Bivio 6310-NC Security Target

17-4135-R-0035

Version: 1.1

April 20, 2018

Prepared For:

Bivio Networks, Inc.

4457 Willow Rd Suite 240

Pleasanton, CA, 94588

Prepared By:

Brad Mitchell

UL Verification Services Inc.

Notices:

Table of Contents

1. Security Target (ST) Introduction

- The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.
- The ST reference shall uniquely identify the ST.
- The TOE reference shall identify the TOE.

The structure of this document is defined by CC v3.1r4 Part 1 Annex A.2, "Mandatory contents of an ST":

- Section 1 contains the ST Introduction, including the ST reference, Target of Evaluation (TOE) reference, TOE overview, and TOE description.
- Section 2 contains conformance claims to the Common Criteria (CC) version, Protection Profile (PP) and package claims, as well as rationale for these conformance claims.
- Section 3 contains the security problem definition, which includes threats, Organizational Security Policies (OSP), and assumptions that must be countered, enforced, and upheld by the TOE and its operational environment.
- Section 4 contains statements of security objectives for the TOE, and the TOE operational environment as well as rationale for these security objectives.
- Section 5 contains definitions of any extended security requirements claimed in the ST.
- Section 6 contains the security function requirements (SFR), the security assurance requirements (SAR), as well as the rationale for the claimed SFR and SAR.
- Section 7 contains the TOE summary specification, which includes the detailed specification of the IT security functions

1.1 Security Target Reference

The Security Target reference shall uniquely identify the Security Target.

ST Title:	Bivio 6310-NC Security Target
ST Version Number:	Version 1.1
ST Author(s):	Brad Mitchell
ST Publication Date:	April 20, 2018
Keywords	Network Device

1.2 Target of Evaluation Reference

The Target of Evaluation reference shall identify the Target of Evaluation.

TOE Developer	Bivio Networks, Inc.
	4457 Willow Rd Suite 240
	Pleasanton, CA, 94588
TOE Name:	Bivio 6310-NC
TOE Version	0.1

1.3 Target of Evaluation Overview

1.3.1 TOE Product Type

The TOE is classified as a Network Device (a generic infrastructure device that can be connected to a network).

1.3.2 TOE Usage

The Bivio 6310-NC device can be used to run a variety of applications for processing network data. There are many such applications, both commercial and open source. It is out of scope for this certification process to include all these applications for evaluation, so a standard application factory-installed to all Bivio 6310-NC devices as part of the base BiviOS will be provided. This application provides the following non-evaluated functionality:

- Inspects packets, and will either drop them or forward them based on configuration.
- Uses the default mechanisms for packet handling, and represents other packet processing applications that a customer may choose to install.

1.3.3 TOE Major Security Features Summary

- Audit
- Cryptography
- Identification and Authentication
- Security Management
- Protection of the TSF
- TOE Access
- Trusted Path/Channels

1.3.4 TOE IT environment hardware/software/firmware requirements

The TOE requires the following hardware / infrastructure to be present:

- Syslog server conformant to RFCs 5424 (Syslog over TCP),
- A local console with an RS-232 port for use with the Bivio provided console cable.

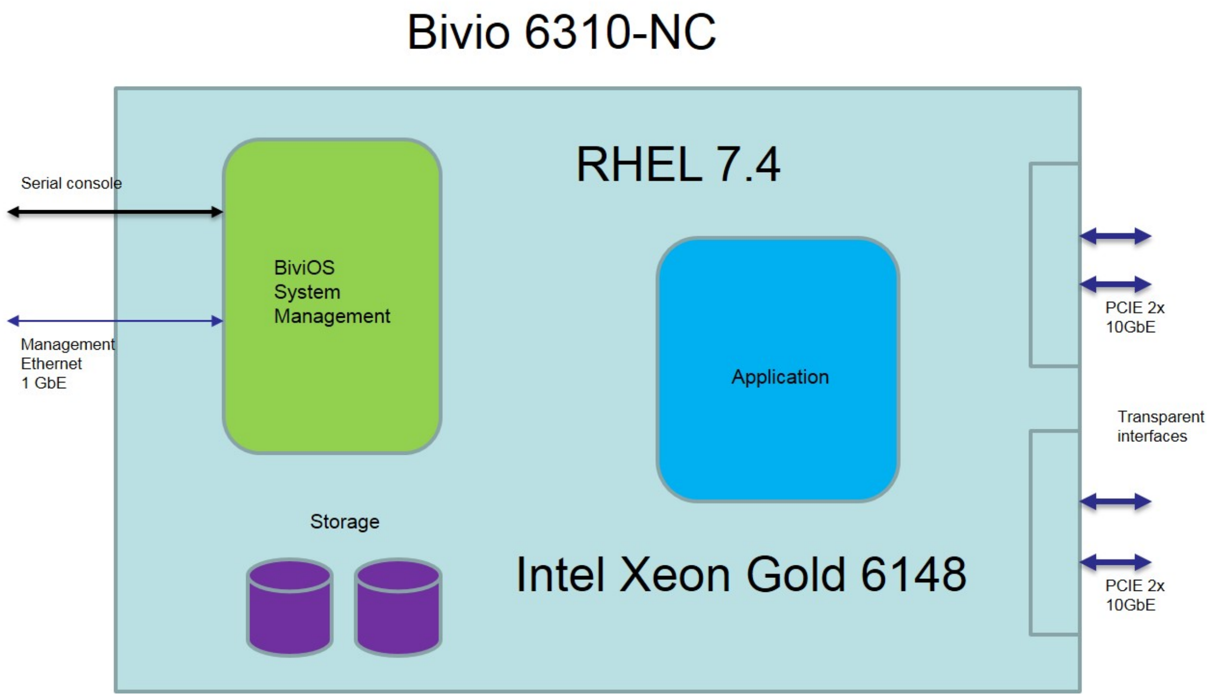
The TOE requires the following software to be present:

- Administrators will need an SSHv2 Client conformant to RFCs 4251, 4252, 4253, 4254, and 6668.
    - The SSHv2 client will need to be capable of supporting AES128-CBC and AES256-CBC encryption algorithms, using HMAC-SHA2-256 or HMAC-SHA2-512 integrity algorithms, and performing key exchange using Diffie-Hellman Group14-SHA1.
    - To perform public key authentication to the TOE, the SSHv2 client will need to be capable of supporting SSH-RSA.
- The TOE also provides a TLS protected server capability, which requires a TLSv1.2 client capable of negotiating one of the following ciphersuites:
    - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

  The client will connect to the TLS server, send standard input over to the TLS server, and print any data received from the server on to standard output.

1.4 Target of Evaluation Description

The Bivio 6310-NC (Target of Evaluation, or TOE) is a network device providing highly variable network functionality. It achieves this by leveraging RHEL 7.4 to provide full hardware access to the networking applications, allowing them to address the high performance hardware devices directly. All access to the networking applications is constrained to be via the management entity, described below.



1.4.1 Target of Evaluation Physical Boundaries

The TOE consists of the following components:

- The management entity is BiviOS 8.3.1 over RHEL 7.4 running on an Intel Xeon Gold 6148 processor, and is the only entity accessible remotely.
- The TOE also supports an "application Entity" which is powered by the same Intel Xeon Gold 6148 processor, running BiviOS 8.3.1 over RHEL 7.4. The only application used for the purposes of this evaluation is the standard, factory-installed application. Applications run natively, and the application entity does not employ any hypervisor.
- The application entity may not be accessed remotely, except via the management entity. Local access for maintenance/debugging purposes is provided via a serial console cable. Both the Management and Application entities directly access a shared set of processor, RAM, and storage; however, the application entity directly accesses a separate set of hardware network interfaces.
- The available application space configurations are described below:

TOE's are identified with a part number in the format

1. B6310-NC-C(1,2,3,5,6)M(1,2,3,4,5)D(1,2,3,4,5,6)N(1,2,3,4)
   a. This chassis is the "standard" product chassis.
2. B6310R-NC-C(5,6)M(1,2,3)D(1,2,3,4,5,6)N(1,2,4).
   a. This chassis is a shorter, ruggedized chassis

3. PacStar 451. This chassis does not have configuration options, and will always use the "C4" processor specification (defined below) and no others.

The naming conventions specified above reference the following hardware:

Part Number	Processor
Options with C1	Dual Intel Xeon Gold 6148, 2.4 GHz w/ 27Mb Cache
Options with C2	Dual Intel Xeon Platinum 8180, 2.5 GHz w/ 38Mb Cache
Options with C3	Dual Intel Xeon Silver 4110, 2.1 GHz w/ 11Mb Cache
Options with C4	Intel Xeon E3-1515Mv5, 2.8 GHz w/ 8Mb Cache
Options with C5	Dual Intel Xeon Gold 6138, 2.0 GHz w/27Mb Cache
Options with C6	Dual Intel Xeon Gold 6152, 2.1 GHz w/30Mb Cache
Part Number	**Installed RAM**
Options with M1	256GB DDR4-2666 memory
Options with M2	512GB DDR4-2666 memory
Options with M3	384GB DDR4-2666 memory
Options with M4	768GB DDR4-2666 memory
Options with M5	1536GB DDR4-2666 memory
Part Number	**Installed Storage**
Options with D1	2x 1TB SSD storage
Options with D2	2x 2TB SSD storage
Options with D3	4x 2TB SSD storage
Options with D4	8x 2TB SSD storage
Options with D5	4x 3.8TB SSD storage
Options with D6	8x 3.8TB SSD storage
Part Number	**Installed NIC Interfaces**
Options with N1	2x 10GbE Fiber interfaces and 4x 1GbE Copper interfaces
Options with N2	4x 10GbE Fiber interfaces and 4x 1GbE Copper interfaces
Options with N3	6x 10GbE Fiber interfaces and 2x 1GbE Copper interfaces
Options with N4	4x 10GbE Fiber interfaces and 2x 1GbE Copper interfaces

All " M", "D", and "N" options are configuration options which do not affect validation, but are part of the model number.

Running:

- BiviOS 8.3.1 (Build 201704241036)

The guidance documentation, [AGD], that is part of the TOE is listed in Section 9 "References" within Table 12: TOE Guidance Documentation

1.4.2 Target of Evaluation Logical Boundary

The logical boundary of the TOE includes those security functions implemented exclusively by the TOE. These security functions are summarized in Section 1.3.3 above and are further described in the following subsections. A more detailed description of the implementation of these security functions are provided in Section 7 "TOE Summary Specification".

1.4.2.1 Audit

- The TOE will audit all events and information defined in Table 7: Auditable Events.
- The TOE will also include the identity of the user that caused the event (if applicable), date and time of the event, type of event, and the outcome of the event.
- The TOE protects storage of audit information from unauthorized deletion.
- The TOE prevents unauthorized modifications to the stored audit records.
- The TOE can transmit audit data to an external IT entity using SSH protocol.

1.4.2.2 Cryptographic Operations

The TSF performs the following cryptographic operations:

For TLS:

- AES-128 in CBC mode for data ciphering, using SHA-1 hashing and RSA key exchange.
- AES-256 in GCM mode for data ciphering, using SHA-384 hashing and ECDHE key exchange.

For SSH:

- AES-128 or AES-256 in CBC mode, HMAC-SHA2-256 or HMAC-SHA2-512 hashing and DH key exchange.
- Public key authentication via SSH-RSA, using HMAC-SHA1 hashing.

The TSF zeroizes all plaintext secret and private cryptographic keys and CSPs once they are no longer required.

1.4.2.3 Identification and Authentication

- The TSF supports passwords consisting of alphanumeric and special characters. The TSF also allows administrators to set a minimum password length and support passwords with 15 characters or more.
- The TSF requires all administrative-users to authenticate before allowing the user to perform any actions other than:
  - Viewing the warning banner
  - Responding to ICMP echo requests
  - Responding to ARP requests with ARP replies
  - Responding to DNS requests

1.4.2.4 Security Management

- The TSF stores and protects the following data:
  - Syslog data, user account data, and local authentication data (such as administrator passwords).
  - Cryptographic keys including pre-shared keys, symmetric keys, and private keys.
- There are two classes of users on the TOE:
  - First, the Admin user. The Admin user has full control over the TOE and can create other users (for instance, multiple administrative users) and control their level of access to the TOE.
  - Second, any administrator-created non-administrative user accounts. This would be a highly unusual configuration, as in most cases there is no reason to create a non-administrator account for the TOE. The TOE does not offer any functionality that requires users to authenticate other than to perform administration of the TOE.

- Management of the TSF:
    - The administrator can perform manual updates, determine the behavior of or modify the behavior of the handling of audit data, modify the behavior of the TSF, enable or disable services offered by the TOE, determine the behavior of or modify the behavior of audit functionality when local audit storage is full, manage TSF data, modify or delete or generate or import cryptographic keys, configure the access banner, and configure the session inactivity timeout period.
    - The administrator may perform these functions locally or remotely using the trusted path provided by SSH and defined in FTP_TRP.1.

