Samsung Electronics Co. Ltd.

Samsung 5G gNB AU, DU v19.A

Assurance Activity Report

Version 1.1

November 10, 2020

Document prepared by

www.lightshipsec.com

Table of Contents

1 Introduction

1 This Assurance Activity Report (AAR) documents the evaluation activities performed by Lightship Security for the evaluation identified in Table 1. The AAR is produced in accordance with National Information Assurance Program (NIAP) reporting guidelines.

1.1 Evaluation Identifiers

Table 1: Evaluation Identifiers

Scheme	Canadian Common Criteria Scheme
Evaluation Facility	Lightship Security
Developer/Sponsor	Samsung Electronics Co. Ltd.
TOE	Samsung 5G gNB AU, DU v19.A
Security Target	Samsung 5G gNB AU, DU v19.A Security Target, v1.4, November 2020
Protection Profile	collaborative Protection Profile for Network Devices, v2.2E, 23-March-2020

1.2 Evaluation Methods

2 The evaluation was performed using the methods, tools and standards identified in Table 2.

Table 2: Evaluation Methods

Evaluation Criteria	CC v3.1R5
Evaluation Methodology	CEM v3.1R5
Supporting Documents	Evaluation Activities for Network Device cPP, v2.2 (NDcPP-SD), December 2019.
Interpretations	**NDcPP v2.2e** 0538 – NIT Technical Decision for Outdated link to allowed-with list 0537 – NIT Technical Decision for Incorrect reference to FCS_TLSC_EXT.2.3 0536 – NIT Technical Decision for Update Verification Inconsistency

	0528 – NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4 0527 – Updates to Certificate Revocation Testing (FIA_X509_EXT.1) *X509 not claimed, and therefore TD0527 not applicable*
Tools	Please refer to the Test Plan [IND].

2 TOE Details

2.1 Overview

1 The [ST] defines the Samsung 5G gNB AU, DU v19.A Target of Evaluation (TOE) for the purposes of Common Criteria (CC) evaluation.

2 The 5G gNB AU and DU devices are components of Samsung's 5G Next Generation NodeB (gNB) base station that provides various functions such as signal processing and resource management.

2.2 TOE models

Type	Model	CPU	Software	Differences
DU	Cabinet DU	Cavium **CN9670**	19.A.0	AU incorporates RU.
AU	AT1K01-A00 (AC)	Cavium **CN8370**	19.A.0	AU comes with different power supply options.
	AT1K01-A10 (DC)	Cavium **CN8370**	19.A.0	

2.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
[ST]	Samsung 5G gNB AU, DU v19.A Security Target, v1.4
[CC_Guide]	Samsung 5G AU/DU V19.A.0 Common Criteria Guide, Version 1.3
[AGD]	Samsung 5G NR AU Command Reference for SVR 19A, v1.0 Samsung 5G NR DU Command Reference for SVR 19A, v1.0 Samsung CONFD CLI User Guide, v1.1
[IND]	Samsung 5G gNB AU, DU v19.A NDcPP 2.2E Test Plan, v1.0
[AVA]	Samsung 5G gNB AU, DU NDcPP 2.2E Vulnerability Assessment, v1.0
[CBG4]	Canadian Common Criteria Scheme Guidance for Evaluators Guide #4, version 3.5, September 2018

2.4 Summary of SFRs

Requirement	Title
FAU_GEN.1	Audit Data Generation

Requirement	Title
FAU_GEN.2	User Identity Association
FAU_STG_EXT.1	Protected Audit Event Storage
FCS_CKM.1	Cryptographic Key Generation
FCS_CKM.2	Cryptographic Key Establishment
FCS_CKM.4	Cryptographic Key Destruction
FCS_COP.1/DataEncryption	Cryptographic Operation (AES Data Encryption/Decryption)
FCS_COP.1/SigGen	Cryptographic Operation (Signature Generation and Verification)
FCS_COP.1/Hash	Cryptographic Operation (Hash Algorithm)
FCS_COP.1/KeyedHash	Cryptographic Operation (Keyed Hash Algorithm)
FCS_RBG_EXT.1	Random Bit Generation
FCS_SSHC_EXT.1	SSH Client Protocol
FCS_SSHS_EXT.1	SSH Server Protocol
FCS_NTP_EXT.1	NTP Protocol
FIA_AFL.1	Authentication Failure Management
FIA_PMG_EXT.1	Password Management
FIA_UIA_EXT.1	User Identification and Authentication
FIA_UAU_EXT.2	Password-based Authentication Mechanism
FIA_UAU.7	Protected Authentication Feedback
FMT_MOF.1/ManualUpdate	Management of security functions behaviour
FMT_MTD.1/CoreData	Management of TSF Data
FMT_MTD.1/CryptoKeys	Management of TSF Data
FMT_SMF.1	Specification of Management Functions
FMT_SMR.2	Restrictions on Security Roles
FPT_SKP_EXT.1	Protection of TSF Data (for reading of all pre-shared, symmetric and private keys)
FPT_APW_EXT.1	Protection of Administrator Passwords

Requirement	Title
FPT_TST_EXT.1	TSF testing
FPT_TUD_EXT.1	Trusted update
FPT_STM_EXT.1	Reliable Time Stamps
FTA_SSL_EXT.1	TSF-initiated Session Locking
FTA_SSL.3	TSF-initiated Termination
FTA_SSL.4	User-initiated Termination
FTA_TAB.1	Default TOE Access Banners
FTP_ITC.1	Inter-TSF trusted channel
FTP_TRP.1/Admin	Trusted Path

Table 4: List of SFRs

3 Evaluation Activities for SFRs

3.1 Security Audit (FAU)

3.1.1 FAU_GEN.1 Audit data generation

3.1.1.1 TSS

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

Findings:	This requirement is addressed in Section 6.1.1 of [ST] – generation of SSH key pair is logged and key reference is logged to identify the relevant key.

4 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.

Findings:	The TOE is not a distributed TOE.

3.1.1.2 Guidance Documentation

5 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).

Findings:	Section 4.2 of [CC_Guide] shows an example of each auditable event required by FAU_GEN.1 (including each mandatory or selection-based SFR in the ST, if there is an auditable event associated with the SFR).

6 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.

Findings:	The evaluator made his determination of the administrative actions, by using FMT SFRs, FIA_PMG_EXT.1, FPT SFRs, and FTA SFRs, as a guide, and navigating through the user guide document for administrative actions related to those functional requirements, specifically, configuration changes related to TSF data and TSF. This includes the following sections in [CC_Guide] and [AGD]:

Updating the TOE ([CC_Guide])
Cryptography ([CC_Guide])
Setting Times ([CC_Guide])
Audit Logging ([CC_Guide])
Administrator Authentication ([CC_Guide])
Trusted Channel ([CC_Guide])
Local cli user password change (Confd CLI User Guide of [AGD])

The evaluator also performed the administrative operations, involving changes to TSF data and TSF behaviour, as part of NDcPP v2.2E Test assurance activities, and confirmed that the documents provide sufficient information and instructions to enable evaluator to conduct NDcPP test requirements successfully.

