Switch " in [CC-Guide] indicates administrators can access a Fabric Engine device by Serial Connection or by SSH. The "SSHv2" heading explains that an administrator can access the device from a remote client by using the ssh command. The admin must provide the appropriate user credentials to gain access to the device. You can close the session by running the exit command. The material under the SSH heading references the section entitled "Secure Shell Configuration" with instructions on how to enable SSH, configure algorithms, rekey limits, etc. on a Fabric Engine switch.

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

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

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

The TOE is not distributed.

## 2.5 Protection of the TSF (FPT)

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

#### 2.5.1.1 NDcPP30e:FPT_APW_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.5.1.2 NDcPP30e:FPT_APW_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

In Section 6.5 of [ST], Passwords are the only authentication data that is subject to this SFR. No passwords are ever stored as clear text. The TOE does not offer any functions that will disclose to any user a plain text password. Passwords are stored on the TOE in a secured partition in non-plaintext. Prior to writing on disks each password is hashed (SHA-256) with a salt. During subsequent authentication attempts passwords are similarly processed and compared in cyphertext (i.e., hash comparison).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

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

#### 2.5.2.1 NDcPP30e:FPT_SKP_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through any

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

Section 6.5 of the ST states that the TOE is designed with a set of self-protection mechanisms. All passwords, and keys are stored on the TOE are protected from unauthorized modification and disclosure. The TOE stores symmetric keys only in volatile memory never on persistent media. The TOE admin interface does not provide any mechanism to view or directly modify passwords, symmetric keys, or private keys. The TOE encrypts and stores all private keys in a secure directory that is not directly accessible to administrators; therefore, there is no administrative interface access provided to directly manipulate the keys. Table 6-3 in Section 6.2 of [ST] indicates how keys are stored.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.5.3 Reliable Time Stamps (NDcPP30e:FPT_STM_EXT.1)

#### 2.5.3.1 NDcPP30e:FPT_STM_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.5.3.2 NDcPP30e:FPT_STM_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

Section 6.5 of [ST] states that the TOE includes its own hardware clock and can synchronize with a NTP server. The clock function is reliant on the system clock provided by the underlying hardware. The TOE can be configured to synchronize its internal clock with an NTP server. The date and time are used as the time stamp that is applied to TOE generated audit records, used to track inactivity of administrative sessions, and perform certificate expiration checks.

The TOE does not obtain time from the underlying virtualization system.

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

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

The section entitled "Set the System Date, Time, and Time Zone" in [CC-Guide] explains how an administrator can set the date, the time and the time zone on the TOE. The section entitled "Specify and Enable the NTP Server" explains the TOE supports NTPv4 and allows up to 10 IPv4 NTP servers and 10 IPv6 NTP servers to be configured.

The TOE does not rely upon 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 followed the guidance instructions to configure the time on the TOE. The evaluator read the time from the TOE using a date command and also found audit records confirming that the time was successfully changed.

Test 2: The TOE does support the use of NTP to set time. The NTP capabilities were tested as part of FCS_NTP_EXT.1 testing.

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

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

#### 2.5.4.1 NDCPP30E:FPT_TST_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.5.4.2 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)

In section 6.5 of [ST], The TOE runs a suite of self-tests during initial start-up to verify its correct operation. If any of the tests fail one of the following will happen: the TOE will enter into an error state until an Security Administrator intervenes or the TOE will automatically reboot and re-run all the tests. During initialization and self-test execution, the module inhibits all access to the cryptographic algorithms. Additionally, the power-on self-tests are performed after the cryptographic systems are initialized but prior to the underlying OS initialization of external interfaces; this prevents the security appliances from passing any data before completing self-tests. In the event of a power-on self-test failure, the cryptographic module will force the platform to reload and reinitialize the operating system and cryptographic components. This operation ensures no cryptographic algorithms can be accessed unless all power on self-tests are successful. By ensuring that cryptographic operations are accurate and that the TOE software image is unmodified, these self-tests are sufficient to demonstrate the TSF operates as correctly.