1.4.2.5 Protection of the TSF

- The TSF protects TSF data from disclosure when the data is transmitted between different parts of the TOE.
- The TSF prevents the reading of secret and private keys.
- The TOE provides reliable time stamps for itself.
- The TOE runs a suite of self-tests during the initial start-up (upon power on) to demonstrate the correction operation of the TSF.
- The TOE provides a means to verify firmware/software updates to the TOE using a published hash prior to installing those updates.

1.4.2.6 TOE Access

- The TOE, for local interactive sessions, will terminate the session after an Authorized Administrator-specified period of session inactivity.
- The TOE terminates a remote interactive session after an Authorized Administrator-configurable period of session inactivity.
- The TOE allows Administrator-initiated termination of the Administrator’s own interactive session.
- Before establishing an administrative user session, the TOE is capable of displaying an Authorized Administrator-specified advisory notice and consent warning message regarding unauthorized use of the TOE.

1.4.2.7 Trusted Path/Channels

- The TOE uses SSH to provide a trusted communication channel between itself and all authorized IT entities that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.
- The TOE permits the TSF, or the authorized IT entities to initiate communication via the trusted channel.
- The TOE uses SSH or TLS to provide a trusted communication path between itself and authorized administrative users that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.
- The TOE permits remote administrators to initiate communication via the trusted path.
- The TOE requires the use of the trusted path for initial administrator authentication and all remote administration actions.

1.5 Notation, formatting, and conventions

The notation, formatting, and conventions used in this Security Target are defined below; these styles and clarifying information conventions were developed to aid the reader.

Where necessary, the ST author has added application notes to provide the reader with additional details to aid understanding; they are italicized and usually appear following the element needing clarification. Those notes specific to the TOE are marked "TOE Application Note;" those taken from the collaborative Protection Profile for Network Devices are marked "Application Note".

The notation conventions that refer to iterations, assignments, selections, and refinements made in this Security Target are in reference to SARs and SFRs taken directly from CC Part 2 and Part 3 as well as any SFRs and SARs taken from a Protection Profile.

The notation used in those PP to indicate iterations, assignments, selections, and refinements of SARs and SFRs taken from CC Part 2 and Part 3 is not carried forward into this document. Additionally, obvious errors in the PP are corrected and noted as such.

The CC permits four component operations (assignment, iteration, refinement, and selection) to be performed on requirement components. These operations are defined in Common Criteria, Part 1; paragraph 6.4.1.3.2, "Permitted operations on components" as:

- Iteration: allows a component to be used more than once with varying operations;
- Assignment: allows the specification of parameters;
- Selection: allows the specification of one or more items from a list; and
- Refinement: allows the addition of details.

Iterations are indicated by a number in parenthesis following the requirement number, e.g., FIA_UAU.1.1(1); the iterated requirement titles are similarly indicated, e.g., FIA_UAU.1(1).

Assignments made by the ST author are identified with **bold text.**

Selections are identified with <u>underlined text</u>**.**

Refinements that add text use ***bold and italicized text*** to identified the added text***.*** Refinements that performs a deletion, identifies the deleted text with ~~***strikeout, bold, and italicized text***~~.

2. Conformance Claims

2.1 Common Criteria Conformance Claims

This Security Target is conformant to the Common Criteria Version 3.1r4, CC Part 2 extended [C2], and CC Part 3 conformant [C3].

2.2 Conformance to Protection Profiles

This Security Target claims exact compliance to the collaborative Protection Profile for Network Devices, Version 1.0, dated February 27, 2015 [S1]. This Protection Profile will be referred to as cPP or PP for convenience throughout this Security Target. This [ST] also conforms the following NIAP-issued technical decisions and network interpretation team technical decisions:

The TOE complies with the following [NDcPP] Technical Decisions:

- TD0291 – NIT technical decision for DH14 and FCS_CKM.1
- TD0281* – NIT Technical Decision for Testing both thresholds for SSH rekey
- TD0255 – NIT Technical Decision for TLS Server Tests - Issue 3: Verification of application of encryption
- TD0235 – NIT Technical Decision adding DH group 14 to the selection in FCS_CKM.2
- TD0228 – NIT Technical Decision for CA certificates - basicConstraints validation
- TD0226 – NIT Technical Decision for TLS Encryption Algorithms
- TD0201* – NIT Technical Decision for Use of intermediate CA certificates and certificate hierarchy depth
- TD0200 – NIT Technical Decision for Password authentication for SSH clients
- TD0199 – NIT Technical Decision for Elliptic Curves for Signatures
- TD0191† – NIT Technical Decision for Using secp521r1 for TLS communication
- TD0189 – NIT Technical Decision for SSH Server Encryption Algorithms
- TD0188* – NIT Technical Decision for Optional use of X.509 certificates for digital signatures
- TD0187 – NIT Technical Decision for Clarifying FIA_X509_EXT.1 test 1
- TD0185* – NIT Technical Decision for Channel for Secure Update.
- TD0184* – NIT Technical Decision for Mandatory use of X.509 certificates
- TD0183* – NIT Technical Decision for Use of the Supporting Document
- TD0182 – NIT Technical Decision for Handling of X.509 certificates related to ssh-rsa and remote comms.
- TD0181* – NIT Technical Decision for Self-testing of integrity of firmware and software.
- TD0170* – NIT Technical Decision for SNMPv3 Support
- TD0169 – NIT Technical Decision for Compliance to RFC5759 and RFC5280 for using CRLs
- TD0168* – NIT Technical Decision for Mandatory requirement for CSR generation
- TD0167 – NIT Technical Decision for Testing SSH 2^28 packets
- TD0164 – NIT Technical Decision for Negative testing for additional ciphers for SSH
- TD0156 – NIT Technical Decision for SSL/TLS Version Testing in the NDcPP v1.0 and FW cPP v1.0
- TD0155 – NIT Technical Decision for TLSS tests using ECDHE in the NDcPP v1.0.
- TD0154 – NIT Technical Decision for Versions of TOE Software in the NDcPP v1.0 and FW cPP v1.0
- TD0151 – NIT Technical Decision for FCS_TLSS_EXT Testing - Issue 1 in NDcPP v1.0.
- TD0150 – NIT Technical Decision for Removal of SSH re-key audit events in the NDcPP v1.0 and FW cPP v1.0

- TD0143 – NIT Technical Decision for Failure testing for TLS session establishment in NDcPP and FWcPP
- TD0130 – NIT Technical Decision for Requirements for Destruction of Cryptographic Keys
- TD0126 – NIT Technical Decision for TLS Mutual Authentication
- TD0117 – NIT Technical Decision for FIA_X509_EXT.1.1 Requirement in NDcPP
- TD0116 – NIT Technical Decision for a Typo in reference to RSASSA-PKCS1v1_5 in NDcPP and FWcPP
- TD0114* – NIT Technical Decision for Re-Use of FIPS test results in NDcPP and FWcPP
- TD0113* – NIT Technical Decision for testing and trusted updates in the NDcPP v1.0 and FW cPP v1.0
- TD0112 – NIT Technical Decision for TLS testing in the NDcPP v1.0 and FW cPP v1.0.
- TD0111* – NIT Technical Decision for third party libraries and FCS_CKM.1 in NDcPP and FWcPP
- TD0096* – NIT Technical Interpretation regarding Virtualization
- TD0095* – NIT Technical Interpretations regarding audit, random bit generation, and entropy in NDcPP
- TD0094 – NIT Technical Decision for validating a published hash in NDcPP
- TD0093† – NIT Technical Decision for FIA_X509_EXT.1.1 Requirement in NDcPP
- TD0090 – NIT Technical Decision for FMT_SMF.1.1 Requirement in NDcPP

* denotes informational TDs.

† denotes superseded TDs.

2.3 Conformance to Security Packages

This Security Target does not claim conformance to any security function requirements or security assurance requirements packages, neither as package-conformant or package-augmented.

2.4 Conformance Claims Rationale

To demonstrate that exact conformance is met, this rationale shows all threats are addressed, all OSP are satisfied, no additional assumptions are made, all objectives have been addressed, and all SFRs and SARs have been instantiated.

The following address the completeness of the threats, OSP, and objectives, limitations on the assumptions, and instantiation of the SFRs and SARs:

- Threats
    - o All threats defined in the cPP are carried forward to this ST;
    - o No additional threats have been defined in this ST.
- Organizational Security Policies
    - o All OSP defined in the cPP are carried forward to this ST;

o No additional OSPs have been defined in this ST.

- Assumptions
    - o All assumptions defined in the cPP are carried forward to this ST;

  - No additional assumptions for the operational environment have been defined in this ST.
- Objectives
  - All objectives defined in the cPP are carried forward to this ST.
- All SFRs and SARs defined in the cPP are carried forward to this Security Target.

Rationale presented in the body of this ST shows all assumptions on the operational environment have been upheld, all the OSP are enforced, all defined objectives have been met and these objectives counter the defined threats.

Additionally, all SFRs and SARs defined in the cPP have been properly instantiated in this Security Target; therefore, this ST shows exact compliance to the cPP.

3. Security Problem Definition

3.1 Threats

The following table defines the security threats for the TOE, characterized by a threat agent, an asset, and an adverse action of that threat agent on that asset. These threats are taken directly from the PP unchanged.

Table 1: Threats

Threat	Description
T.UNAUTHORIZED_ADMINISTRATOR_ACCESS	Threat agents may attempt to gain administrator access to the network device by nefarious means such as masquerading as an administrator to the device, masquerading as the device to an administrator, replaying an administrative session (in its entirety, or selected portions), or performing man-in-the-middle attacks, which would provide access to the administrative session, or sessions between network devices. Successfully gaining administrator access allows malicious actions that compromise the security functionality of the device and the network on which it resides.
T.WEAK_CRYPTOGRAPHY	Threat agents may exploit weak cryptographic algorithms or perform a cryptographic exhaust against the key space. Poorly chosen encryption algorithms, modes, and key sizes will allow attackers to compromise the algorithms, or brute force exhaust the key space and give them unauthorized access allowing them to read, manipulate and/or control the traffic with minimal effort.
T.UNTRUSTED_COMMUNICATION_CHANNELS	Threat agents may attempt to target network devices that do not use standardized secure tunneling protocols to protect the critical network traffic. Attackers may take advantage of poorly designed protocols or poor key management to successfully perform man-in-the-middle attacks, replay attacks, etc. Successful attacks will result in loss of confidentiality and integrity of the critical network traffic, and potentially could lead to a compromise of the network device itself.
T.WEAK_AUTHENTICATION_ENDPOINTS	Threat agents may take advantage of secure protocols that use weak methods to authenticate the endpoints – e.g., shared password that is guessable or transported as plaintext. The consequences are the same as a poorly designed protocol, the attacker could masquerade as the administrator or another device, and the attacker could insert themselves into the network stream and perform a man-in-the-middle attack. The result is the critical network traffic is exposed and there could be a loss of confidentiality and integrity, and potentially the network device itself could be compromised.
T.UPDATE_COMPROMISE	Threat agents may attempt to provide a compromised update of the software or firmware which undermines the security functionality of the device. Non-validated updates or updates validated using non-secure or weak cryptography leave the update firmware vulnerable to surreptitious alteration.
T.UNDETECTED_ACTIVITY	Threat agents may attempt to access, change, and/or modify the security functionality of the network device without administrator awareness. This could result in the attacker finding an avenue (e.g., misconfiguration, flaw in the product) to compromise the device and the administrator would have no knowledge that the device has been compromised.
T.SECURITY_FUNCTIONALITY_COMPROMISE	Threat agents may compromise credentials and device data enabling continued access to the network device and its critical data. The compromise of credentials include replacing existing credentials with an attacker's credentials, modifying

Table 1: Threats

Threat	Description
	existing credentials, or obtaining the administrator or device credentials for use by the attacker.
T.PASSWORD_CRACKING	Threat agents may be able to take advantage of weak administrative passwords to gain privileged access to the device. Having privileged access to the device provides the attacker unfettered access to the network traffic, and may allow them to take advantage of any trust relationships with other network devices.
T.SECURITY_FUNCTIONALITY_FAILURE	A component of the network device may fail during start-up or during operations causing a compromise or failure in the security functionality of the network device, leaving the device susceptible to attackers.

3.2 Organizational Security Policies

The following table defines the organizational security policies which are a set of rules, practices, and procedures imposed by an organization to address its security needs. These threats are taken directly from the PP unchanged.

Table 2: Organizational Security Policies

OSP	Description
P.ACCESS_BANNER	The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE.

3.3 Assumptions

This section describes the assumptions on the operational environment in which the TOE is intended to be used. It includes information about the physical, personnel, and connectivity aspects of the environment. The operational environment must be managed in accordance with the provided guidance documentation. The following table defines specific conditions that are assumed to exist in an environment where the TOE is deployed. These assumptions are taken directly from the PP unchanged.