3.1.1.3 Tests

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

8 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.

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

Findings: These tests are conducted throughout the test plan.

3.1.2 FAU_GEN.2 User identity association

3.1.2.1 TSS

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

3.1.2.2 Tests

11 This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

12 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.

Findings:	These activities are performed in conjunction with the testing of FAU_GEN.1.1.

3.1.3 FAU_STG_EXT.1 Protected audit event storage

3.1.3.1 TSS

13 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.

Findings:	In Section 6.1.3 of [ST], it states that audit data are transferred via SFTP to the external audit server (i.e. USM) periodically.

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

Findings:	In Section 6.1.3 of [ST], it describes the amount of audit data which are stored locally; it also describes what happens when local audit data store is full, and protection of local audit data.

15 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.

Findings:	The TOE is a single, standalone component and not a distributed TOE.

16 The evaluator shall examine the TSS to ensure that 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.

Findings:	In Section 6.1.3 of [ST], it states that audit log files are rotated – the oldest file will be overwritten with latest audit data.

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

Findings:	In Section 6.1.3 of [ST], it states that transmission of audit information to an external IT entity is done periodically: it is transferred to external audit server, i.e. USM, at 30 minutes after the hour, every hour.

18 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).

Findings:	N/A. TOE is not a distributed TOE.

19 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.

Findings:	N/A. TOE is not a distributed TOE.

3.1.3.2 Guidance Documentation

20 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.

Findings:	The evaluator checked the [CC_Guide] and determined that the Sections of “Audit Logging” and “Trusted Channel” describe audit server protocol (i.e. SFTH), configuration of SSH protection, generation SSH client public/private key pair, import remote SSH server’s public key to trusted store, etc.

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

Findings:	The evaluator checked the [CC_Guide] and found that the Section of “Audit Logging” describes the relationship between local audit data and the audit data that are sent to audit log server.

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

Findings:	The evaluator checked the [CC_Guide] and determined that the behaviour related to FAU_STG_EXT.1.3 is not configurable, and this is consistent with ST.

3.1.3.3 Tests

23 Testing of the trusted channel mechanism for audit will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. The evaluator shall perform the following additional tests 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.

Findings:	Verification that the data is encrypted is satisfied by FTP_ITC.1 for the logging channel. The logging server uses OpenSSH 7.4p1 as described in the Test Setup. Due to the log-forwarding mechanism used on logging server, the audit records are therefore confirmed to have been successfully received by the audit server whenever the test cases are run.

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

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

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

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

High-Level Test Description
After running the system for several days, show that the logs roll over based on the pre-existing log rotation policy.
Findings: PASS

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

Findings:	The TOE does not claim this functionality.

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

applied. The evaluator shall verify that the transfer of audit data to an external audit server is implemented.

Findings:	The TOE is not a distributed TOE.

3.2 Cryptographic Support (FCS)

3.2.1 FCS_CKM.1 Cryptographic Key Generation

3.2.1.1 TSS

24 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.

Findings:	In Section 6.2.1 of [ST], it specifies the key sizes supported by the TOE; and it also identifies the usage for each of two schemes the TOE supports.

3.2.1.2 Guidance Documentation

25 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.

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable.

3.2.1.3 Tests

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

Key Generation for FIPS PUB 186-4 RSA Schemes

27 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*.

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

a. Random Primes:

- Provable primes
- Probable primes

b. Primes with Conditions:

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

32 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.

33 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

34 and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

35 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.

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

37 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.

38 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

39 for each FFC parameter set and key pair.

Diffie-Hellman Group 14

40 Testing for FFC Schemes using Diffie-Hellman group 14 is done as part of testing in CKM.2.1.

Findings:	To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFRs supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Algorithm Capability	Certificate
ECDSA Key Gen (186-4)	C1875

3.2.2 FCS_CKM.2 Cryptographic Key Establishment

3.2.2.1 TSS

41 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.

42 If Diffie-Hellman group 14 is selected from FCS_CKM.2.1, the TSS shall affirm that the TOE implements RFC 3526 Section 3.

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

Scheme	*SFR*	*Service*
RSA	*FCS_TLSS_EXT.1*	*Administration*
ECDH	*FCS_SSHC_EXT.1*	*Audit Server*
Diffie-Hellman (Group 14)	*FCS_SSHC_EXT.1*	*Backup Server*
ECDH	*FCS_IPSEC_EXT.1*	*Authentication Server*

44 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.

Findings:	In Section 6.2.2 of [ST], it specifies key establishment schemes TOE supports and the usage/service of each of those schemes.

3.2.2.2 Guidance Documentation

45 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable.

3.2.2.3 Tests

Key Establishment Schemes

46 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

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

Function Test

48 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.

49 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.

50 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.

51 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.

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

Validity Test

53 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.

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

55 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 schemes

56 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.

Diffie-Hellman Group 14

57 The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 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 Diffie-Hellman group 14.

FFC Schemes using "safe-prime" groups

58 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.

Findings:	To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standards, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Algorithm Capability	Certificate
KAS-ECC	C1875

3.2.3 FCS_CKM.4 Cryptographic Key Destruction

3.2.3.1 TSS

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

Findings:	In Section 6.2.3 and Table 15 of [ST], it lists all relevant keys with their origin, storage, and key destruction method.

60 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).

Findings:	In Table 15 of [ST], it describes destruction of keys stored in plaintext in non-volatile memory and interface which is invoked to destroy keys.

61 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.

Findings:	The mechanism by which the TOE destroys plaintext keys in non-volatile memory is described in Table 15 of the ST.

62 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.

Findings:	As per Table 15 of [ST], keys are stored in plaintext.

63 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

[1] Where keys are stored encrypted or wrapped under another key then this may need to be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions.

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.

Findings:	The TSS does not identify any configuration or circumstances that may not conform to the key destruction requirement.

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

Findings:	The TOE does not claim this selection.

3.2.3.2 Guidance Documentation

65 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.

66 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[2] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable. Furthermore, in [CC_Guide], it does not identify any situations that could prevent or delay key destruction or might not strictly conform to the key destruction requirement.

3.2.4 FCS_COP.1/DataEncryption Cryptographic Operation (AES Data Encryption/Decryption)

3.2.4.1 TSS

67 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.

Findings:	In Section 6.2.4 of [ST], it identifies the AES key sizes and mode supported by the TOE.

[2] 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).

3.2.4.2 Guidance Documentation

68 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.

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable.

3.2.4.3 Tests

AES-CBC Known Answer Tests

69 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.

70 **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.

71 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.

72 **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.

73 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.

74 **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].

75 To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key 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.

76 **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].

77 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

78 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.