These tests include:

· 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.

· 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.

· PRNG/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.

· SHA 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.

· 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.

· Software Integrity Test - This test is run automatically whenever the system images is loaded and confirms through use of digital signature verification that the image file that's about to be loaded was properly signed and maintained its integrity since being signed. The system image is digitally signed prior to being made available for download from Extreme.

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

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

The section entitled "Self-Test Audit Log Records" in [CC-Guide] explains that failure of any self-test during the start-up process stops the process and prompts you to reload.

The TOE is not distributed.

**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.

### 2.5.5 TRUSTED UPDATE (NDCPP30E:FPT_TUD_EXT.1)

#### 2.5.5.1 NDCPP30E:FPT_TUD_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.5.5.2 NDCPP30E:FPT_TUD_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.5.5.3 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.

In section 6.5 of [ST], The Authorized Administrator can query the software version running on the TOE, and can initiate updates to software images. When software updates are made available, a Security Administrator can obtain, verify the integrity of via digital signature, and install those updates. The updates can be downloaded from <https://support.extremenetworks.com>. 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. The public keys used by the update verification mechanism are contained on the TOE. The TOE compares the update files' signature using a certificate that comes pre-loaded on the device and is stored in the trust store. As part of the build process, the update image is signed with the Extreme private key. This is done using an RSA 2048/SHA-256 digital signature. Only if the signature/hash is correct, will the image be installed. If an update is unsuccessful, a warning is displayed to the Security Administrator. Since the update process attempts to update a different partition than what is currently being run, the current active image remains the same until the reboot. The reboot prompt is offered as part of the update process. The evaluator verified the behavior described matches the behavior performed by the TOE when updating.

Neither options 'support automatic checking for updates' or 'support automatic updates' are chosen from the selection in FPT_TUD_EXT.1.2

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.

The section entitled "Software Upgrade" in [CC-Guide] begins with an introduction to the process for upgrading the TOE software. This section describes that the Fabric Engine software used delayed activation method for installation. It explains that software is installed in a software inventory before being activated. Once activated, the new software becomes the primary image, the current primary becomes the backup image, and the switch must be reset for the change to complete the upgrade.

The "Display Software Inventory" sub-heading provides the command necessary to have the TOE display the available releases that have been installed in the TOE. This sub-heading states that the phrase "Primary Release" identifies the active running software.

The section entitled "Software Upgrade" under the sub-heading of "Upgrade the software", states that during upgrade, the system verifies the digital signatures that are embedded in the upgrade files and rejects installation of an image that has an invalid signature.

The TOE does not use published hashes and is not distributed.

**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: The evaluator verifies the current version of the product. Then the evaluator uses a legitimate update and guidance documentation to install the update to the TOE. The evaluator then uses the documentation to load and activate the update on the TOE. Upon activation, the evaluator verifies the update was committed to the TOE.

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.

## 2.6 TOE ACCESS (FTA)

### 2.6.1 TSF-INITIATED TERMINATION (NDCPP30E:FTA_SSL.3)

#### 2.6.1.1 NDcPP30e:FTA_SSL.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

In Section 6.6 of [ST], The TOE provides the administrative user with an ability to configure inactivity time out periods for administrative sessions. The inactivity period for CLI (local and remote) administrative access is configured separately through the TOE administrative interfaces. When the administrative interface has been idle for more than the configured period of time the session is terminated. After termination, administrative authentication is required to access any of the administrative functionality of the TOE. This is applicable from both local and remote administrative sessions.

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

The section entitled, "Configure a Session Inactivity Timeout Threshold" in [CC-Guide] explains that upon timeout of a remote SSH session, the session is terminated and the user must login again.

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

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

The evaluator followed the guidance to configure the session timeout periods for SSH CLI remote sessions. The evaluator confirmed that the session was terminated after the configured time period. The inactivity time period was configured for periods of 1 minute, 3 minutes and 5 minutes.