Table 3: Assumptions

Assumption	Description
A.PHYSICAL_PROTECTION	The network device is assumed to be physically protected in its operational environment and not subject to physical attacks that compromise the security and/or interfere with the device's physical interconnections and correct operation. This protection is assumed to be sufficient to protect the device and the data it contains. As a result, the cPP will not include any requirements on physical tamper protection or other physical attack mitigations. The cPP will not expect the product to defend against physical access to the device that allows unauthorized entities to extract data, bypass other controls, or otherwise manipulate the device.
A.LIMITED_FUNCTIONALITY	The device is assumed to provide networking functionality as its core function and not provide functionality/services that could be deemed as general purpose computing. For example the device should not provide computing platform for general purpose Applications (unrelated to networking functionality).
A.NO_THRU_TRAFFIC_PROTECTION	A standard/generic network device does not provide any assurance regarding the protection of traffic that traverses it. The intent is for the network device to protect data that originates on or is destined to the

Table 3: Assumptions

Assumption	Description
	device itself, to include administrative data and audit data. Traffic that is traversing the network device, destined for another network entity, is not covered by the ND cPP. It is assumed that this protection will be covered by cPPs for particular types of network devices (e.g, firewall).
A.TRUSTED_ADMINISTRATOR	The Security Administrator(s) for the network device are assumed to be trusted and to act in the best interest of security for the organization. This includes being appropriately trained, following policy, and adhering to guidance documentation. Administrators are trusted to ensure passwords/credentials have sufficient strength and entropy and to lack malicious intent when administering the device. The network device is not expected to be capable of defending against a malicious administrator that actively works to bypass or compromise the security of the device.
A.REGULAR_UPDATES	The network device firmware and software is assumed to be updated by an administrator on a regular basis in response to the release of product updates due to known vulnerabilities.
A.ADMIN_CREDENTIALS_SECURE	The administrator’s credentials (private key) used to access the network device are protected by the platform on which they reside.

4. Security Objectives

4.1 Security Objectives for the Operational Environment

Table 4: Security Objectives for the Operational Environment

Objective	Description
OE.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is provided by the environment.
OE.NO_GENERAL_PURPOSE	There are no general-purpose computing capabilities (e.g., compilers or user applications) available on the TOE, other than those services necessary for the operation, administration and support of the TOE.
OE.NO_THRU_TRAFFIC_PROTECTION	The TOE does not provide any protection of traffic that traverses it. It is assumed that protection of this traffic will be covered by other security and assurance measures in the operational environment
OE.TRUSTED_ADMIN	TOE Administrators are trusted to follow and apply all guidance documentation in a trusted manner.
OE.UPDATES	The TOE firmware and software is updated by an administrator on a regular basis in response to the release of product updates due to known vulnerabilities.
OE.ADMIN_CREDENTIALS_SECURE	The administrator's credentials (private key) used to access the TOE must be protected on any other platform on which they reside.

5. Extended Components Definition

This section provides definition of the extended security functional and assurance requirements; the components that are CC Part 2 extended, and CC Part 3 conformant, i.e., NIAP interpreted requirements, and extended requirements.

5.1 Extended Security Functional Requirements Definitions

There are no extended Security Functional Requirements defined in this Security Target. All extended SFRs were taken from the cPP.

5.2 Extended Security Assurance Requirement Definitions

There are no extended Security Assurance Requirements defined in this Security Target. All extended SARs were taken from the cPP.

6. Security Requirements

This section describes the security functional and assurance requirements for the TOE; those that are CC Part 2 conformant, CC Part 2 extended, CC Part 3 conformant, and CC Part 3 extended.

6.1 Security Function Requirements

This section describes the functional requirements for the TOE. The security functional requirement components in this security target are CC Part 2 conformant or CC Part 2 extended as defined in Section 2, Conformance Claims. Operations that were performed in the cPP are not signified in this section. Operations performed by the ST are denoted according to the formatting conventions in Section 1.5.

Table 5: Security Functional Requirements

#	SFR	Description
1	FAU_GEN.1	Audit Data Generation
2	FAU_GEN.2	User Audit Association
3	FAU_STG.1	Protected Audit Trail Storage (optional)
4	FAU_STG_EXT.1	Protected Audit Event Storage
5	FAU_STG_EXT.3	Display Warning for Local Storage Space (optional)
6	FCS_CKM.1	Cryptographic Key Generation (Refined)
7	FCS_CKM.2	Cryptographic Key Establishment (Refined)
8	FCS_CKM.4	Cryptographic Key Destruction
9	FCS_COP.1(1)	Cryptographic Operation (AES Data Encryption/Decryption)
10	FCS_COP.1(2)	Cryptographic Operation (Signature Generation and Verification)
11	FCS_COP.1(3)	Cryptographic Operation (Hash Algorithm)
12	FCS_COP.1(4)	Cryptographic Operation (Keyed Hash Algorithm)
13	FCS_RBG_EXT.1	Random Bit Generation
14	FCS_SSHC_EXT.1	SSH Client Protocol (selection based)
15	FCS_SSHS_EXT.1	SSH Server Protocol (selection based)
16	FCS_TLSS_EXT.1	TLS Server Protocol (selection based)
17	FIA_PMG_EXT.1	Password Management
18	FIA_UIA_EXT.1	User Identification and Authentication
19	FIA_UAU_EXT.2	Password-based Authentication Mechanism
20	FIA_UAU.7	Protected Authentication Feedback
21	FIA_X509_EXT.1	X.509 Certificate Validation
22	FIA_X509_EXT.2	X.509 Certificate Authentication
23	FIA_X509_EXT.3	X.509 Certificate Requests
24	FMT_MOF.1/LocSpace	Management of Security Behaviour (optional)
25	FMT_MOF.1(1)/Trusted Update	Management of Security Functions Behaviour
26	FMT_MOF.1(2)/Audit	Management of Security Functions Behaviour (optional)

Table 5: Security Functional Requirements

#	SFR	Description
27	FMT_MOF.1(1)/AdminAct	Management of Security Functions Behaviour (optional)
28	FMT_MOF.1(2)/AdminAct	Management of Security Functions Behaviour (optional)
29	FMT_MTD.1	Management of TSF Data
30	FMT_MTD.1/AdminAct	Management of TSF Data (optional)
31	FMT_SMF.1	Specification of Management Functions
32	FMT_SMR.2	Restrictions on Security Roles
33	FPT_APW_EXT.1	Protection of Administrator Passwords
34	FTP_ITC.1	Inter-TSF Trusted Channel
35	FPT_SKP_EXT.1	Protection of TSF Data (for reading all symmetric keys)
36	FPT_STM.1	Reliable Time Stamps
37	FPT_TST_EXT.1	TSF Testing
38	FPT_TUD_EXT.1	Trusted Update
39	FTA_SSL_EXT.1	TSF-initiated Session Locking
40	FTA_SSL.3	TSF-initiated Termination
41	FTA_SSL.4	User-initiated Termination
42	FTA_TAB.1	Default TOE Access Banners
43	FTP_TRP.1	Trusted Path

6.1.1 Class FAU: Security Audit

6.1.1.1 FAU_GEN.1 Audit Data Generation

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shut-down of the audit functions;
b) All auditable events for the not specified level of audit; and
c) All administrative actions comprising:
    - Administrative login and logout (name of user account shall be logged if individual user accounts are required for administrators).
    - Security related configuration changes (in addition to the information that a change occurred it shall be logged what has been changed).
    - Generating/import of, changing, or deleting of cryptographic keys (in addition to the action itself a unique key name or key reference shall be logged).
    - Resetting passwords (name of related user account shall be logged).
    - Starting and stopping services (if applicable)
    - **All entered commands.**
d) Specifically defined auditable events listed in Table 7.

Application Note 1

If the list of 'administrative actions' appears to be incomplete, the assignment in the selection should be used to list additional administrative actions which are audited.

The ST author replaces the cross-reference to the table of audit events with an appropriate cross-reference for the ST. This must also include the relevant parts of Table 3 and Table 4 for optional and selection-based SFRs included in the ST.

Application Note 2

The ST author can include other auditable events directly in the table; they are not limited to the list presented.

The TSS should identify what information is logged to identify the relevant key for the administrative task of generating/import of, changing, or deleting of cryptographic keys.

With respect to FAU_GEN.1.1 the term 'services' refers to trusted path and trusted channel communications, on demand self-tests, trusted update and administrator sessions (that exist under the trusted path) (e.g. netconf).

FAU_GEN1.2

The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and
b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, information specified in column three of Table 7.

Application Note 3

The ST author replaces the cross-reference to the table of audit events with an appropriate cross-reference for the ST. This must also include the relevant parts of Table 3 and Table 4 for optional and selection-based SFRs included in the ST.

Table 6: Auditable Events

#	SFR	Auditable Events	Additional Audit Record Contents
1	FAU_GEN.1	None.	None.
2	FAU_GEN.2	None.	None.
3	FAU_STG.1 (optional)	None.	None.
4	FAU_STG_EXT.1	None.	None.
5	FAU_STG_EXT.3 (optional)	Warning about low storage space for audit events	None.
6	FCS_CKM.1	None.	None.
7	FCS_CKM.2	None.	None.
8	FCS_CKM.4	None.	None.
9	FCS_COP.1(1)	None.	None.
10	FCS_COP.1(2)	None.	None.
11	FCS_COP.1(3)	None.	None.
12	FCS_COP.1(4)	None.	None.
13	FCS_RBG_EXT.1	None.	None.

Table 6: Auditable Events

#	SFR	Auditable Events	Additional Audit Record Contents
14	FCS_SSHC_EXT.1 (selection based)	Failure to establish an SSH session	Reason for failure
15	FCS_SSHS_EXT.1 (selection based)	Failure to establish an SSH session	Reason for failure.[1]
16	FCS_TLSS_EXT.1 (selection based)	Failure to establish a TLS Session	Reason for failure
17	FIA_PMG_EXT.1	None.	None.
18	FIA_UIA_EXT.1	All use of the identification and authentication mechanism.	Provided user identity, origin of the attempt (e.g., IP address).
19	FIA_UAU_EXT.2	All use of the identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
20	FIA_UAU.7	None.	None.
21	FIA_X509_EXT.1	Unsuccessful attempt to validate a certificate.	Reason for Failure
22	FIA_X509_EXT.2	None.	None.
23	FIA_X509_EXT.3	None.	None.
24	FMT_MOF.1/LocSpace	Modification of the behaviour of the audit functionality when Local Audit Storage Space is full.	None.
25	FMT_MOF.1(1)/ TrustedUpdate	Any attempt to initiate a manual update.	None.
26	FMT_MOF.1.1(2)/ Audit	Modification of the behaviour of the handling of audit data.	None
27	FMT_MOF.1(1)/ AdminAct	Modification of the behavior of the TSF.	None.
28	FMT_MOF.1(2)/ AdminAct	Starting and stopping of services.	None.
29	FMT_MTD.1	All management activities of TSF data.	None.
30	FMT_MTD.1/ AdminAct	Modification, deletion, generation/import of cryptographic keys.	None.
31	FMT_SMF.1	None.	None.
33	FMT_SMR.2	None.	None.
35	FPT_APW_EXT.1	None.	None.
43	FTP_ITC.1	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions.	Identification of the initiator and target of failed trusted channels establishment attempt.
34	FPT_SKP_EXT.1	None.	None.

[1] Modification per TD0150

Table 6: Auditable Events			
#	SFR	Auditable Events	Additional Audit Record Contents
38	FPT_STM.1	Changes to time.	The old and new values for the time. Origin of the attempt to change time for success and failure (e.g., IP address).
36	FPT_TST_EXT.1	None.	None.
37	FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure)	No additional information.
39	FTA_SSL_EXT.1	Any attempts at unlocking of an interactive session.	None.
40	FTA_SSL.3	The termination of a remote session by the session locking mechanism.	None.
41	FTA_SSL.4	The termination of an interactive session.	None.
42	FTA_TAB.1	None.	None.
44	FTP_TRP.1	Initiation of the trusted path. Termination of the trusted path. Failure of the trusted path functions.	Identification of the claimed user identity.

Application Note 4

Additional audit events will apply to the TOE depending on the optional and selection-based requirements adopted from Appendix A and Appendix B. The ST author must therefore include the relevant additional events specified in the tables in Table 4 and Table 5. The audit event for FIA_X509_EXT.1 is based on the TOE not being able to complete the certificate validation by ensuring the following:

- *the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.*
- *Verification of the digital signature of the trusted hierarchical CA*
- *read/access the CRL or access the OCSP server.*

If any of these checks fails, then an audit event with the failure should be written to the audit log

Assurance Activity:

Guidance

The evaluator shall check the administrative guide and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP is described and that the description of the fields contains the information required in FAU_GEN1.2, and the additional information specified in the table of auditable events.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of cPP. 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 security relevant with respect to the cPP. 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.

Test

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. Logging of all activities related to trusted update should be tested in detail and with utmost diligence. 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.

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

6.1.1.2 FAU_GEN.2 User Identity Association

FAU_GEN.2.1

For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user that caused the event.