79 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

80 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]
```

81 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.

82 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

83 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.

84 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.

85 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.

86 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

87 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. Since the Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

88 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.

89 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.

90 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.

91 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].

92 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

93 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

94 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]
```

95 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.

96 There is no need to test the decryption engine.

Findings:	To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Algorithm Capability	Certificate
AES-CTR	C1875

3.2.5 FCS_COP.1/SigGen Cryptographic Operation (Signature Generation and Verification

3.2.5.1 TSS

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

Findings:	In Section 6.2.5 of [ST], it specifies the crypto algorithm and key size for signature services.

3.2.5.2 Guidance Documentation

98 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.

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable.

3.2.5.3 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

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

102 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

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

public key *e* values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

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

Findings:	To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Algorithm Capability	Certificate
ECDSA Sig Gen (186-4) ECDSA Sig Ver (186-4)	C1875

3.2.6 FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)

3.2.6.1 TSS

105 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.

Findings:	In Section 6.2.6 of [ST], it lists other TSF crypto functions associated with the hash function.

3.2.6.2 Guidance Documentation

106 The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable.

3.2.6.3 Tests

107 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.

108 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

109 The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0

to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

112 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

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

Findings:	To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Algorithm Capability	Certificate
SHA-1, SHA-256, SHA-512	C1875

3.2.7 FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithm)

3.2.7.1 TSS

114 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.

Findings:	In Section 6.2.7 of [ST], it specifies key length, hash function used, block size, and MAC length for HMAC function.

3.2.7.2 Guidance Documentation

115 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.

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable.

3.2.7.3 Tests

116 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.

Findings:	To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity.

Algorithm Capability	Certificate
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	C1875

3.2.8 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

117 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [NDcPP].

3.2.8.1 TSS

118 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.

Findings:	In Section 6.2.9 of [ST], it specifies the DRBG type, entropy source seeding the DRBG, and assumed min-entropy supplied in the seed value.

3.2.8.2 Guidance Documentation

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

Findings:	As per section of "Cryptography" in [CC_Guide], cryptographic settings are not configurable.

3.2.8.3 Tests

120 The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

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

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

123 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.

Findings:	To demonstrate that all cryptographic requirements are satisfied, the Assurance Activity Report must clearly indicate all SFRs for which a CAVP certificate is claimed and include, at a minimum, the cryptographic operation, the NIST standard, the SFR supported, the CAVP algorithm list name (e.g. AES, KAS, CVL, etc.) and the CAVP Certificate number. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL

	verification of the NIST validation will constitute performance of the associated assurance activity.

Algorithm Capability	Certificate
DRBG	C1875

3.3 Identification and Authentication (FIA)

3.3.1 FIA_AFL.1 Authentication Failure Management

3.3.1.1 TSS

124 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.

Findings:	In Section 6.3.5 of [ST], it describes TOE's capability of tracking authentication failures of remote administrators, the method by which the remote administrator is locked out, and actions to restore the ability of logging in to the TOE remotely.

125 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).

Findings:	In Section 6.3.5 of [ST], it states that the local console does not implement the lockout mechanism.

3.3.1.2 Guidance Documentation

126 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.

Findings:	The evaluator checked [CC_Guide] and determined that, in section of "Administrator Authentication", it describes how to configure the authentication failure limit and the lockout time period when the limit has been reached.

127 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.

Findings:	The evaluator checked [CC_Guide] and determined that, in section of "Administrator Authentication", it claims that the local console does not enforce account locking.

3.3.1.3 Tests

128 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.

High-Level Test Description
Using the identified TSFI, set the lock parameters to 3 attempts, and 1 minute. Log into an interface twice with the wrong password and then correctly on the third and show the lock is not engaged. Log into the interface with the wrong password 3 times and then on the 4th attempt, with the correct password, and show the lock is engaged. Using a second IP address, show that the user cannot log in from a different IP address. Show the user can log in from the local console without being affected by the lock. Wait 1 minute since the last authentication attempt. At the end of 1 minute, try to login again with the correct password and show it is successful.
Findings: PASS

3.3.2 FIA_PMG_EXT.1 Password Management

3.3.2.1 TSS

129 The evaluator shall examine the TSS to determine that it contains the lists of the supported special character(s) and minimum and maximum number of charters supported for administrator passwords.

Findings:	In Section 6.3.1 of [ST], it specifies what characters make up the password and password length.

3.3.2.2 Guidance Documentation

130 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.

Findings:	The evaluator checked the [CC_Guide] and determined that password policy is described in the Section of “Administrator Authentication”.

3.3.2.3 Tests

131 The evaluator shall perform the following tests.

132 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.

High-Level Test Description
Change the password length to be 15 characters. Change the password for the lteuser user using the identified TSFI to use some of all of the claimed characters. Show that the password can be used to login to the other interfaces. Change the password for the built-in lteuser back to a known good password.
Findings: PASS

133 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.

High-Level Test Description
Change the password length to be 9 characters. Change the password for the lteuser to be only 8 characters and show it is rejected.
Findings: PASS

3.3.3 FIA_UIA_EXT.1 User Identification and Authentication

3.3.3.1 TSS

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

Findings:	In Section 6.3.2, it identified logon methods TOE supports and describes logon process for each logon method.

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

Findings:	In Section 6.3.2, it describes what actions are allowed prior to user identification and authentication.

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

Findings:	The TOE is not a distributed TOE.

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

Findings:	The TOE is not a distributed TOE.

3.3.3.2 Guidance Documentation

138 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.

Findings:	The evaluator checked the [CC_Guide] and determined that any necessary preparatory steps, such as loading trusted user public keys, are described in the Section of “Administrator Authentication”.

3.3.3.3 Tests

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

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

High-Level Test Description
Log into the identified management interface using a known-good credential and logout.

High-Level Test Description
Attempt to log into the identified management interface using a known-bad credential and show it is not permitted. Log into the identified management interface using an unknown username and password and show it is not permitted. Ensure the appropriate audit messages appear.
Findings: PASS

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.

High-Level Test Description
The device does not have any services configured prior to I&A. All claimed services available to remote entities are identified as part of AVA_VAN.1 test scanning.
Findings: PASS

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.

High-Level Test Description
The device does not have any services configured prior to I&A. All claimed services available to local entities are identified as part of AVA_VAN.1 test scanning.
Findings: PASS

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

Findings:	The TOE is not a distributed TOE

3.3.4 FIA_UAU_EXT.2 Password-based Authentication Mechanism

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

3.3.5 FIA_UAU.7 Protected Authentication Feedback

3.3.5.1 Guidance Documentation

141 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.

Findings:	The evaluator examined the [CC_Guide] and [AGD] and determined that there are NOT any necessary preparatory steps to ensure authentication data is not revealed while entering for each local login allowed; in other words, it is enforced by default.

3.3.5.2 Tests

142 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.