### 2.6.2 User-initiated Termination (NDcPP30e:FTA_SSL.4)

#### 2.6.2.1 NDcPP30e:FTA_SSL.4.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

In section 6.6 of [ST], The TOE provides the function to logout (or terminate) both local and remote user sessions as directed by the user.

**Component Guidance Assurance Activities**: The evaluator shall confirm that the guidance documentation states how to terminate a remote interactive session (and if applicable the local administrative session).

The section entitled "Access to the Switch" in [CC-Guide] explains that the 'exit' or 'logout' command can be used at the CLI to terminate the user's interactive session on either the local console or a remote SSH connection.

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

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

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

Test 1: The evaluator logged in to the local console and then typed in the command "logout". The evaluator observed that the session ended and a login prompt was presented.

Test 2: The evaluator repeated this test using an SSH connection and observed that the session ended and the SSH connections was terminated.

### 2.6.3 TSF-initiated Session Locking (NDcPP30e:FTA_SSL_EXT.1)

#### 2.6.3.1 NDcPP30e:FTA_SSL_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

In section 6.6 in [ST], The TOE terminates local sessions that have been inactive for an administrator-configured period of time.

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

The section entitled, "Configure a Session Inactivity Timeout Threshold" in [CC-Guide] contains instructions to configure the inactivity timeout period for console sessions. This section explains that upon timeout of a local console session, the session is terminated and the user must login again.

**Component Testing Assurance Activities**: The evaluator shall perform the following test:

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

The evaluator followed the guidance to configure the idle timeout periods for the Local Console session and confirmed that the session was terminated after the configured time period. The inactivity time period was configured using the "console timeout" command for periods of 1 minute, 3 minutes and 5 minutes.

### 2.6.4 Default TOE Access Banners (NDcPP30e:FTA_TAB.1)

#### 2.6.4.1 NDcPP30e:FTA_TAB.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

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

Section 6.6 of [ST] states that the local console CLI and remote SSH CLI can be configured to display a custom login banner. This banner will be displayed prior to allowing Security Administrator access through either interface.

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

The section entitled "Configure the Banner Message" in [CC-Guide] provides the commands that an administrator can use to configure the message that users see before they log in and the message of the day that they see after they log in. This section also explains that the banner is displayed on the serial connection and SSHv2 CLI.

**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.

## 2.7 Trusted path/channels (FTP)

### 2.7.1 Inter-TSF trusted channel (NDcPP30e:FTP_ITC.1)

#### 2.7.1.1 NDcPP30e:FTP_ITC.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.7.1.2 NDcPP30e:FTP_ITC.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.7.1.3 NDcPP30e:FTP_ITC.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Section 6.7 of [ST] indicates that the TOE protects trusted channels with audit servers (syslog servers) using the TLS v1.2 protocol. The TOE is a TLS client in the communications with the audit servers. The TOE provides assured identification of the non-TSF endpoint by validating X.509 certificates. The TOE implements a trust store containing trust anchors which it uses to verify identities of those non-TSF certificates. The TOE utilizes TLS as described in Section 6.2 of [ST].

**Component Guidance Assurance Activities**: The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

The section entitled "Enable a TLS Connection to the Syslog Server" in [CC-Guide] provides instructions to configure the TOE to connect to an external syslog server using TLS to protect the communication pathway.

**Component Testing Assurance Activities**: The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

a. Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b. Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

c. Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

d. Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public- facing document or report.

The TOE utilizes TLS to protect communications with an external audit server (syslog server).

A successful TOE TLS connection supporting communication to an external audit server was established. Examining the packet capture from that test evaluators saw that the connection between the TOE component and the external syslog server was established; the TOE initiated the connection; and Application data that was transferred is encrypted (i.e., not plaintext).

The evaluator began a packet capture 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 TLS connections, the evaluator waited for the syslog server to begin receiving audit data again and stopped the packet capture shortly after traffic began flowing after the disruption. The evaluator observed that no data was transmitted unprotected.

The evaluator also used the TOE to initiate a TLS protected communication pathway to an external authentication server. Examination of the packet capture obtained during this activity showed that the connection was protected by TLS, 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 TLS protected syslog was demonstrated.