Assurance Activity

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

6.1.1.3 FAU_STG.1 Protected Audit Trail Storage (Optional)

FAU_STG.1.1

The TSF shall protect the stored audit records in the audit trail from unauthorised deletion.

FAU_STG.1.2

The TSF shall be able to prevent unauthorised modifications to the stored audit records in the audit trail.

Assurance Activity

TSS

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

Guidance

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

Test

The evaluator shall perform the following tests:

a) Test 1: The evaluator shall access the audit trail as an unauthorized administrator and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail.
b) Test 2: The evaluator shall access the audit trail as an authorized administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.

6.1.1.4 FAU_STG_EXT.1 Protected Audit Event Storage

FAU_STG_EXT.1.1

The TSF shall be able to transmit the generated audit data to an external IT entity using a trusted channel according FTP_ITC.1.

Application Note 5:

For selecting the option of transmission of generated audit data to an external IT entity the TOE relies on a non-TOE audit server for storage and review of audit records. The storage of these audit records and the ability to allow the administrator to review these audit records is provided by the operational environment in that case.

FAU_STG_EXT.1.2

The TSF shall be able to store generated audit data on the TOE itself.

FAU_STG_EXT.1.3

The TSF shall overwrite previous audit records according to the following rule: **Periodic audit log rotation feature** when the local storage space for audit data is full.

Application Note 6

The external log server might be used as alternative storage space in case the local storage space is full. The 'other action' could in this case be defined as 'send the new audit data to an external IT entity'.

Assurance Activity

TSS

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

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

If the TOE complies with FAU_STG_EXT.2 the evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2 are correct when performing the tests for FAU_STG_EXT.1.3.

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.

Guidance

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

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

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

Test

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

a) Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing.

The evaluator shall 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:

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

6.1.1.5 FAU_STG_EXT.3 Display Warning for Local Storage Space (Optional)

FAU_STG_EXT.3.1

The TSF shall generate a warning to inform the user before the local space to store audit data is used up and/or the TOE will lose audit data due to insufficient local space.

Application Note 42

This option should be chosen if the TOE generates as warning to inform the user before the local storage space for audit data is used up. This might be useful if auditable events are stored on local storage space *only. It has to be ensured that the warning message required by FAU_STG_EXT.1.3 can be communicated to the user. The communication should be done via the audit log itself because it cannot be guaranteed that an administrative session is active at the time the event occurs.*

Assurance Activity

TSS

The evaluator shall examine the TSS to ensure that it details how the user is warned before the local storage for audit data is full.

Guidance

The evaluator shall also ensure that the guidance documentation describes how the user is warned before the local storage for audit data is full and how this warning is displayed or stored (since there is no guarantee that an administrator session is running at the time the warning is issued, it is probably stored in the log files). The description in the guidance documentation shall correspond to the description in the TSS.

Test

The evaluator shall verify that a warning is issued by the TOE before the local storage space for audit data is full.

6.1.2 Class FCS: Cryptographic Support

6.1.2.1 FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1

The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm:

- RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3;
- ECC schemes using "NIST curves" P-256 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;
- FFC Schemes using Diffie-Hellman group 14 that meet the following: RFC 3526, Section 3

Application Note 7:[2]

The ST author selects all key generation schemes used for key establishment and device authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2.1 and selected cryptographic protocols must match the selection. When key generation is used for device authentication, the public key is expected to be associated with an X.509v3 certificate.

"If the TOE acts as a receiver in the key establishment schemes and is not configured to support mutual authentication, the TOE does not need to implement key generation."

For further information, please see the NIT interpretation at: https://www.niap-ccevs.org/Documents_and_Guidance/ccevs/NITDecisionRfI201703.pdf

[2] Modification per TD0227

Assurance Activity

TSS

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.

Guidance

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 uses defined in this PP.

Test

See Annex A for Algorithm Validation Assurance Activities. Testing for FFC Schemes using Diffie-Hellman group 14 is done as part of testing in CKM.2.1.

6.1.2.2 FCS_CKM.2 Cryptographic Key Establishment

FCS_CKM.2.1

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:

- <u>Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”;</u>
- <u>Key establishment scheme using Diffie-Hellman group 14 that meet the following: RFC 3526, Section 3</u>

Application Note 8

This is a refinement of the SFR FCS_CKM.2 to deal with key establishment rather than key distribution.

The ST author selects all key establishment schemes used for the selected cryptographic protocols. For Diffie-Hellman group 14, ST authors should make the corresponding selection from the SFR instead of using the Finite field-based key establishment selection.

The RSA-based key establishment schemes are described in Section 9 of NIST SP 800-56B Revision 1; however, Section 9 relies on implementation of other sections in SP 800-56B Revision 1.

The elliptic curves used for the key establishment scheme correlate with the curves specified in FCS_CKM.1.1.

The domain parameters used for the finite field-based key establishment scheme are specified by the key generation according to FCS_CKM.1.1.

Assurance Activity

TSS

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 (including whether the TOE acts as a sender, a recipient, or both). If Diffie-Hellman group 14 is selected from FCS_CKM.2.1, the TSS shall describe how the implementation meets RFC 3526 Section 3. **Guidance**

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

See Annex A for Algorithm Validation Assurance Activities.

6.1.2.3 FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM.4.1 [3]

The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method

• For plaintext keys in volatile storage, the destruction shall be executed by a single overwrite consisting of zeroes, destruction of reference to the key directly followed by a request for garbage collection;

• For plaintext keys in non-volatile storage, the destruction shall be executed by the invocation of an interface provided by a part of the TSF that

- o logically addresses the storage location of the key and performs a single overwrite consisting of zeroes;

that meets the following: *No Standard.*

Application Note

In parts of the selections where keys are identified as being destroyed by "a part of the TSF", the TSS identifies the relevant part and the interface involved. The interface referenced in the requirement could take different forms for different TOEs, the most likely of which is an application programming interface to an OS kernel. There may be various levels of abstraction visible. For instance, in a given implementation the application may have access to the file system details and may be able to logically address specific memory locations. In another implementation the application may simply have a handle to a resource and can only ask another part of the TSF such as the interpreter or OS to delete the resource.

Where different key destruction methods are used for different keys and/or different destruction situations then the different methods and the keys/situations they apply to are described in the TSS (and the ST may use separate iterations of the SFR to aid clarity). The TSS describes all relevant keys used in the implementation of SFRs, including cases where the keys are stored in a non-plaintext form. In the case of non-plaintext storage, the encryption method and relevant key-encrypting-key are identified in the TSS.

Some selections allow assignment of "a value that does not contain any CSP". This means that the TOE uses some specified data not drawn from an RBG meeting FCS_RBG_EXT requirements, and not being

[3] Modification per TD0130

any of the particular values listed as other selection options. The point of the phrase "does not contain any CSP" is to ensure that the overwritten data is carefully selected, and not taken from a general pool that might contain current or residual data that itself requires confidentiality protection.

Key destruction does not apply to the public component of asymmetric key pairs.

Assurance Activity

TSS

The evaluator shall check to ensure the TSS lists each type of plaintext key material and its origin and storage location.

The evaluator shall verify that the TSS describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, etc.).

The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

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

FCS_COP.1.1(1)

The TSF shall perform encryption/decryption in accordance with a specified cryptographic algorithm AES used in CBC, GCM mode and cryptographic key sizes 128 bits, 256 bits that meet the following: AES as specified in ISO 18033-3, CBC as specified in ISO 10116, GCM as specified in ISO 19772.

Application Note 9

For the first selection of FCS_COP.1.1(1), the ST author should choose the mode or modes in which AES operates. For the second selection, the ST author should choose the key sizes that are supported by this functionality. The modes and key sizes selected here correspond to the cipher suite selections made in the trusted channel requirements.

Assurance Activity

Test

See Annex A for Algorithm Validation Assurance Activities.

6.1.2.5 FCS_COP.1(2) Cryptographic Operation (Signature Generation and Verification)

FCS_COP.1.1(2) [4][5]

The TSF shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm:

- RSA Digital Signature Algorithm and cryptographic key sizes (modulus) **2048 bits** ,
- Elliptic Curve Digital Signature Algorithm and cryptographic key sizes **256 bits**

that meet the following:

- For RSA schemes: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3
- For ECDSA schemes: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 6 and Appendix D, Implementing “NIST curves” P-256; ISO/IEC 14888-3, Section 6.4

Application Note 10

The ST Author chooses the algorithm(s) implemented to perform digital signatures. For the algorithm(s) chosen, the ST author makes the appropriate assignments/selections to specify the parameters that are implemented for that algorithm. The ST author ensures that the assignments and selections for this SFR include all the parameter values necessary for the cipher suites selected for the protocol SFRs (see Appendix B.2.1) that are included in the ST. The ST Author checks for consistency of selections with other FCS requirements, especially when supporting elliptic curves.

Assurance Activity

Test

See Annex A for Algorithm Validation Assurance Activities.

6.1.2.6 FCS_COP.1(3) Cryptographic Operation (Hash Algorithm)

FCS_COP.1.1(3)

The TSF shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm SHA-1, SHA-256, SHA-384, SHA-512 that meet the following: ISO/IEC 10118-3:2004.

Application Note 11

Vendors are strongly encouraged to implement updated protocols that support the SHA-2 family; until updated protocols are supported, this PP allows support for SHA-1 implementations in compliance with SP 800-131A. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1) and FCS_COP.1(2) (for example, SHA 256 for 128-bit keys).

Assurance Activity

TSS

[4] Modification per TD0116

[5] Modification per TD0199

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.

Guidance

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

Test

See Annex A for Algorithm Validation Assurance Activities.

6.1.2.7 FCS_COP.1(4) Cryptographic Operation (Keyed Hash Algorithm)

FCS_COP.1.1(4)

The TSF shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm <u>HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512</u> and cryptographic key sizes **160, 256, 384, 512** and message digest sizes <u>160, 256, 384, 512</u> bits that meet the following: ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2".

Application Note 12

The key size [k] in the assignment falls into a range between L1 and L2 (defined in ISO/IEC 10118 for the appropriate hash function). For example, for SHA-256, L1=512, L2=256, where L2<=k<=L1.

Assurance Activity

TSS

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.

Test

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 IV using a known good implementation.

6.1.2.8 FCS_RBG_EXT.1 Random Bit Generation

FCS_RBG_EXT.1.1

The TSF shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using <u>CTR_DRBG (AES)</u>.

FCS_RBG_EXT.1.2

The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from <u>**one**</u> <u>hardware-based noise source</u> with a minimum of <u>256 bits</u> of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 "Security Strength Table for Hash Functions", of the keys and hashes that it will generate.

Application Note 13

For the first selection in FCS_RBG_EXT.1.2, the ST selects at least one of the types of noise sources. If the TOE contains multiple noise sources of the same type, the ST author fills the assignment with the appropriate number for each type of source (e.g., 2 software-based noise sources, 1 hardware-based noise *source). The documentation and tests required in the Evaluation Activity for this element necessarily cover each source indicated in the ST.*

ISO/IEC 18031:2011 contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used, and include the specific underlying cryptographic primitives used in the requirement. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.

If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1 may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2, the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

Assurance Activity

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

Test

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

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

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

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

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

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

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

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

6.1.2.9 FCS_SSHC_EXT.1 SSH Client Protocol (Selection-based)

FCS_SSHC_EXT.1.1

The TSF shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254, and, 6668, no other RFCs.

Application Note 63

The ST author selects which of the additional RFCs to which conformance is being claimed. Note that these need to be consistent with selections in later elements of this component (e.g., cryptographic algorithms permitted). RFC 4253 indicates that certain cryptographic algorithms are "REQUIRED". This means that the implementation must include support, not that the algorithms must be enabled for use. Ensuring that algorithms indicated as "REQUIRED" but not listed in the later elements of this component are implemented is out of scope of the assurance activity for this requirement.

Assurance Activity

TSS

The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHC_EXT.1.5, and ensure that password-based authentication methods are also allowed.

FCS_SSHC_EXT.1.2 [6]

The TSF shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, password-based

Assurance Activity

TSS

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.

Test[7]

Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms 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.

[6] Modification per TD0200

[7] Modification per TD0164

Test 2: This test is only applicable if password-based authentication has been selected in FCS_SSHC_EXT.1.2 in the ST. Otherwise this test shall be omitted. Using the guidance documentation, the evaluator shall configure the TOE to perform password-based authentication to an SSH server, and demonstrate that a user can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.

FCS_SSHC_EXT.1.3

The TSF shall ensure that, as described in RFC 4253, packets greater than **262144 (256*1024)** bytes in an SSH transport connection are dropped (the connection will be aborted by OpenSSH).

Application Note 64

RFC 4253 provides for the acceptance of "large packets" with the caveat that the packets should be of "reasonable length" or dropped. The assignment should be filled in by the ST author with the maximum packet size accepted, thus defining "reasonable length" for the TOE.