High-Level Test Description
Log into the local management interface. Ensure the password field does not echo characters.
Findings: PASS

3.4 Security management (FMT)

3.4.1 General requirements for distributed TOEs

3.4.1.1 TSS

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

Findings:	The TOE is not a distributed TOE.

3.4.1.2 Guidance Documentation

144 For distributed TOEs it is required to verify the Guidance Documentation to describe management of each TOE component. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

Findings:	The TOE is not a distributed TOE.

3.4.1.3 Tests

145 Tests defined to verify the correct implementation of security management functions shall be performed for every TOE component. For security management functions that are implemented centrally, sampling should be applied when defining the evaluator's tests (ensuring that all components are covered by the sample).

Findings:	The TOE is not a distributed TOE.

3.4.2 FMT_MOF.1/ManualUpdate

3.4.2.1 TSS

146 For distributed TOEs see chapter 4.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings:	The TOE is not a distributed TOE.

3.4.2.2 Guidance Documentation

147 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).

Findings:	The evaluator checked the [CC_Guide] and determined that manual update is described in section of “Updating the TOE”.

148 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).

Findings:	The TOE is not a distributed TOE.

3.4.2.3 Tests

149 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 shall fail.

150 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.

High-Level Test Description
This test case is covered in FPT_TUD_EXT.1.
Findings: PASS

3.4.3 FMT_MTD.1/CoreData Management of TSF Data

3.4.3.1 TSS

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

Findings:	As per Section 6.4.2 of [ST], users are required to login before being provided with access to any administrative functions.

152 If 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.

Findings:	N/A - SFRs of FIA_X509_EXT family are not claimed.

3.4.3.2 Guidance Documentation

153 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.

Findings:	The evaluator checked the user guide and was able to identify each of TSF-data-manipulating functions, such as: Updating the TOE ([CC_Guide]) Setting Times ([CC_Guide]) Audit Logging ([CC_Guide]) Administrator Authentication ([CC_Guide]) Trusted Channel ([CC_Guide]) Local cli user password change (Confd CLI User Guide of [AGD])

154 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.

Findings:	The TOE does not support handling of X.509v3 certificates.

3.4.4 FMT_SMF.1 Specification of Management Functions

155 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/ManualUpdate, FMT_MOF.1/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/Services, and FMT_MOF.1/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.

3.4.4.1 TSS (containing also requirements on Guidance Documentation and Tests)

156 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).

Findings:	In Section 6.4.5 of [ST]. it lists security management functions supported by the TOE and which security management functions are accessible to which interfaces.

157 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.

Findings:	In Section 6.4.5 of [ST], it describes security management functions accessible via local administrative interface.

158 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.

Findings:	The TOE is not a distributed TOE.

3.4.4.2 Guidance Documentation

159 See section 3.4.4.1.

3.4.4.3 Tests

160 The evaluator tests management functions as part of testing the SFRs identified in section 4.4.4. 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.

Note:	There are no explicit test activities and therefore none are recorded here. All management functions in FMT_SMF.1.1 were exercised in other test cases.

3.4.5 FMT_SMR.2 Restrictions on security roles

3.4.5.1 TSS

161 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.

Findings:	In Section 6.4.3 of [ST], it identifies the TOE supported roles with any restrictions of the roles.

3.4.5.2 Guidance Documentation

162 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.

Findings:	The evaluator verified that [CC_Guide] identifies interfaces for administering the TOE both locally and remotely in section of “Administration Interfaces”. By following instructions in sections of “Administrator Authentication”, “Audit Logging”, and CONFD CLI User Guide in [AGD], the evaluator was able to complete all tests successfully.

3.4.5.3 Tests

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

Findings:	There are no explicit test activities and therefore none are recorded here. All interfaces are tested throughout this test plan.

3.5 Protection of the TSF (FPT)

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

3.5.1.1 TSS

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

Findings:	In Section 6.5.1 of [ST], it describes that any pre-shared keys, symmetric keys, and private keys are stored in plaintext; it further states that, in all cases, plaintext keys cannot be viewed through an interface designed specifically for that purpose.

3.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

3.5.2.1 TSS

165 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.

Findings:	In Section 6.5.2 of [ST], it states that the SHA-512 hashed passwords are kept in the storage and they are unable to be viewed through an interface designed specifically for that purpose**.**

3.5.3 FPT_TST_EXT.1 TSF testing

3.5.3.1 TSS

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

Findings:	In Section 6.5.3 of [ST], it describes self-tests that are run by the TSF. The evaluator determines that the tests include crypto module self-tests, TOE image integrity test and CPU and BIOS self-tests, which are sufficient to demonstrate that TSF is operating correctly.

167 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.

Findings:	The TOE is not a distributed TOE.

3.5.3.2 Guidance Documentation

168 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.

Findings:	The evaluator checked the [CC_Guide] and determined that the section of "Power-on Self-tests" (section 2.3) describes possible errors from self-testing and actions following errors. The evaluator considered the error states described in the [CC_Guide] in section 2.3: - Cryptographic functions of the TOE; - CPU and BIOS tests; and - Integrity of the TOE image. Each of the failure states of these self-tests in the [CC_Guide] matches those failure states described in the [ST] TSS section 6.5.3. The administrator is required to follow "…any troubleshooting steps indicated by an error message" and if such steps are unsuccessful, "Samsung Technical Support should be engaged for troubleshooting or repair." The administrator actions are appropriate given the perceived severity of the error states and the complexity of the product.

169 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.

Findings:	The TOE is not a distributed TOE.

3.5.3.3 Tests

170 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 fulfil any of the SFRs.

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

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

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

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

173 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.

High-Level Test Description
Reset the TOE and witness that the startup includes an indicator that self-tests were executed and passed permitting the device to operate.
Findings: PASS

3.5.4 FPT_TUD_EXT.1 Trusted Update

3.5.4.1 TSS

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

Findings:	In Section 6.5.4 of [ST], it describes how to query the currently active version. It does not claim that a trusted update can be installed with a delayed activation.

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

Findings:	In Section 6.5.4 of [ST], it describes the TSF software update mechanism. Especially, digital signature verification is used to check the authenticity of the software update – how digital signature verification is done and what happens if the verification fails.

176 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.

Findings:	The options “support automatic checking for updates” or “support automatic updates” are not claimed.

177 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.

Findings:	The TOE is not a distributed TOE.

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

Findings:	N/A – published hash is not used for TOE's trusted update mechanism

3.5.4.2 Guidance Documentation

179 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.

Findings:	The evaluator checked the [CC_Guide] and found that section of "Verifying the TOE" describes how to query the currently active version; and it claims that the TOE does not support delayed activation in section of "Updating the TOE".

180 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.

Findings:	The evaluator checked the [CC_Guide] and found that it claims that TOE update files are digitally signed (ECDSA using NIST P-256 / SHA-256) and the signature is verified using a hardcoded ECDSA public key prior to installation of the update; and if verification fails, the update is aborted, and an error message is displayed.