Test 2: The TOE initiated a TLS connection for TLS protected syslog.

Test 3: Syslog communication was not plaintext.

Test 4: A physical disruption in the network resulted in a TLS session interruption and no data was transmitted unprotected.

### 2.7.2 Trusted Path (NDcPP30e:FTP_TRP.1/Admin)

#### 2.7.2.1 NDcPP30e:FTP_TRP.1.1/Admin

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.7.2.2 NDcPP30e:FTP_TRP.1.2/Admin

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.7.2.3 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.

In section 6.7 of [ST], The TOE provides SSH to ensure secure remote administration. The Security Administrator can initiate the remote SSH session, the remote SSH session is secured from disclosure and modification using CAVP tested cryptographic operations.

**Component Guidance Assurance Activities**: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

The section entitled "Set the Management IP Address" in [CC-Guide] explains how to define the management IP address that allows administrators to remotely access the TOE using the out-of-band management port.

The section entitled "Access to the Switch" indicates administrators normally have 2 options to access a Fabric Engine device by Serial Connection, or by SSH. The "SSHv2" heading explains that an administrator can access the device from a remote client by using the ssh command. The admin must provide the appropriate user credentials to gain access to the device. You can close the session by running the exit command. The material under the SSH heading references the section entitled "Secure Shell Configuration" with instructions on how to enable SSH, configure algorithms, rekey limits, etc. on a Fabric Engine switch.

The material under the Serial Connection and SSHv2 are allowed remote administration methods in the evaluated CC configuration. The section entitled "Disable Unused and Unsupported Services" provides instructions on how change the TOE configuration to remove unevaluated features (e.g., HTTP and HTTPS management interface, telnet daemon, ftp daemon, and iqagent.

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

a. Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

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

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

The TOE offers remote administration via SSHv2 to provide the trusted path (with protection from disclosure and modification) for all remote administration sessions.

The evaluator performed the following on the SSH protected CLI.

a) The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE.

b) The evaluator connected to the TOE and performed a login using an administrator account

c) The evaluator then terminated the connection by 'Logout' and terminated the packet capture.

The evaluator established a packet capture of an SSH over IPsec connection to the TOE respectively, then caused a physical disruption of the network connection between the administrative workstation and the TOE. The disruption lasted several minutes. The session terminated and needed to be renegotiated following the reconnection of the network. No date 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 SSH protected remote administration was demonstrated.

Test 2: Remote administration sessions protected by SSH did not contain plaintext data.

# 3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

## 3.1 Development (ADV)

### 3.1.1 Basic functional specification (ADV_FSP.1)

**Assurance Activities**: The EAs for this assurance component focus on understanding the interfaces (e.g., application programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Explicitly labeling TSFI as security relevant or non-security relevant is not necessary. A TSFI is implicitly security relevant if it is used to satisfy an

evaluation activity, or if it is identified in the ST or guidance documentation as adhering to the security policies (as presented in the SFRs). The intent is that these interfaces will be adequately tested and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied. According to the description above 'security relevant' corresponds to the combination of 'SFR-enforcing' and 'SFR-supporting' as defined in CC Part 3, paragraph 224 and 225.

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

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

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

The evaluator shall use the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

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

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

The Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

## 3.2 Guidance Documents (AGD)

### 3.2.1 Operational user guidance (AGD_OPE.1)

**Assurance Activities**: It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to Security Administrators and users (as appropriate) as part of the TOE.

Note that additional Evaluation Activities for the guidance documentation in the case of a distributed TOE are defined in Appendix B.4.2.1.

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR. For the related evaluation activities, the evaluation evidence documents Security Target, AGD documentation (user guidance) and functional specification documentation (if provided) shall be used as input documents. Each input document is subject to ALC_CMS.1-2 requirements.

In addition, the evaluator performs the EAs specified below.

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR. For the related evaluation activities, the evaluation evidence documents Security Target, AGD documentation (user guidance) and functional specification documentation (if provided) shall be used as input documents. Each input document is subject to ALC_CMS.1-2 requirements.