Assurance Activity

TSS

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

Test

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

FCS_SSHC_EXT.1.4 [8]

The TSF shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: aes128-cbc, aes256-cbc

Application Note 65

RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as encryption algorithms when the same algorithm is being used as the MAC algorithm. Corresponding FCS_COP entries are included in the ST for the algorithms selected here.

Assurance Activity

TSS

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.

Guidance

[8] Modification per TD0189

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).

Test

Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms 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.

Test 2: The evaluator shall configure an SSH server to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

FCS_SSHC_EXT.1.5

The TSF shall ensure that the SSH transport implementation uses ssh-rsa, and no other public key algorithms as its public key algorithm(s) and rejects all other public key algorithms.

Application Note 66

Implementations that select only ssh-rsa will not achieve the 112-bit security strength in the digital signature generation for SSH authentication as is recommended in NIST SP 800-131A. Future versions of this profile may remove ssh-rsa as a selection. If x509v3-ecdsa-sha2-nistp256 or 509v3-ecdsa-sha2-nistp384 are selected, then the list of trusted certification authorities must be selected in FCS_SSHC_EXT.1.9.

Assurance Activity

TSS

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.

Guidance

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).

Test

Test 1: The evaluator shall establish a 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 2: The evaluator shall configure an SSH server to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

FCS_SSHC_EXT.1.6

The TSF shall ensure that the SSH transport implementation uses hmac-sha2-256, hmac-sha2-512 and no other MAC algorithms as its data integrity MAC algorithm(s) and rejects all other MAC algorithm(s).

Application Note 67

RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as MAC algorithms when the same algorithm is being used as the encryption algorithm. RFC 6668 specifies the use of the sha2 algorithms in SSH.

Assurance Activity

TSS

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

Guidance

The evaluator shall also check the guidance documentation to ensure that it contains instructions to the 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).

Test

Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms 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.

Test 2: The evaluator shall configure an SSH server to only allow the “none” MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

Test 3: The evaluator shall configure an SSH server to only allow the hmac-md5 MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

FCS_SSHC_EXT.1.7

The TSF shall ensure that diffie-hellman-group14-sha1, and no other methods are the only allowed key exchange methods used for the SSH protocol.

Assurance Activity

TSS

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

Guidance

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

Test

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.

FCS_SSHC_EXT.1.8 [9]

The TSF shall ensure that within SSH connections the same session keys are used for a threshold of no longer than one hour, and no more than one gigabyte of transmitted data. After either of the thresholds are reached a rekey needs to be performed.

Application Note:

This SFR defines two thresholds - one for the maximum time span the same session keys can be used and the other one for the maximum amount of data that can be transmitted using the same session keys. Both thresholds need to be implemented and a rekey needs to be performed on whichever threshold is reached first. For the maximum transmitted data threshold the total incoming and outgoing data needs to be counted. The rekey applies to all session keys (encryption, integrity protection) for incoming and outgoing traffic. It is acceptable for a TOE to implement lower thresholds than the maximum values defined in the SFR.

For any configurable threshold related to this requirement the guidance documentation needs to specify how the threshold can be configured. The allowed values must either be specified in the guidance documentation and must be lower or equal to the thresholds specified in this SFR or the TOE must not accept values beyond the thresholds specified in this SFR.

Assurance Activity

TSS

The evaluator shall check the TSS to ensure that it describes how the SFR is met. This comprises checking that the TSS clarifies that both thresholds are checked by the TOE and that rekeying is performed upon reaching the threshold whichever is hit first.

Guidance

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.

Test

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

For testing of the time-based threshold the evaluator shall use 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 a rekey has been performed. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports

[9] Modification per TD0167

auditing of rekey events. The evaluator uses available methods and tools to demonstrate that rekeying occurs.

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.

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH server, and shall transmit data from and to the TOE within the active SSH session until the threshold for transmitted traffic is reached. The transmitted traffic is the total traffic comprising incoming and outgoing traffic.

The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and a rekey has been performed. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports auditing of rekey events. The evaluator uses available methods and tools to demonstrate that rekeying occurs.

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.

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(1)/AdminAct).

FCS_SSHC_EXT.1.9

The TSF shall ensure that the SSH client authenticates the identity of the SSH server using a local database associating each host name with its corresponding public key or <u>no other methods</u> as described in RFC 4251 section 4.1.

Application Note 68

The list of trusted certification authorities can only be selected if x509v3-ecdsa-sha2-nistp256 or x509v3-ecdsa-sha2-nistp384 are selected in FCS_SSHC_EXT.1.5.

Assurance Activity

Test

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 user to accept or deny the key before continuing the connection.

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. 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).

6.1.2.10 FCS_SSHS_EXT.1 SSH Server Protocol (Selection-based)

FCS_SSHS_EXT.1.1

The TSF shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254, and, 6668, no other RFCs.

Application Note 69

The ST author selects which of the additional RFCs to which conformance is being claimed. Note that these need to be consistent with selections in later elements of this component (e.g., cryptographic algorithms permitted). RFC 4253 indicates that certain cryptographic algorithms are "REQUIRED". This means that the implementation must include support, not that the algorithms must be enabled for use. Ensuring that algorithms indicated as "REQUIRED" but not listed in the later elements of this component are implemented is out of scope of the assurance activity for this requirement.

FCS_SSHS_EXT.1.2

The TSF shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, password-based.

Assurance Activity

TSS

The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHS_EXT.1.5, and ensure that password-based authentication methods are also allowed.

Test

Test 1: The evaluator shall, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.

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 3: Using the guidance documentation, the evaluator shall configure the TOE to accept password-based authentication, and demonstrate that a user can be successfully authenticated to the TOE over SSH using a password as an authenticator.

Test 4: The evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

FCS_SSHS_EXT.1.3

The TSF shall ensure that, as described in RFC 4253, packets greater than **262144 (256*1024)** bytes in an SSH transport connection are dropped (the connection will be aborted by OpenSSH).

Application Note 70

RFC 4253 provides for the acceptance of "large packets" with the caveat that the packets should be of "reasonable length" or dropped. The assignment should be filled in by the ST author with the maximum packet size accepted, thus defining "reasonable length" for the TOE.

Assurance Activity

TSS

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

Test

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

FCS_SSHS_EXT.1.4 [10]

The TSF shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: aes128-cbc, aes256-cbc

Application Note 71

RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as encryption algorithms when the same algorithm is being used as the MAC algorithm. Corresponding FCS_COP entries are included in the ST for the algorithms selected here.

Assurance Activity

TSS

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.

Guidance

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).

Test[11]

[10] Modification per TD0189

[11] Modification per TD0164

Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms 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.

Test 2: The evaluator shall configure an SSH client to only allow an encryption 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.

FCS_SSHS_EXT.1.5

The TSF shall ensure that the SSH transport implementation uses ssh-rsa, and no other public key algorithms as its public key algorithm(s) and rejects all other public key algorithms.

Application Note 72

Implementations that select only ssh-rsa will not achieve the 112-bit security strength in the digital signature generation for SSH authentication as is recommended in NIST SP 800-131A. Future versions of this profile may remove ssh-rsa as a selection.

Assurance Activity

TSS

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.

Guidance

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).

Test

Test 1: The evaluator shall establish a 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.

Test 2: The evaluator shall configure an SSH client to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

FCS_SSHS_EXT.1.6

The TSF shall ensure that the SSH transport implementation uses, hmac-sha2-256, hmac-sha2-512 and no other MAC algorithms as its data integrity MAC algorithm(s) and rejects all other MAC algorithm(s).

Application Note 73

RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as MAC algorithms when the same algorithm is being used as the encryption algorithm. RFC 6668 specifies the use of the sha2 algorithms in SSH.

Assurance Activity

TSS

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

Guidance

The evaluator shall also check the guidance documentation to ensure that it contains instructions to the 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).

Test

Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms 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.

Test 2: The evaluator shall configure an SSH client to only allow the "none" MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Test 3: The evaluator shall configure an SSH client to only allow the hmac-md5 MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

FCS_SSHS_EXT.1.7

The TSF shall ensure that diffie-hellman-group14-sha1 and no other methods are the only allowed key exchange methods used for the SSH protocol.

Assurance Activity

Guidance

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

Test

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

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

FCS_SSHC_EXT.1.8 [12]

The TSF shall ensure that within SSH connections the same session keys are used for a threshold of no longer than one hour, and no more than one gigabyte of transmitted data. After either of the thresholds are reached a rekey needs to be performed.

[12] Modification per TD0167

Application Note:

This SFR defines two thresholds - one for the maximum time span the same session keys can be used and the other one for the maximum amount of data that can be transmitted using the same session keys. Both thresholds need to be implemented and a rekey needs to be performed on whichever threshold is reached first. For the maximum transmitted data threshold the total incoming and outgoing data needs to be counted. The rekey applies to all session keys (encryption, integrity protection) for incoming and outgoing traffic. It is acceptable for a TOE to implement lower thresholds than the maximum values defined in the SFR.

For any configurable threshold related to this requirement the guidance documentation needs to specify how the threshold can be configured. The allowed values must either be specified in the guidance documentation and must be lower or equal to the thresholds specified in this SFR or the TOE must not accept values beyond the thresholds specified in this SFR.

Assurance Activity

TSS

The evaluator shall check the TSS to ensure that it describes how the SFR is met. This comprises checking that the TSS clarifies that both thresholds are checked by the TOE and that rekeying is performed upon reaching the threshold whichever is hit first.

Guidance

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.

Test

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

For testing of the time-based threshold the evaluator shall use 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 a rekey has been performed. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports auditing of rekey events. The evaluator uses available methods and tools to demonstrate that rekeying occurs.

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.

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH server, and shall transmit data from and to the TOE within the active SSH session until the threshold for transmitted traffic is reached. The transmitted traffic is the total traffic comprising incoming and outgoing traffic.

The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and a rekey has been performed. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports auditing of rekey events. The evaluator uses available methods and tools to demonstrate that rekeying occurs.

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.

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(1)/AdminAct).

Assurance Activity

Test

The evaluator shall configure the TOE to create a log entry when a rekey occurs. The evaluator shall connect to the TOE with an SSH client and cause 2^28 packets to be transmitted from the client to the TOE, and subsequently review the audit log to ensure that a rekey occurred.

6.1.2.11 FCS_TLSS_EXT.1 TLS Server Protocol (Selection-based)

FCS_TLSS_EXT.1.1 [13][14]

The TSF shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

Application Note 83

"The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the ciphersuites that are supported. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. Note that RFC 5246 makes TLS_RSA_WITH_AES_128_CBC_SHA a mandatory ciphersuite, but it is treated as optional for the purposes of conformance with this cPP (i.e. the selection of 'TLS 1.2 (RFC 5246)' will be accepted as conformant with this SFR even if TLS_RSA_WITH_AES_128_CBC_SHA is not one of the ciphersuites listed in the ST)." These requirements will be revisited as new TLS versions are standardized by the IETF.

[13] Modification per TD0151

[14] Modification per TD0226

In a future version of this cPP TLS v1.2will be required for all TOEs.

Assurance Activity

TSS

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

Guidance

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

Test

The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_TLSS_EXT.1.3 (or FCS_TLSS_EXT.2.3). For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.[15]

Note that this testing can be accomplished in conjunction with the other testing activities

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

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

The requestor is correct in the RSA case. If a ciphersuite requiring RSA key exchange has been selected, then the server must terminate the connection without an alert after receiving the client's ChangeCipherSpec message or Finished message. If a ciphersuite requiring Diffie-Hellman key agreement has been selected, then the server may send an alert or simply terminate the connection after receiving the client's ChangeCipherSpec message or Finished message. Since the server cannot distinguish the different cases the Test Case 3 shall be changed to:

Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the

[15] Modification per TD0155

TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE either sends an alert after receiving the client's ChangeCipherSpec message or Finished message; or terminates the connection after receiving the client's ChangeCipherSpec message or Finished message.[16]

Test 4: The evaluator shall perform the following modifications to the traffic:

a)
b) Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.
c) After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.
d) Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

FCS_TLSS_EXT.1.2 [17]

The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0, and TLS 1.1.

Application Note 84

All SSL versions and TLS v1.0 are denied. Any TLS versions not selected in FCS_TLSS_EXT.1.1 should be selected here.

Assurance Activity

TSS

The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

Guidance

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

Test

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

.**FCS_TLSS_EXT.1.3**[18]

The TSF shall generate key establishment parameters using RSA with key size 2048 bits generate EC Diffie-Hellman parameters over NIST curves secp256r1 and no other curves; generate Diffie-Hellman parameters of size 2048 bits.

Application Note 85

[16] Modification per TD0143

[17] Modification per TD0156

[18] Modification per TD0226, which supersedes TD0191

If the ST lists a DHE or ECDHE ciphersuite in FCS_TLSS_EXT.1.1, the ST must include the Diffie-Hellman or NIST curves selection in the requirement. FMT_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the TLS connection.