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

Findings:	The TOE does not rely on hash-based integrity mechanisms.

182 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 System Administrator; it does not need to give information about the internal communication that takes place when applying updates.

Findings:	The TOE is not a distributed TOE.

183 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.

Findings:	The TOE is not a distributed TOE.

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

Findings:	The TOE does not use certificate-based mechanism.

3.5.4.3 Tests

185 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.

High-Level Test Description
Get the current version of the TOE. Attempt to install a legitimate version of the TOE. After the install, get the current version of the TOE and ensure it is consistent with the newly installed version.
Findings: PASS

b. Test 2 [conditional]:

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

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

2) An image that has not been signed

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

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

High-Level Test Description
Attempt to install a bad image, an unsigned image and a badly signed image – i.e. bad signature. After each attempt, get the current version of the TOE using all available means and ensure they are consistent.
Findings: PASS

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

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

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

that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

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

Findings:	Not Applicable

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

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

Findings:	The TOE only supports manual updates.

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

Findings:	The TOE is not a distributed TOE.

3.5.5 FPT_STM_EXT.1 Reliable Time Stamps

3.5.5.1 TSS

189 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.

Findings:	In Section 6.5.5 of [ST], it lists each security function that uses reliable timestamp; it also describes how the reliable time is maintained.

3.5.5.2 Guidance Documentation

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

Findings:	The evaluator checked the [CC_Guide] and determined that manual date setting and synchronization with NTP server are described in the section of “Setting Time”.

3.5.5.3 Tests

191 The evaluator shall perform the following tests:

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

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

High-Level Test Description
Get the current date. Change the date. Get the date to show that it was successful. Verify the audit log contains the necessary information. Refer to FCS_NTP_EXT.1 for Test 2.
Findings: PASS

192 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.

Findings:	Not Applicable – the TOE is not a distributed TOE

3.6 TOE Access (FTA)

3.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

3.6.1.1 TSS

193 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.

Findings:	In Section 6.6.1 of [ST], it describes that local admin session is terminated after a period of time of inactivity. It also describes that the Security Administrator may configure the TOE to terminate an inactive local interactive session following a specified period of time.

3.6.1.2 Guidance Documentation

194 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.

Findings:	The evaluator checked the [CC_Guide] and confirmed that configuration of local console session locking or termination is described in the section of “Administrator Authentication”.

3.6.1.3 Tests

195 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 re-authentication is needed when trying to unlock the session.

High-Level Test Description
For various inactivity times, specifically 1 minute and 2 minutes, do the following: Change the idle timeout to this value; Log into the device; Wait for the full duration of the timeout without sending any keep alives. The session should terminate.
Findings: PASS

3.6.2 FTA_SSL.3 TSF-initiated Termination

3.6.2.1 TSS

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

Findings:	In Section 6.6.2 of [ST], it describes the remote admin session termination after specific period of time inactivity. It also describes that the Security Administrator may configure the TOE to terminate an inactive remote interactive session following a specified period of time.

3.6.2.2 Guidance Documentation

197 The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

Findings:	The evaluator checked the [CC_Guide] and confirmed that configuration of remote administrative session locking or termination is described in the section of “Administrator Authentication”.

3.6.2.3 Tests

198 For each method of remote administration, the evaluator shall perform the following test:

a. Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive

session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

High-Level Test Description
For various inactivity times, specifically 5 minutes and 6 minutes, do the following: Change the idle timeout to this value; Log into the device; Wait for the full duration of the timeout. The session should terminate.
Findings:

3.6.3 FTA_SSL.4 User-initiated Termination

3.6.3.1 TSS

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

Findings:	As per Section 6.6.3 of [ST], administrative users may terminate their own sessions at any time.

3.6.3.2 Guidance Documentation

200 The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Findings:	The evaluator checked the [CC_Guide] in section 3.2 and found that the 'exit' command can be used to terminate all variants of shell access. In addition, the "close-session" RPC command can be used to terminate remote NetconfD access.

3.6.3.3 Tests

201 For each method of remote administration, the evaluator shall perform the following tests:

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

High-Level Test Description
Log into the serial console. Log out using the TSFI previous discussed. Verify that the session has been terminated.
Findings: PASS

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

High-Level Test Description
Log into the SSH CLI interface.

High-Level Test Description
Log out using the TSFI previous discussed. Log into the SHS/Netconf CLI interface. Log out using the TSFI previous discussed. Log into the SHS/Netconf interface. Log out using the TSFI previous discussed. In all cases show that the session is terminated.
Findings: PASS

3.6.4 FTA_TAB.1 Default TOE Access Banners

3.6.4.1 TSS

202 The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access and might be configured during initial configuration (e.g. via configuration file).

Findings:	In Section 6.6.4 of [ST], it describes that the TOE is displaying an advisory notice and a consent warning message for all administrative methods of access.

3.6.4.2 Guidance Documentation

203 The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Findings:	The evaluator checked the [CC_Guide] and found that configuration of banner is described in the section of “Administrator Authentication”.

3.6.4.3 Tests

204 The evaluator shall also perform the following test:

a. Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

High-Level Test Description
Log into the SSH CLI. Change the local and remote banner to a random string. Log into fresh SSH and local CLI sessions and show that the banner was modified and is presented prior to I&A.
Findings: PASS

3.7 Trusted path/channels (FTP)

3.7.1 FTP_ITC.1 Inter-TSF trusted channel

3.7.1.1 TSS

205 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.

Findings:	In Section 6.7.1 of [ST], it identifies only one secure communication with authorized IT entities, i.e. secure communication with external audit server, and in this case, the TOE acts as SSH client.

3.7.1.2 Guidance Documentation

206 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.

Findings:	The evaluator checked the [CC_Guide] and found that TOE's connection to remote audit server is protected over SSH and the section of "Trusted Channel" covers establishment of the SSH protocol with remote audit server.

3.7.1.3 Tests

207 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.

208 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.

Note:	The TOE maintains trusted channels to the remote audit log, which is set up as per the evaluated configuration. It is constantly tested throughout the evaluation.

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.

High-Level Test Description
Engage wireshark over the appropriate interface.
Execute the log uploading function on the TOE.
Examine wireshark and verify that the traffic is encrypted and it is initiated by the TOE.

High-Level Test Description
Findings: PASS

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

High-Level Test Description
The only a trusted communication channel is one to remote audit server. See previous test case.
Findings: PASS

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.

High-Level Test Description
Engage wireshark over the logging interface. Physically disconnect the remote logging server by disconnecting the physical cable at the switch (rather than at the TOE side) while ensuring the trusted channel attempts to start. Immediately plug the device back in. This interrupts the MAC layer. Ensure the traffic starts back up without too much trouble. Repeat, but this time disconnect for 5 minutes. Show that the traffic fails but no information is leaked in plaintext.
Findings: PASS

209 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.