In addition, the evaluator performs the EAs specified below.

The evaluator shall ensure the Operational guidance documentation is distributed to Security Administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that Security Administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

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

The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

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

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

b. The evaluator shall verify that this process includes instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

c. The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

As identified throughout this AAR, the [CC-Guide] provides instructions for configuring the TOE's cryptographic security functions. The [CC-Guide] provides instructions for configuring the cryptographic algorithms and parameters used for the evaluated configuration. The [CC-Guide] is clear that no other cryptographic configuration has been evaluated or tested. There are warnings and notes throughout the [CC-Guide] regarding use of functions that are and are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP30e:FPT_TUD_EXT.1.

### 3.2.2 PREPARATIVE PROCEDURES (AGD_PRE.1)

**Assurance Activities**: The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

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

In addition, the evaluator performs the EAs specified below.

The evaluator shall examine the Preparative procedures to ensure they include a description of how the Security Administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

The evaluator shall examine the preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

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

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

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

The preparative procedures must

a. include instructions to provide a protected administrative capability; and

b. identify TOE passwords that have default values associated with them and mandate that they shall be changed.

The evaluation team had the following documents to use when configuring the TOE:

- Extreme Fabric Engine Common Criteria Configuration Guide 9.1.100, May 2026 [**CC- Guide]**

The completeness of this documentation is addressed by its use in the AA's carried out in the evaluation.

## 3.3 Life-cycle support (ALC)

### 3.3.1 Labelling of the TOE (ALC_CMC.1)

**Assurance Activities**: When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

ALC_CMC.1-1 The evaluator shall check that the TOE provided for evaluation is labelled with its reference.

989 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

Alternatively, the unique reference provided for the TOE may be the combination of the unique reference of each component from which the TOE is comprised (e.g. in the case of a composed TOE).

ALC_CMC.1-2 The evaluator shall check that the TOE references used are consistent.

If the TOE is labelled more than once then the labels have to be consistent. For example, it should be possible to relate any labelled guidance documentation supplied as part of the TOE to the evaluated operational TOE. This ensures that consumers can be confident that they have purchased the evaluated version of the TOE, that they have installed this version, and that they have the correct version of the guidance to operate the TOE in accordance with its ST.

The evaluator also verifies that the TOE reference is consistent with the ST.

If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumers will be able to determine whether they have the appropriate items.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software

versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

### 3.3.2 TOE CM coverage (ALC_CMS.1)

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

ALC_CMS.1-1 The evaluator shall check that the configuration list includes the following set of items:

a) the TOE itself;

b) the evaluation evidence required by the SARs in the ST.

ALC_CMS.1-2 The evaluator shall examine the configuration list to determine that it uniquely identifies each configuration item.

1103 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

See section 3.3.1 above for an explanation of how all CM items are addressed.

## 3.4 Tests (ATE)

### 3.4.1 Independent testing - conformance (ATE_IND.1)

**Assurance Activities**: The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2, 3 and 4.

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

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in Appendix B.4.3.1.

The evaluation team exercised the independent tests specified in the *'collaborative Protection Profile for Network Devices', Version 3.0e, 06 December 2023/'Functional Package for Secure Shell (SSH)', Version 1.0, 13 May 2021 (NDcPP30e/SSH10)* ) against the evaluated configuration of the TOE. The following diagram indicates the test environment.