Assurance Activity

TSS

The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

Guidance

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

Test

The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

6.1.3 Class FIA: Identification and Authentication

6.1.3.1 FIA_PMG_EXT.1 Password Management

FIA_PMG_EXT.1.1

The TSF shall provide the following password management capabilities for administrative passwords:

a) Passwords shall be able to be composed of any combination of upper and lower case letters, numbers, and the following special characters: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)”, **all other printable characters**;
b) Minimum password length shall be settable by the Security Administrator, and shall support passwords of 15 characters or greater.

Application Note 14

The ST author selects the special characters that are supported by TOE; they may optionally list additional special characters supported using the assignment. "Administrative passwords“ refers to passwords used by administrators at the local console, over protocols that support passwords, such as SSH and HTTPS, or to grant configuration data that supports other SFRs in the Security Target.

Assurance Activity

Guidance

The evaluator shall examine the guidance documentation to determine that it provides guidance to security administrators on the composition of strong passwords, and that it provides instructions on setting the minimum password length.

Test

The evaluator shall perform the following tests.

a) Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to 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, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

6.1.3.2 FIA_UIA_EXT.1 User Identification and Authentication

FIA_UIA_EXT.1.1

The TSF shall allow the following actions prior to requiring the non-TOE entity to initiate the identification and authentication process:

- Display the warning banner in accordance with FTA_TAB.1;
- **Respond to ICMP Echo Request, respond to ARP requests with ARP replies, make DNS Requests, respond to TLS ClientHello messages with TLS ServerHello messages on TCP port 27777, respond to SSH login messages on TCP port 22**

FIA_UIA_EXT.1.2

The TSF shall require each administrative user to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that administrative user.

Application Note 15

This requirement applies to users (administrators and external IT entities) of services available from the TOE directly, and not services available by connecting through the TOE. While it should be the case that few or no services are available to external entities prior to identification and authentication, if there are some available (perhaps ICMP echo) these should be listed in the assignment statement; otherwise "no other actions" should be selected.

Authentication can be password-based through the local console or through a protocol that supports passwords (such as SSH), or be certificate based (such as SSH, TLS).

For communications with external IT entities (e.g., an audit server or NTP server, for instance), such connections must be performed in accordance with FTP_ITC.1, whose protocols perform identification and authentication. This means that such communications (e.g., establishing the IPsec connection to the authentication server) would not have to be specified in the assignment, since establishing the connection "counts" as initiating the identification and authentication process.

Assurance Activity

TSS

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".

Guidance

The evaluator shall examine the guidance documentation to determine that any necessary reparatory 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.

Test

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

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/ login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.
b) Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.
c) Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.

6.1.3.3 FIA_UAU_EXT.2 Password-based Authentication Mechanism

FIA_UAU_EXT.2.1

The TSF shall provide a local password-based authentication mechanism, none to perform administrative user authentication.

Application Note 16

The assignment should be used to identify any additional local authentication mechanisms supported. Local authentication mechanisms are defined as those that occur through the local console; remote administrative sessions (and their associated authentication mechanisms) are specified in FTP_TRP.1.

Assurance Activity

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.

6.1.3.4 FIA_UAU.7 Protected Authentication Feedback

FIA_UAU.7.1

The TSF shall provide only obscured feedback to the administrative user while the authentication is in progress at the local console.

Application Note 17

"Obscured feedback" implies the TSF does not produce a visible display of any authentication data entered by a user (such as the echoing of a password), although an obscured indication of progress may be provided (such as an asterisk for each character). It also implies that the TSF does not return any information during the authentication process to the user that may provide any indication of the authentication data.

Assurance Activity

Test

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.

6.1.3.5 FIA_X509_EXT.1 X.509 Certificate Validation (Selection Based)[19]

FIA_X509_EXT.1.1 [20][21][22]

The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using a Certificate Revocation List (CRL) as specified in RFC 5280 Section 6.3.The TSF shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.
    - OCSP Certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field.

Application Note 18

FIA_X509_EXT.1.1 lists the rules for validating certificates. The ST author selects whether revocation status is verified using OCSP or CRLs. The trusted channel/path protocols require that certificates are used; this use requires that the extendedKeyUsage rules are verified.

The validation is expected to end in a trusted root CA certificate in a root store managed by the platform.

The TSS shall describe when revocation checking is performed. It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

FIA_X509_EXT.1.2

[19] Modification per TD0182

[20] Modification per TD0117

[21] Modification per TD0169

[22] Modification per TD0187

The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

Application Note 19

This requirement applies to certificates that are used and processed by the TSF and restricts the certificates that may be added as trusted CA certificates.

Assurance Activity[23]

TSS

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

The TSS shall describe when revocation checking is performed. It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

The evaluator shall ensure the TSS describes when the check of validity of the certificates takes place. It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not sufficient to verify the status of a X.509 certificate only when it's loaded onto the device. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

The TSS shall describe when revocation checking is performed. It is expected that revocation checking is performed when a certificate is used when performing trusted updates. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device.

"The evaluator shall ensure the TSS describes when the check of validity of the certificates takes place. It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not sufficient to verify the status of a X.509 certificate only when it's loaded onto the device.

It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

Test

The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

The evaluator shall perform the following tests for FIA_X509_EXT.1.1:

[23] Modifications per TD0228

a) Test 1a: The evaluator shall load a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds.
b) Test 1b: The evaluator shall then delete one of the certificates in the chain (i.e. the root CA certificate or other intermediate certificate, but not the end-entity certificate), and show that the function fails.
   c) Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.
   d) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-– conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the TOE certificate and revocation of the TOE intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.
   e) Test 4: If OCSP is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have
   f) Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)
   g) Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)
   h) Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

The evaluator shall perform the following tests for FIA_X509_EXT.1.2. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

It is not necessary to verify the revocation status of X.509 certificates during power-up.

The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used when performing trusted updates .It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device.

The goal of the following tests is to verify that the TOE accepts only certificates that have been marked as CA certificates by using basicConstraints with the CA flag set to True (and implicitly that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

For each of the following tests the evaluator shall create a chain of at least four certificates: a self-signed root CA certificate, two intermediate CA certificates, and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).Test 2: The evaluator shall ensure that at least one of the CA certificates in the chain has a basicConstraints extension in which the CA flag is set to FALSE. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate with the CA flag set to FALSE to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

6.1.3.6 FIA_X509_EXT.2 X.509 Certificate Authentication (Selection Based)[24]

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and no additional uses.

Application Note 20

The ST author's selection matches the selection of FTP_ITC.1.1. Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1) and for integrity verification (FPT_TST_EXT.2).

FIA_X509_EXT.2.2

When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall not accept the certificate.

Application Note 21

Often a connection must be established to check the revocation status of a certificate -either to download a CRL or to perform a lookup using OCSP. The selection is used to describe the behavior in the event that such a connection cannot be established (for example, due to a network error). If the TOE has determined the certificate valid according to all other rules in FIA_X509_EXT.1, the behavior indicated in the selection determines the validity. The TOE must not accept the certificate if it fails any of the other validation rules in FIA_X509_EXT.1. If the administrator-configured option is selected by the ST Author, the ST Author also selects the corresponding function in FMT_SMF.1.

Assurance Activity

TSS

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

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

[24] Modification per TD0182

Test

The evaluator shall perform the following test for each trusted channel:

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

6.1.3.7 FIA_X509_EXT.3 X.509 Certificate Requests (Selection Based)[25]

FIA_X509_EXT.3.1

The TSF shall generate a Certificate Request Message as specified by RFC 2986 and be able to provide the following information in the request: public key and <u>Common Name, Organization, Organizational Unit, Country.</u>

Application Note 22

The public key is the public key portion of the public-private key pair generated by the TOE as specified in FCS_CKM.1(1).

FIA_X509_EXT.3.2

The TSF shall validate the chain of certificates from the Root CA upon receiving the CA Certificate Response.

Assurance Activity

TSS

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

Guidance

The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request Message. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the certificate request message.

Test

The evaluator shall perform the following tests:

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

[25] Modification per TD0182

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

6.1.4 Class FMT: Security Management

6.1.4.1 FMT_MOF.1(1)/TrustedUpdate Management of Security Functions Behaviour

FMT_MOF.1.1(1)/TrustedUpdate

The TSF shall restrict the ability to enable the functions to perform manual update to Security Administrators.

Application Note 23

FMT_MOF.1(1)/TrustedUpdate restricts the initiation of manual updates to Security Administrators.

Assurance Activity

Test

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). This test should fail.

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

6.1.4.2 FMT_MOF.1(2)/Audit Management of Security Functions Behaviour (Optional)

FMT_MOF.1.1(2)/Audit

The TSF shall restrict the ability to determine the behaviour of, modify the behaviour of the functions handling of audit data to Security Administrators.

Application Note 44

FMT_MOF.1(2)/Audit should only be chosen if the handling of audit data is configurable. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

Assurance Activity

Test

The evaluator shall try to modify all parameters for configuration of the handling of audit data 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). This test should fail. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

The evaluator shall try to modify all parameters for configuration of the handling of audit data with prior authentication as security administrator. The effects of the modifications should be confirmed. The term ‘handling of audit data’ refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

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

6.1.4.3 FMT_MOF.1(1)/AdminAct Management of Security Behaviour (Optional)

FMT_MOF.1.1(1)/AdminAct

The TSF shall restrict the ability to modify the behaviour of the functions TOE Security Functions to Security Administrators.

Application Note 45

FMT_MOF.1(1)/AdminAct should only be chosen if the behaviour of the TOE Security Functions is configurable.

Assurance Activity

Test

The evaluator shall try to perform at least one of the related actions 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). These attempts should fail.

The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. These attempts should succeed.

6.1.4.4 FMT_MOF.1(2)/AdminAct Management of Security Behaviour (Optional)

FMT_MOF.1.1(2)/AdminAct

The TSF shall restrict the ability to enable, disable the functions services to Security Administrators.

Application Note 46

FMT_MOF.1(2)/AdminAct should only be chosen if the Security Administrator has the ability to start and stop services.

Assurance Activity

Test

The evaluator shall try to perform at least one of the related actions 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). These attempts should fail.

The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. These attempts should succeed.

6.1.4.5 FMT_MTD.1 Management of TSF Data

FMT_MTD.1.1

The TSF shall restrict the ability to manage the TSF Data to Security Administrators.

Application Note 24

The word "manage" includes but is not limited to create, initialize, view, change default, modify, delete, clear, and append. This SFR includes also the resetting of user passwords by the Security Administrator.

Assurance Activity

TSS

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.

Guidance

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.

6.1.4.6 FMT_MTD.1/AdminAct Management of TSF Data (Optional)

FMT_MTD.1.1/AdminAct

The TSF shall restrict the ability to modify, delete, generate/import the cryptographic keys to Security Administrators.

Application Note 48

FMT_MTD.1.1/AdminAct should only be chosen if cryptographic keys can be modified, deleted or generated/imported by the Security Administrator.

Assurance Activity

Test

The evaluator shall try to perform at least one of the related actions 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). This test should fail.

The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This test should pass.

6.1.4.7 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 [26]

The TSF shall be capable of performing the following management functions:

- Ability to administer the TOE locally and remotely;
- Ability to configure the access banner;
- Ability to configure the session inactivity time before session termination or locking;
- Ability to update the TOE, and to verify the updates using <u>hash comparison</u> [27]prior to installing those updates;

[26] Modification per TD0167

[27] Modification per TD0090

- Ability to configure audit behaviour;
- Ability to configure the cryptographic functionality;
- Ability to configure thresholds for SSH rekeying

Application Note 25

The TOE must provide functionality for both local and remote administration, including the ability to configure the access banner for FTA_TAB.1 and the session inactivity time(s) for FTA_SSL.3 & FTA_SSL.4. The item "Ability to update the TOE, and to verify the updates using digital signature capability prior to installing those updates" includes the relevant management functions from FMT_MOF.1(1)/TrustedUpdate, FMT_MOF.1(2)/TrustedUpdate (if included in the ST), FIA_X509_EXT.2.2 and FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action). Similarly, the selection "Ability to configure audit behavior" includes the relevant management functions from FMT_MOF.1(1)/Audit, FMT_MOF.1(2)/Audit, FMT_MOF.1.1(1)/AdminAct, FMT_MOF.1.1(2)/AdminAct and FMT_MOF.1/LocSpace (for all of these SFRs that are included in the ST). If the TOE offers the ability for the administrator to configure the audit behaviour, configure the services available prior to identification or authentication, or if any of the cryptographic functionality on the TOE can be configured, then the ST author makes the appropriate choice or choices in the second selection, otherwise select "No other capabilities."

The selection 'Ability to configure thresholds for SSH rekeying' shall be included in the ST if the TOE supports configuration of the thresholds for the mechanisms used to fulfil FCS_SSHC_EXT.1.8 or FCS_SSHS_EXT.1.8 (such configuration then requires the inclusion of FMT_MOF.1(1)/AdminAct in the ST). If the TOE places limits on the values accepted for the thresholds then this is stated in the TSS.