210 The vendor 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.

Findings:	Not Applicable

3.7.2 FTP_TRP.1/Admin Trusted Path

3.7.2.1 TSS

211 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.

Findings:	In Section 6.7.1 of [ST], it identifies the methods of remote TOE administration and how those communications are secured.

3.7.2.2 Guidance Documentation

212 The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Findings:	The evaluator checked the [CC_Guide] in section 3.2 and found that administration interfaces are all over SSH. By following standard way of connecting to SSH server, the evaluator was able to connect to remote Bash shell over SSH and netconfd CLI over SSH.

3.7.2.3 Tests

213 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.

Note:	The only trusted paths are Bash CLI over SSH, Netconfd CLI SSH, and NetconfSSH over SSH, which are all set up as per the evaluated configuration. They are constantly tested throughout the evaluation.

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

High-Level Test Description
Engage wireshark over the appropriate interface. Log into the trusted path. Examine wireshark and verify that the trusted path sends encrypted traffic.
Findings: PASS

214 Further assurance activities are associated with the specific protocols.

215 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.

Findings:	Not Applicable as the TOE is not a distributed TOE.

4 Evaluation Activities for Selection-Based Requirements

4.1 Cryptographic Support (FCS)

4.1.1 FCS_NTP_EXT.1 NTP Protocol

4.1.1.1 TSS

FCS_NTP_EXT.1.1

216 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.

Findings:	In Section 6.2.8 of [ST], it identifies that NTP version 4 is supported; TOE uses SHA-1 authentication to ensure the integrity of the timestamp.

217 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.

Findings:	As per Section 6.2.8 of [ST], TOE supports one method of synchronization with the NTP server, as described above.

4.1.1.2 Guidance Documentation

FCS_NTP_EXT.1.1

218 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.

Findings:	The evaluator checked [CC_Guide] and [AGD] and found that they provide guidance on configuration of NTP in Section of "Setting Time" of [CC_Guide] and "ntp-info" in [AGD] (AU/DU Command Reference).

FCS_NTP_EXT.1.2

219 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.

Findings:	The evaluator checked [CC_Guide] and [AGD] and found that they provide guidance on configuration of NTP in Section of "Setting Time" of [CC_Guide] and "ntp-info" in [AGD] (AU/DU Command Reference); however, the authentication algorithm is not configurable. The pre-shared NTP key can be set with "set-ntp-update-key" command.

Assurance Activity Note:

220 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 Security Administrator how to configure the TOE to use the chosen option(s).

Findings:	The TOE only supports SHA-1 and it is not configurable.

FCS_NTP_EXT.1.3

221 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.

Findings:	By default, the TOE does not accept broadcast and multicast NTP packets.

4.1.1.3 Tests

FCS_NTP_EXT.1.1

222 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.

High-Level Test Description
Configure the TOE to have a preshared key consistent with the NTP server. Reboot the TOE and show the time is picked up correctly and verify that the NTP version is correct.
Findings: PASS

FCS_NTP_EXT.1.2

223 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.

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

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

226 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.

High-Level Test Description
Configure the TOE to have a preshared key consistent with the NTP server. Reboot the TOE and show the time is picked up correctly. Configure the NTP server to use an integrity mechanism different than that of the TOE. Reboot the TOE. Show that the TOE does not pick up the new time.
Findings: PASS

FCS_NTP_EXT.1.3

227 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.

High-Level Test Description
Reboot the TOE and show that the TOE (attempts) to query the NTP server, but is rejected and that the time is not set when broadcast and multicast requests are transmitted by the NTP server instead.
Findings: PASS

FCS_NTP_EXT.1.4

228 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.

High-Level Test Description
Reboot the TOE and show that the TOE queries each of the configured NTP servers. Only one of the servers will be configured to return back a valid timestamp which we will show is received by the TOE.
Findings: PASS

229 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.

High-Level Test Description
Configure a test machine to send spoofed NTP messages to the TOE and show that the TOE does not use the time information contained therein.
Findings: PASS

4.1.2 FCS_SSHC_EXT.1 SSH Client

4.1.2.1 TSS

FCS_SSHC_EXT.1.2

230 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication and that this list conforms to FCS_SSHC_EXT.1.5. and ensure that if password-based authentication methods have been selected in the ST then these are also described.

Findings:	In Section 6.2.10 of [ST], it describes that the public key algorithms ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 are used for authentication. The TOE does not support password authentication if it acts as SSH client.

FCS_SSHC_EXT.1.3

231 The evaluator shall check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

Findings:	In Section 6.2.10 of [ST], it states that TOE drops the packet larger than 256KB in size.

FCS_SSHC_EXT.1.4

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

Findings:	In Section 6.2.10 of [ST], it specifies the encryption algorithms supported by the TOE. The encryption algorithms specified are identical to those listed for this component.

FCS_SSHC_EXT.1.5

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

Findings:	In Section 6.2.10 of [ST], it specifies the public key algorithms supported by the TOE. The public key algorithms specified are identical to those listed for this component.

234 If x509v3-based public key authentication algorithms are claimed, the evaluator shall confirm that the TSS includes the description of how the TOE establishes the server’s identity and how this identity is confirmed with the one that is presented in the provided certificate. For example, the TOE could verify that a server’s configured IP address matches the one presented in the server’s x.509v3 certificate.

Findings:	N/A – x509-v3-based public key authentication algorithms are not claimed.

FCS_SSHC_EXT.1.6

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

Findings:	In Section 6.2.10 of [ST], it specifies the data integrity algorithms supported by the TOE. The data integrity algorithms specified are identical to those listed for this component.

FCS_SSHC_EXT.1.7

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

Findings:	In Section 6.2.10 of [ST], it specifies key exchange algorithms supported by the TOE. The key exchange algorithms specified are identical to those listed for this component.

FCS_SSHC_EXT.1.8

237 The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

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

Findings:	In Section 6.2.10 of [ST], it specifies both time-based and traffic-based thresholds. It also describes that rekeying is done whichever threshold is hit first.

4.1.2.2 Guidance Documentation

FCS_SSHC_EXT.1.4

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

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHC_EXT.1.5

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

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHC_EXT.1.6

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

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHC_EXT.1.7

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

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHC_EXT.1.8

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

Findings:	As per section 3.8 “Trusted Channel” of [CC_Guide], the rekey thresholds are not configurable in the evaluated configuration.

4.1.2.3 Tests

FCS_SSHC_EXT.1.2

243 Test 1: If password-based authentication methods have been selected in the ST then using the guidance documentation, the evaluator shall configure the TOE to perform password-based authentication to an SSH server and demonstrate that a Security Administrator can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.