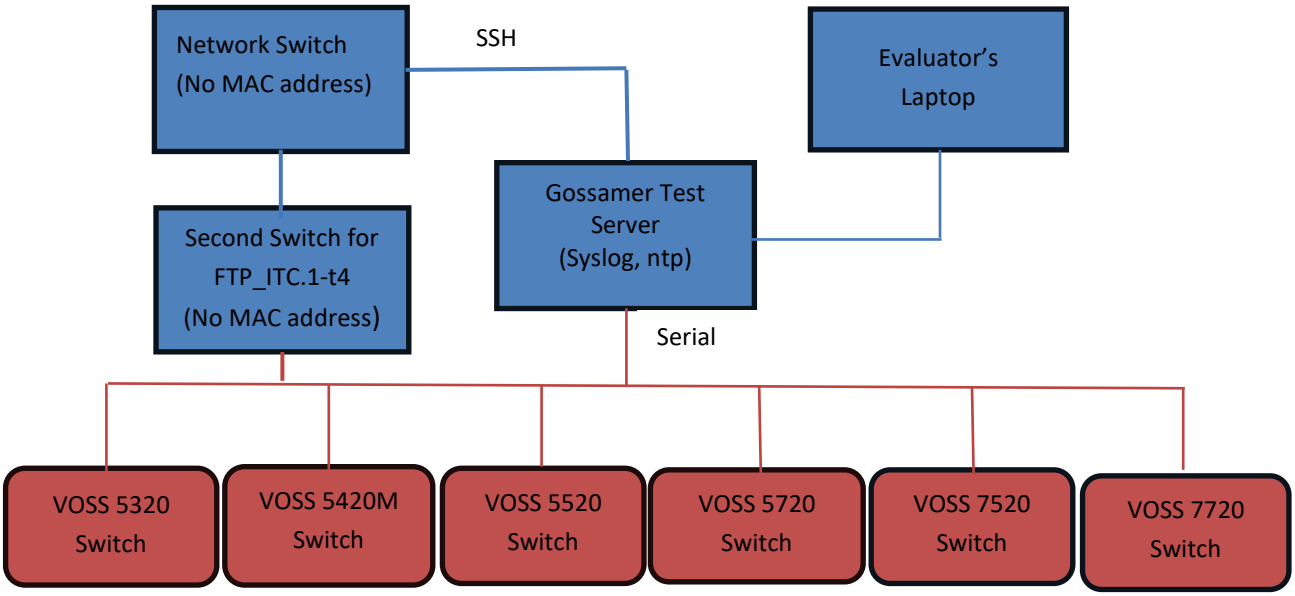


**Figure 1 Test Setup**

**TOE Platforms:**

- VOSS5320
- VOSS5420M
- VOSS5520
- VOSS5720
- VOSS7520
- VOSS7720

**Supporting Products and Software:**

The Gossamer Test server utilized both Windows and Ubuntu Environments.

The Windows supporting environment included the following:

- Putty version 0.82 (used to connect to device console and Ubuntu environment)
- Wireshark version 4.6.4

The Ubuntu 16 supporting environment included the following:

- tcpdump version 4.9.3
- libpcap version 1.7.4
- OpenSSL 1.0.2g-fips 1 Mar 2016
- Nmap version 7.01
- rsysogd 8.16.0
- OpenSSH-client version 9.3p1

The Evaluators also utilized a second Ubuntu 24 setup for the purposes of TLSC and FAU_STG_EXT.1-t1 testing as well as all VOSS5420M testing:

- tcpdump version 4.99.4
- libpcap version 1.10.4
- OpenSSH-client version 9.6p1
- OpenSSL 3.0.13+GSS 4
- Nmap version 7.94SVN
- rsyslogd  8.2312.0

### 3.4.2 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, ref NVD), MITRE CVE Database, National Vulnerability Database, and CVE details (https://www.cve.org/, https://web.nvd.nist.gov/vuln/search, and https://www.cvedetails.com/vulnerability-search.php, ref CVE), Known Vulnerability Exploit Catalog (https://www.cisa.gov/known-exploited-vulnerabilities-catalog, ref KEV), Vulnerability Notes Database (http://www.kb.cert.org/vuls/, ref VND), Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities, ref Rapid7), Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories, ref ZDI), Tenable Network Security (http://nessus.org/plugins/index.php?view=search, ref TEN), Offensive Security Exploit Database (https://www.exploit-db.com/, ref EDB) on 5/28/2026 (from 12/1/2022) with the following search terms: "Extreme", "VOSS", "VSP", "Intel Atom", "Fabric Engine", "Extreme Networks", "Mocana", "Broadcom", "DigiCert".

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.