Assurance Activity

The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_TAB.1, FTA_SSL.3, FTA_SSL.4, FMT_MOF.1(1)/TrustedUpdate, FMT_MOF.1(2)/TrustedUpdate (if included in the ST), IA_X509_EXT.2.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(1)/Audit, FMT_MOF.1(2)/Audit, FMT_MOF.1.1(1)/AdminAct, FMT_MOF.1.1(2)/AdminAct and FMT_MOF.1/LocSpace (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

6.1.4.8 FMT_SMR.2 Restrictions on Security Roles

FMT_SMR.2.1

The TSF shall maintain the roles:

- Security Administrator.

FMT_SMR.2.2

The TSF shall be able to associate users with roles.

FMT_SMR.2.3

The TSF shall ensure that the conditions:

- The Security Administrator role shall be able to administer the TOE locally;
- The Security Administrator role shall be able to administer the TOE remotely

are satisfied.

Application Note 26

FMT_SMR.2.3 requires that a Security Administrator be able to administer the TOE through the local console and through a remote mechanism (IPsec, SSH, TLS, HTTPS).

Assurance Activity

Guidance

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.

Test

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.

6.1.5 Class FPT: Protection of the TSF

6.1.5.1 FPT_APW_EXT.1 Protection of Administrator Passwords

FPT_APW_EXT.1.1

The TSF shall store passwords in non-plaintext form.

FPT_APW_EXT.1.2

The TSF shall prevent the reading of plaintext passwords.

Application Note 28

The intent of the requirement is that raw password authentication data are not stored in the clear, and that no user or administrator is able to read the plaintext password through "normal" interfaces. An all-powerful administrator of course could directly read memory to capture a password but is trusted not to do so.

Assurance Activity

TSS

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.

6.1.5.2 FPT_SKP_EXT.1 Protection of TSF Data (for reading of all symmetric keys)

FPT_SKP_EXT.1.1

The TSF shall prevent reading of all pre-shared keys, symmetric keys, and private keys.

Application Note 27

The intent of this requirement is for the device to protect keys, key material, and authentication credentials from unauthorized disclosure. This data should only be accessed for the purposes of their assigned security functionality, and there is no need for them to be displayed/accessed at any other time. This requirement does not prevent the device from providing indication that these exist, are in use, or are still valid. It does, however, restrict the reading of the values outright.

Assurance Activity

TSS

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.

6.1.5.3 FPT_TST_EXT.1 TSF Testing

FPT_TST_EXT.1.1(1)

The TSF shall run a suite of the following self-tests during initial start-up (on power on) to demonstrate the correct operation of the TSF: **memory, RDRAND, software integrity, cryptographic tests**.

FPT_TST_EXT.1.1(2)

The TSF shall run a suite of the following self-tests periodically during normal operation to demonstrate the correct operation of the TSF: **cryptographic tests.**

FPT_TST_EXT.1.1(3)

The TSF shall run a suite of the following self-tests at the request of the authorised user, to demonstrate the correct operation of the TSF: **software integrity and cryptographic self-tests.**

Application Note 29

It is expected that self-tests are carried out during initial start-up (on power on). Other options should only be used if the developer can justify why they are not carried out during initial start-up. It is expected that at least self-tests for verification of the integrity of the firmware and software as well as for the correct operation of cryptographic functions necessary to fulfil the SFRs will be performed. If not all self-test are performed during start-up multiple iterations of this SFR are used with the appropriate options selected. In future versions of this cPP the suite of self-tests will be required to contain at least mechanisms for measured boot including self-tests of the components which perform the measurement.

Application Note 30

If certificates are used by the self-test mechanism (e.g. for verification of signatures for integrity verification), certificates are validated in accordance with FIA_X509_EXT.1 and should be selected in FIA_X509_EXT.2.1. Additionally, FPT_TST_EXT.2 must be included in the ST.

Assurance Activity

TSS

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.

Guidance

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.

Test

Future versions of this cPP will mandate a clearly defined minimum set of self tests. But also for this version of the cPP it is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE
b) Verification of the correct operation of the cryptographic functions necessary to fulfill any of the SFRs.

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

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

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

6.1.5.4 FPT_TUD_EXT.1 Trusted Update[28]

FPT_TUD_EXT.1.1 [29]

The TSF shall provide Security Administrators the ability to query the currently executing version of the TOE firmware/software and <u>the most recently installed version of the TOE firmware/software</u>

Application Note 31

"If a trusted update can be installed on the TOE with a delayed activation the version of both the currently executing image and the installed but inactive image must be provided. In this case the option 'the most recently installed version of the TOE firmware/software' needs to be chosen from the selection in FPT_TUD_EXT.1.1 and the TSS needs to describe how and when the inactive version becomes active. If all trusted updates become active as part of the installation process, only the currently executing version needs to be provided. In this case the option 'no other TOE firmware/software version' shall be chosen from the selection in FPT_TUD_EXT.1.1."

[28] Modification per TD0094

[29] Modification per TD0154

FPT_TUD_EXT.1.2

The TSF shall provide Security Administrators the ability to manually initiate updates to TOE firmware/software and <u>no other update mechanism</u>.

Application Note 32

The selection in FPT_TUD_EXT.1.2 distinguishes the support of automatic checking for updates and support of automatic updates. The first option refers to a TOE that checks whether a new update is available, communicates this to the administrator (e.g. through a message during an administrator session, through log files) but requires some action by the administrator to actually perform the update. The second option refers to a TOE that checks for updates and automatically installs them upon availability.

The TSS explains what actions are involved in the TOE support when using the 'support automatic checking for updates' or 'support automatic updates' selections.
When published hash values (see FPT_TUD_EXT.1.3) are used to protect the trusted update mechanism, the TOE must not automatically download the update file(s) together with the hash value (either integrated in the update file(s) or separately) and automatically install the update without any active authorization by the Security Administrator, even when the calculated hash value matches the published hash value. When using published hash values to protect the trusted update mechanism, the option 'support of automatic updates' must not be used (automated checking for updates is permitted, though). The TOE may automatically download the update file(s) themselves but not to the hash value.

For the published hash approach, it is intended that a Security Administrator is always required to give active authorisation for installation of an update (as described in more detail under FPT_TUD_EXT.1.3) below. Due to this, the type of update mechanism is regarded as 'manually initiated update', even if the update file(s) may be downloaded automatically. A fully automated approach (without Security Administrator intervention) can only be used when 'digital signature mechanism' is selected in FPT_TUD_EXT.1.3 below.

FPT_TUD_EXT.1.3

The TSF shall provide means to authenticate firmware/software updates to the TOE using a <u>published hash</u> prior to installing those updates.

Application Note 33

The digital signature mechanism referenced in the selection of FPT_TUD_EXT.1.3 is one of the algorithms specified in FCS_COP.1(2). The published hash referenced in FPT_TUD_EXT.1.3 is generated by one of the functions specified in FCS_COP.1(3). The ST author should choose the mechanism implemented by the TOE; it is acceptable to implement both mechanisms.

When published hash values are used to secure the trusted update mechanism, an active authorization of the update process by the Security Administrator is always required. The secure transmission of an authentic hash value from the developer to the Security Administrator is one of the key factors to protect the trusted update mechanism when using published hashes and the guidance documentation needs to describe how this transfer has to be performed. For the verification of the trusted hash value by the Security Administrator different use cases are possible. The Security Administrator could obtain the published hash value as well as the update file(s) and perform the verification outside the TOE while the hashing of the update file(s) could be done by the TOE or by other means. Authentication as Security Administrator and initiation of the trusted update would in this case be regarded as 'active authorization' of the trusted update. Alternatively, the Administrator could provide the TOE with the published hash value together with the update file(s) and the hashing and hash comparison is performed by the TOE. In case *of successful hash verification the TOE can perform the update without any additional step by the Security Administrator. Authentication as Security Administrator and sending the hash value to the TOE is regarded as 'active authorization' of the trusted update (in case of successful hash verification), because the administrator is expected to load the hash value only to the TOE when intending to perform the update. As long as the transfer of the hash value to the TOE is performed by the Security Administrator, loading of the update file(s) can be performed by the Security Administrator or can be automatically downloaded by the TOE from a repository.*

Application Note 34

Future versions of this cPP will mandate the use of a digital signature mechanism for trusted updates.

Application Note 35

If certificates are used by the update verification mechanism, certificates are validated in accordance with FIA_X509_EXT.1 and should be selected in FIA_X509_EXT.2.1. Additionally, FPT_TUD_EXT.2 must be included in the ST.

Application Note 36

“Update” in the context of this SFR refers to the process of replacing a non-volatile, system resident software component with another. The former is referred to as the NV image, and the latter is the update image. While the update image is typically newer than the NV image, this is not a requirement. There are legitimate cases where the system owner may want to rollback a component to an older version (e.g. when the component manufacturer releases a faulty update, or when the system relies on an undocumented feature no longer present in the update). Likewise, the owner may want to update with the same version as the NV image to recover from faulty storage.

All discrete software components (e.g. applications, drivers, kernel, firmware) of the TSF, should be digitally signed by the corresponding manufacturer and subsequently verified by the mechanism performing the update. Since it is recognized that components may be signed by different manufacturers, it is essential that the update process verify that both the update and NV images were produced by the same manufacturer (e.g. by comparing public keys) or signed by legitimate signing keys (e.g. successful verification of certificates when using X.509 certificates).

Assurance Activity

TSS

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

If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or guidance documentation) describes how the certificates are installed/updated/selected, if necessary.

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."

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

Guidance

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

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

Test

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 as well as the most recently installed version (should be the same version before updating). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.
b) Test 2 (if digital signatures are used): 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 gap between the installation of an update and a required reboot or activation to execute the updated code, 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.

Test 2 (if published hash is verified on the TOE): 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 user 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 storing the published hash value on 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 a gap between the installation of an update and a required reboot or activation to execute the updated code, 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.

The evaluator shall perform the Tests 1 and 2 for all methods supported (manual updates, automatic checking for updates, automatic updates).

If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 2 shall be skipped.

6.1.5.5 FPT_STM.1 Reliable Time Stamps

FPT_STM.1.1

The TSF shall be able to provide reliable time stamps.

Application Note 37

The TSF does not provide reliable information about the current time at the TOE's location by itself, but depends on external time and date information, either provided manually by the administrator or through the use of an NTP server. The term 'reliable time stamps' refers to the strict use of the time and date information, that is provided externally, and the logging of all changes to the time settings including information about the old and new time. With this information the real time for all audit data can be calculated.

Assurance Activity

TSS

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

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.

Test

The evaluator shall perform the following tests:

a) Test 1: The evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

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

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

6.1.6 Class FTA: TOE Access

6.1.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

FTA_SSL_EXT.1.1

The TSF shall, for local interactive sessions,

- terminate the session

after a Security Administrator-specified time period of inactivity.

Assurance Activity

Test

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.

6.1.6.2 FTA_SSL.3 TSF-initiated Termination

FTA_SSL.3.1

The TSF shall terminate a remote interactive session after a Security Administrator-configurable time interval of session inactivity.

Assurance Activity

Test

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.

6.1.6.3 FTA_SSL.4 User-initiated Termination

FTA_SSL.4.1

The TSF shall allow Administrator-initiated termination of the Administrator’s own interactive session.

Assurance Activity

Test

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 as been terminated.

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

6.1.6.4 FTA_TAB.1 Default TOE Access Banners

FTA_TAB.1.1

Before establishing an administrative user session the TSF shall display a Security Administrator-specified advisory notice and consent warning message regarding use of the TOE.

Application Note 38

This requirement is intended to apply to interactive sessions between a human user and a TOE. IT entities establishing connections or programmatic connections (e.g., remote procedure calls over a network) are not required to be covered by this requirement.

Assurance Activity

TSS

The evaluator shall check the TSS to ensure that it details each method of access (local and remote) available to the administrator (e.g., serial port, SSH, HTTPS).

Test

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.

6.1.7 Class FTP: Trusted Path/Channels

6.1.7.1 FTP_ITC.1 Inter-TSF Trusted Channel

FTP_ITC.1.1

The TSF shall be capable of using SSH to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: audit server, **no other capabilities** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

FTP_ITC.1.2

The TSF shall permit the TSF, or the authorized IT entities to initiate communication via the trusted channel.

FTP_ITC.1.3

The TSF shall initiate communication via the trusted channel for **audit server**.

Application Note 39

The intent of the above requirement is to provide a means by which a cryptographic protocol may be used to protect external communications with authorized IT entities that the TOE interacts with to perform its functions. The TOE uses at least one of the listed protocols for communications with the server that collects the audit information. If it communicates with an authentication server (e.g., RADIUS), then the ST author *chooses "authentication server" in FTP_ITC.1.1 and this connection must be capable of being protected by one of the listed protocols. If other authorized IT entities (e.g., NTP server) are protected, the ST author makes the appropriate assignments (for those entities) and selections (for the protocols that are used to protect those connections). The ST author selects the mechanism or mechanisms supported by the TOE, and then ensures that the detailed protocol requirements in Appendix B corresponding to their selection are included in the ST. If TLS is selected, the ST author will claim FCS_TLSC_EXT.2 instead of FCS_TLSC_EXT.1.*

While there are no requirements on the party initiating the communication, the ST author lists in the assignment for FTP_ITC.1.3 the services for which the TOE can initiate the communication with the authorized IT entity.