244 Note: Public key authentication is tested as part of testing for FCS_SSHC_EXT.1.5.

High-Level Test Description
Configure the TOE for password-based authentication for the SSH client and show that providing the correct password the Security Administrator logs into the remote target host.
Findings: PASS

FCS_SSHC_EXT.1.3

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

High-Level Test Description
Using a custom server tool, force the TOE to receive a packet larger than the expected TOE buffer size and show that the TOE rejects the packet in some way.
Findings: PASS

FCS_SSHC_EXT.1.4

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

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

High-Level Test Description
Connect TOE SSH client to a test SSH server and capture the TOE client's advertised supported cipher algorithms. Verify that the advertised set matches the claimed set. Forcibly use an SSH server to permit connections from the TOE client using only one of those claimed ciphers and show that the connection is successful.
Findings: PASS

FCS_SSHC_EXT.1.5

247 Test 1: The evaluator shall establish an SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test. Test objective: The purpose of this positive test is to check the authentication of the server by the client (when establishing the transport layer connection), and not for checking generation of the authentication message from the client (in the User Authentication Protocol). The evaluator shall therefore establish sufficient separate SSH connections (with an appropriately configured server) to cause the TOE to demonstrate use of all public key algorithms claimed in FCS_SSHC_EXT.1.5 in the ST.

High-Level Test Description
Use the TOE client and connect to a test SSH server which only provides a host key using the specified public key algorithms in turn. Show that the client authenticates the host.
Findings: PASS

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

High-Level Test Description
Using the non-TOE SSH server with a host key algorithm not claimed by the TOE, try to connect the TOE to the non-TOE server and show that it is not permitted.
Findings: PASS

FCS_SSHC_EXT.1.6

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

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

High-Level Test Description
Using an SSH Server, forcibly permit only the claimed integrity algorithms and show that connections by the TOE SSH client are accepted to form successful connections.
Findings: PASS

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

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

High-Level Test Description
Using an SSH Server, forcibly permit an integrity algorithm which is not claimed by the TOE and show that a TOE SSH client connection attempt results in a failed connection.
Findings: PASS

FCS_SSHC_EXT.1.7

253 Test 1: The evaluator shall configure an SSH server to permit all allowed key exchange methods. The evaluator shall attempt to connect from the TOE to the SSH server using each allowed key exchange method and observe that each attempt succeeds.

High-Level Test Description
Using an SSH server, forcibly permit only one claimed key exchange mechanism at a time and show that the TOE client will successfully connect using that algorithm.
Findings: PASS

FCS_SSHC_EXT.1.8

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

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

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

257 For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH server and shall transmit data to and/or receive data from the TOE within

the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHC_EXT.1.8).

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

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

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

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

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

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

High-Level Test Description
Using a custom SSH server, use the TOE client to connect to the server and trickle data over the channel to avoid disconnection due to idle timeout. Ensure that the TOE rekeys before 1 hour has elapsed. Ensure that the TOE is responsible for sending the rekey initiation. Using a custom SSH server, permit the TOE client to connect to the server. The server will send large amounts of data over the channel back to the client. Ensure that the TOE rekeys before 1 GB in the aggregate has been transmitted. Ensure that the TOE is responsible for sending the rekey initiation.
Findings: PASS

FCS_SSHC_EXT.1.9

262 Test 1: The evaluator shall delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator shall initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the Security Administrator to accept or deny the key before continuing the connection.

High-Level Test Description
Clear the known host key database. Using the TOE SSH client, connect to an SSH server and show that the TOE either warns the administrator that the host is unknown or that it rejects the connection attempt until after the host key has been manually added.

High-Level Test Description
Findings: PASS

263 Test 2: The evaluator shall add an entry associating a host name with a public key into the TOE's local database. The evaluator shall replace, on the corresponding SSH server, the server's host key with a different host key. If 'password-based' is selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords). If 'password-based' is not selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using public key-based authentication and shall ensure that the TOE rejects the connection.

High-Level Test Description
Add a host key to the known hosts database either explicitly or implicitly depending on the mechanism for inserting the key. Generate a different host key for the non-TOE SSH server. Using the TOE SSH client, connect to the SSH server that advertises the wrong host key and show that the TOE rejects the connection. Verify that passwords are not transmitted; verify public key authentication fails.
Findings: PASS

4.1.3 FCS_SSHS_EXT.1 SSH Server

4.1.3.1 TSS

FCS_SSHS_EXT.1.2

264 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication and that this list conforms to FCS_SSHS_EXT.1.5. and ensure that if password-based authentication methods have been selected in the ST then these are also described

Findings:	In Section 6.2.11 of [ST], it describes that the public key algorithms ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 are used for authentication. The TOE also supports password authentication if it acts as SSH server, and password authentication description can be found in Section 6.3.3 for authentication process (including password authentication process), Section 6.3.1 for password composition criteria, Section 6.3.2 for actions allowed/available prior to authentication, and Section 6.3.5 for reaction to failed authentication attempts.

FCS_SSHS_EXT.1.3

265 The evaluator shall check that the TSS describes how "large packets" in terms of RFC 4253 are detected and handled.

Findings:	In Section 6.2.11 of [ST], it states that TOE drops the packet larger than 256KB in size.

FCS_SSHS_EXT.1.4

266 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to

ensure that the encryption algorithms specified are identical to those listed for this component.

Findings:	In Section 6.2.11 of [ST], it specifies the encryption algorithms supported by the TOE. The encryption algorithms specified are identical to those listed for this component.

FCS_SSHS_EXT.1.5

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

Findings:	In Section 6.2.11 of [ST], it specifies the public key algorithms supported by the TOE. The public key algorithms specified are identical to those listed for this component.

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

Findings:	In Section 6.2.11 of [ST], it describes that TOE maintains authorized public keys in a local database.

FCS_SSHS_EXT.1.6

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

Findings:	In Section 6.2.11 of [ST], it specifies the data integrity algorithms supported by the TOE. The data integrity algorithms specified are identical to those listed for this component.

FCS_SSHS_EXT.1.7

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

Findings:	In Section 6.2.11 of [ST], it specifies key exchange algorithms supported by the TOE. The key exchange algorithms specified are identical to those listed for this component.

FCS_SSHS_EXT.1.8

271 The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

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

Findings:	In Section 6.2.11 of [ST], it specifies both time-based and traffic-based thresholds. It also describes that rekeying is done whichever threshold is hit first.

4.1.3.2 Guidance Documentation

FCS_SSHS_EXT.1.4

272 The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the

TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHS_EXT.1.5

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

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHS_EXT.1.6

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

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHS_EXT.1.7

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

Findings:	As per Section of “Cryptography” in [CC_Guide], cryptographic settings are not configurable.

FCS_SSHS_EXT.1.8

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

Findings:	As per section 3.8 “Trusted Channel” of [CC_Guide], the rekey thresholds are not configurable in the evaluated configuration.

4.1.3.3 Tests

FCS_SSHS_EXT.1.2

277 Test 1: If password-based authentication methods have been selected in the ST then using the guidance documentation, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication succeeds when the correct password is provided by the user.

High-Level Test Description
Using a SSH client, connect to the TOE with correct password.
Findings: PASS

278 Test 2: If password-based authentication methods have been selected in the ST then the evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

279 Note: Public key authentication is tested as part of testing for FCS_SSHS_EXT.1.5.

High-Level Test Description
Using a SSH client, try to connect to the TOE with incorrect password. The authentication shall fail.
Findings: PASS

FCS_SSHS_EXT.1.3

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

High-Level Test Description
Using a custom tool, transmit a packet larger than the expected TOE buffer size and show that the TOE rejects the packet in some way.
Findings: PASS

FCS_SSHS_EXT.1.4

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

High-Level Test Description
Using an SSH client, connect to the TOE server and capture the TOE server's advertised supported cipher algorithms. Verify that the advertised set matches the claimed set. Forcibly use an SSH client to connect using only one of those ciphers and show that the connection is successful.
Findings: PASS

FCS_SSHS_EXT.1.5

282 Test 1: The evaluator shall establish an SSH connection using each of the public key algorithms specified by the requirement to authenticate the TOE to an SSH client. It

is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Using an SSH client, connect to the TOE server using the specified public key algorithms in turn. This requires the TOE to be loaded with a public key corresponding to the key pair.
Findings: PASS

283 Test 2: The evaluator shall choose one public key algorithm supported by the TOE. The evaluator shall generate a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails. Test objective: The purpose of this negative test is to verify that the server rejects authentication attempts of clients that present a public key that does not match public key(s) associated by the TOE with the identity of the client (i.e. the public keys are unknown to the server). To demonstrate correct functionality, it is sufficient to determine that an SSH connection was not established after using a valid username and an unknown key of supported type.

High-Level Test Description
Using an SSH client, connect to the TOE server using a private key that does not match the public key half loaded in the TOE. The TOE will reject the authentication attempt.
Findings: PASS

284 Test 3: The evaluator shall configure an SSH client to only allow a public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

High-Level Test Description
Generated a public/private key pair of unsupported public key algorithm. Using an SSH client, try to connect using the newly generated private key. The connection attempt should fail.
Findings: PASS

FCS_SSHS_EXT.1.6

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

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

High-Level Test Description
Using an SSH client, forcibly negotiate only the claimed integrity algorithms and show that they are accepted to form a successful connection.
Findings: PASS

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

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

High-Level Test Description
Using an SSH client, forcibly negotiate an integrity algorithm which is not claimed by the TOE and show that it results in a failed connection.
Findings: PASS

FCS_SSHS_EXT.1.7

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

High-Level Test Description
Using an SSH client, forcibly negotiate the diffie-hellman-group1-sha1 key exchange algorithm which is not supported by the TOE and show that it results in a failed connection.
Findings: PASS

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

High-Level Test Description
Using an SSH client, forcibly negotiate each of the claimed key exchange algorithms in turn and show that it results in a successful connection.
Findings: PASS

FCS_SSHS_EXT.1.8

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

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

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

294 For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key

is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8).

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

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

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

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

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

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

High-Level Test Description
Using a custom SSH client, connect to the TOE and trickle data over the channel to avoid disconnection due to idle timeout. Ensure that the TOE rekeys before 1 hour has elapsed. Ensure that the TOE is responsible for sending the rekey initiation. Using a custom SSH client, connect to the TOE send large amounts of data over the channel. Ensure that the TOE rekeys before 1 GB in the aggregate has been transmitted. Ensure that the TOE is responsible for sending the rekey initiation.
Findings: PASS

4.2 Security management (FMT)

4.2.1 FMT_MTD.1/CryptoKeys Management of TSF Data

4.2.1.1 TSS

299 For distributed TOEs see chapter 3.4.1.1.

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

Findings:	The TOE is not a distributed TOE. In Section 6.4.4 of [ST], it lists the keys, i.e. SSH keys, that security administrator is able to manage, including the options which are available, i.e. generating keys and deleting keys.

4.2.1.2 Guidance Documentation

301 For distributed TOEs see chapter 3.4.1.2

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

Findings:	The evaluator checked the [CC_Guide] and found that the section of “Trusted Channel” covers SSH host key generation, SSH client key generation, import of remote trusted server public key, and import of trusted user public key, etc.

4.2.1.3 Tests

303 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.

304 The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

High-Level Test Description
The only Security Administrator with access to the shell is the lteuser. Log in as lteuser, and modify the following crypto keys: Modify the host ssh key and show the key is modified. Modify the SSH client private key and show the key is modified.
Findings: PASS

5 Vulnerability Assessment

5.6.1.1 Evaluation Activity: 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[3] that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating hypotheses during their analysis.

Findings:	The evaluator collected this information from the developer which was used to feed into the Type 1 Flaw Hypotheses search (below).

5.6.1.2 Evaluation Activity: The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

Findings:	The following sources of public vulnerabilities were considered in formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform key-word searches during the evaluation of the TOE. Hypothesis sources for public vulnerabilities were: NIST National Vulnerabilities Database (can be used to access CVE and US-CERT databases identified below): https://web.nvd.nist.gov/view/vuln/search Common Vulnerabilities and Exposures: http://cve.mitre.org/cve/ US-CERT: http://www.kb.cert.org/vuls/html/search Exploit / Vulnerability Search Engine: www.exploitsearch.net SecurITeam Exploit Search: www.securiteam.com Tenable Network Security: http://nessus.org/plugins/index.php?view=search Tipping Point Zero Day Initiative: http://www.zerodayinitiative.com/advisories Offensive Security Exploit Database: https://www.exploit-db.com/ Rapid7 Vulnerability Database: https://www.rapid7.com/db/vulnerabilities

[3] In this sub-section the term “components” refers to parts that make up the TOE. It is therefore distinguished from the term “distributed TOE components”, which refers to the parts of a TOE that are present in one physical part of a distributed TOE. Each distributed TOE component will therefore generally include a number of the hardware and software components that are referred to in this sub-section: for example, each distributed TOE component will generally include hardware components such as processors and software components such as an operating system and libraries.

OpenSSL Vulnerabilities: https://www.openssl.org/news/vulnerabilities.html

Google

Type 1 Hypothesis searches were conducted on September 11, 2020 and included the following search terms (version details have been removed in this public document):

1) Samsung 5G gNB AU, DU

2) OpenSSL

3) OpenSSH

The evaluation team determined that no residual vulnerabilities exist based on these searches that are exploitable by attackers with Basic Attack Potential.

The evaluation team considered the type 2 flaw hypothesis in the NDcPP v2.2E Appendix A.1.2 and determined that they were not applicable to the TOE.

The evaluation team developed Type 3 flaw hypotheses in accordance with Sections A.1.3, A.1.4, and A.2, and no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.

The evaluation team developed Type 4 flaw hypotheses in accordance with Sections A.1.3, A.1.4, and A.2, and no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.