The requirement implies that not only are communications protected when they are initially established, but also on resumption after an outage. It may be the case that some part of the TOE setup involves manually setting up tunnels to protect other communication, and if after an outage the TOE attempts to re-establish the communication automatically with (the necessary) manual intervention, there may be a window created where an attacker might be able to gain critical information or compromise a connection.

Assurance Activity

TSS

The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each communications mechanism is identified in terms of the allowed protocols for that IT entity. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

Guidance

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.

Test

The evaluator shall perform the following tests:

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

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

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

d) Test 4: The evaluators shall, for each protocol associated with each authorized IT entity tested during test 1, the connection is physically interrupted. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

Further assurance activities are associated with the specific protocols.

6.1.7.2 FTP_TRP.1 Trusted Path

FTP_TRP.1.1

The TSF shall be capable of using SSH, TLS to provide a communication path between itself and authorized remote administrators that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from disclosure and provides detection of modification of the channel data.

FTP_TRP.1.2

The TSF shall permit remote administrators to initiate communication via the trusted path.

FTP_TRP.1.3

The TSF shall require the use of the trusted path for initial administrator authentication and all remote administration actions.

Application Note 40

This requirement ensures that authorized remote administrators initiate all communication with the TOE via a trusted path, and that all communication with the TOE by remote administrators is performed over this path. The data passed in this trusted communication channel are encrypted as defined by the protocol chosen in the first selection. The ST author selects the mechanism or mechanisms supported by the TOE, and then ensures that the detailed protocol requirements in Appendix B corresponding to their selection are included in the ST.

Assurance Activity

TSS

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.

Guidance

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

Test

The evaluator shall perform the following tests:

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

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

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

d) Test 4: The evaluators shall ensure that, for each protocol associated with each authorized IT entity tested during test 1, the connection is physically interrupted. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

Further assurance activities are associated with the specific protocols.

6.2 Security Assurance Requirements

This Security Target is conformant with the assurance requirements specified in the cPP.

Table 7: Assurance Requirements

Assurance Class	Assurance Component
Security Target (ASE)	Conformance claims (ASE_CCL.1)
	Extended components definition (ASE_ECD.1)
	ST introduction (ASE_INT.1)
	Security objectives for the operational environment (ASE_OBJ.1)
	Stated security requirements (ASE_REQ.1)
	Security Problem Definition (ASE_SPD.1)
	TOE summary specification (ASE_TSS.1)
Development (ADV)	Basic functional specification (ADV_FSP.1)
Guidance documents (AGD)	Operational user guidance (AGD_OPE.1)
	Preparative procedures (AGD_PRE.1)
Life cycle support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM coverage (ALC_CMS.1)
Tests (ATE)	Independent testing –sample (ATE_IND.1)
Vulnerability assessment (AVA)	Vulnerability survey (AVA_VAN.1)

6.2.1 Extended Security Assurance Requirements

These requirements are taken directly from the cPP.

6.2.1.1 ASE: Security Target

The ST is evaluated as per ASE activities defined in the CEM. In addition, there may be Evaluation Activities specified within the SD that call for necessary descriptions to be included in the TSS that are specific to the TOE technology type.

Appendix D provides a description of the information expected to be provided regarding the quality of entropy in the random bit generator.

The TOE summary specification shall describe how the TOE meets each SFR. In the case of entropy analysis the TSS is used in conjunction with required supplementary information on Entropy.

The requirements for exact conformance of the Security Target are described in section 2 and in [SD, 3.1].

6.2.1.1.1 Conformance Claims (ASE_CCL.1)

The table below indicates the actions to be taken for particular ASE_CCL.1 elements in order to determine exact conformance with a cPP.

Table 8: Conformance Claims

ASE_CCL.1 Element	Evaluator Action
ASE_CCL.1.8C	The evaluator shall check that the statements of security problem definition in the PP and ST are identical.

ASE_CCL.1.9C	The evaluator shall check that the statements of security objectives in the PP and ST are identical.
ASE_CCL.1.10C	The evaluator shall check that the statements of security requirements in the ST include all the mandatory SFRs in the cPP, and all of the selection-based SFRs that are entailed by selections made in other SFRs (including any SFR iterations added in the ST). The evaluator shall check that if any other SFRs are present in the ST (apart from iterations of SFRs in the cPP) then these are taken only from the list of optional SFRs specified in the cPP (the cPP will not necessarily include optional SFRs, but may do so). If optional SFRs from the cPP are included in the ST then the evaluator shall check that any selection-based SFRs entailed by the optional SFRs adopted are also included in the ST.

6.2.1.2 ADV: Development

The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any required supplementary information required by this cPP that is not to be made public.

6.2.1.2.1 Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TOE Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this cPP will necessarily have interfaces to the Operational Environment that are not directly invokable by TOE users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional "functional specification" documentation is necessary to satisfy the Evaluation Activities specified in the SD.

The Evaluation Activities in the SD are associated with the applicable SFRs; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

Evaluation Activities

The evaluator shall check the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g., audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent, is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any supplementary information required by the cPP for aspects such as entropy analysis or cryptographic key management architecture[30]: no additional "functional specification" documentation is necessary to satisfy the Evaluation Activities. The interfaces that need to be evaluated are also identified by reference to the assurance activities listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any supplementary information required by the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the Evaluation Activities for each SFR also means that the tracing required in ADV_FSP.1.2D is treated as implicit, and no separate mapping information is required for this element.

However, if the evaluator is unable to perform some other required Evaluation Activity because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

6.2.1.3 AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel.

Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes:

- instructions to successfully install the TSF in that environment; and
- instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
- instructions to provide a protected administrative capability.

Guidance pertaining to particular security functionality must also be provided; requirements on such guidance are contained in the Evaluation Activities specified in the SD.

6.2.1.3.1 Operational User Guidance (AGD_OPE.1)

The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages.

The developer should review the Evaluation Activities contained in the SD to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Evaluation Activities

The evaluator shall check the requirements below are met by the guidance documentation.

Guidance documentation shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

[30] The Security Target and AGD documentation are public documents. Supplementary information may be public or proprietary: the cPP and/or Evaluation Activity descriptions will identify where such supplementary documentation is permitted to be proprietary and non-public.

Guidance documentation must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

The contents of the guidance documentation will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.

c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

6.2.1.3.2 Preparative Procedures (AGD_PRE.1)

As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

Evaluation Activities

The evaluator shall check the requirements below are met by the preparative procedures.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

Preparative procedures shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

Preparative procedures must include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target). The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

Preparative procedures must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

The preparative procedures must include

a) instructions to successfully install the TSF in each Operational Environment; and

b) instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and

c) instructions to provide a protected administrative capability.

6.2.1.4 Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this cPP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it is a reflection on the information to be made available for evaluation at this assurance level.

6.2.1.4.1 Labelling of the TOE (ALC_CMC.1)

This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a "hard label" (e.g., stamped into the metal, paper label) or a "soft label" (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

6.2.1.4.2 TOE CM Coverage (ALC_CMS.1)

Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

6.2.1.5 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. For this cPP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

6.2.1.5.1 Independent Testing – Conformance (ATE_IND.1)

Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes "evaluated configuration" instructions). The focus of the testing is to confirm that the requirements specified in Section 5 are being met. The Evaluation Activities in the SD identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

Evaluation Activities

The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

The evaluator shall prepare a test plan that covers all of the testing actions for ATE_IND.1 in the CEM and in the SFR-related Evaluation Activities. While it is not necessary to have one test case per test listed in an Evaluation Activity, the evaluator must show in the test plan that each applicable testing requirement in the SFR-related Evaluation Activities is covered.

The test plan identifies the platforms to be tested, and for any platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition and configuration of each platform to be tested, and any setup actions that are necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of any cryptographic engine to be used (e.g. for cryptographic protocols being evaluated).

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives, and the expected results.

The test report (which could just be an updated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure, so that a fix was then installed and then a successful re-run of the test was carried out, then the report would show a “fail” result followed by a“pass” result (and the supporting details), and not just the “pass” result[31].

6.2.1.6 Class AVA: Vulnerability Assessment

For the first generation of this cPP, the iTC is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products and provide that content into the AVA_VAN discussion. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. This information will be used in the development of future protection profiles.

6.2.1.6.1 Vulnerability Survey (AVA_VAN.1)

Appendix A in [SD] provides a guide to the evaluator in performing a vulnerability analysis.

Evaluation Activities

The evaluator shall document their analysis and testing of potential vulnerabilities with respect to this requirement. This report could be included as part of the test report for ATE_IND, or could be a separate document.

[31] It is not necessary to capture failures that were due to errors on the part of the tester or test environment. The intention here is to make absolutely clear when a planned test resulted in a change being required to the originally specified test configuration in the test plan, to the evaluated configuration identified in the ST and guidance documentation, or to the TOE itself.

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.5. The evaluator shall then perform vulnerability analysis in accordance with Appendix A.4. The results of the analysis shall be documented in the report according to Appendix A.5.

7. TOE Summary Specification

This section provides evaluators and potential consumers of the TOE with a high-level description of each SFR, thereby enabling them to gain a general understanding of how the TOE is implemented. These descriptions are intentionally not overly detailed, thereby disclosing no proprietary information. These sections refer to SFRs defined in Section 6, Security Requirements.

The TOE consists of the following Security Functions:

- Security Audit
- Cryptographic Support
- Identification and Authentication
- Security Management
- Protection of the TSF
- TOE Access
- Trusted Path/Channels

7.1 Security Audit

7.1.1 Audit Generation

BiviOS creates and stores audit records for the following events:

Start up and shutdown of the audit function (as startup and shutdown messages for the OS, since logging may not be started or stopped independently of BiviOS startup and shutdown).

All administrative actions, including:

- Administrative login and logout, including username
- Security related configuration changes (in addition to the information that a change occurred it shall be logged what has been changed)
- Generating/import of, changing, or deleting of cryptographic keys (in addition to the action itself a unique key name or key reference shall be logged)
- Resetting passwords (name of related user account shall be logged)
- Starting and stopping services
- All commands entered during administrative sessions

All events specified in Table 7: Auditable Events.

The detail of audit record contents is specified in Table 7: Auditable Events.

Auditing is done using syslog. Syslog is configured to store the audit logs locally, and transmit them to one or more configured syslog audit servers via SSH. Local audit logs are stored in /var/log/ in the underlying file system. Only an authorized administrator can read log files, modify or delete log files, or archive log files through the CLI, and such actions require being first authenticated as an authorized administrator using the trusted path provided in FTP_TRP.1. The TOE only defines one user type, “administrator”.

For each audit event, the TSF associates the audit event with the identity of the user that caused the event. Each audit log contains a date-and-time stamp of the event, the type of event, subject identity, and outcome (success or failure) of the event. Additional information, if available per column 4 of Table 7, is also logged.

FAU_GEN.1 & FAU_GEN.2

7.1.2 Audit Storage

The TOE stores audit records locally as well as transmits them to one or more administrator-configurable syslog servers. Such audit records are unable to be viewed, modified, or deleted except by authorized administrators.

FAU_STG.1

The TOE transmits audit data to external audit servers via SSH, provided by FTP_ITC.1.

How the TOE stores data is configurable by the administrator. The whole partition on the management entity's onboard storage (effectively about 30GB) is available for storing audit logs. However, the configuration should prevent the partition being filled. The administrator can set up the configuration by editing the file /etc/audit/auditd.conf with particular attention to the following parameters:

The parameter "num_logs" defines the number of files to keep, which defaults to 5 files.

The parameter "max_log_file" sets the maximum size in megabytes an audit log file is allowed to reach, and then it is rotated by adding a suffix in the range [0 – 99].

The parameter "max_log_file_action" determines what the system will do when the maximum size is reached. As asserted in [T1], the TOE will be configured to use the option "rotate" to delete the oldest syslog log file to create space for a new file. This parameter must not be changed.

All stored audit logs and archived files, are protected against unauthorized deletion, modification, or viewing. To view these files, a user must be authenticated as an authorized administrator via the trusted path provide by SSH (FTP_TRP.1).

When the entire allocated space for logs is full, the oldest archived file will be deleted to make space for new audit logs. This behavior should not be modified by the administrator.

FAU_STG_EXT.1

In the /etc/audit/auditd.conf file, the parameter "space_left_action" determines the action taken when the system is running low on disk space. When this happens, the TOE will place a notice in "syslog". This behavior must not be modified by the authorized administrator.

FAU_STG_EXT.3

7.2 Cryptographic Support

The TOE utilizes the following CAVP-approved algorithms in the evaluated configuration:

Function	Certificate Number	SFR	Comments
AES - Encryption / Decryption	5349	FCS_COP.1(1)	The TOE performs AES in the CBC or GCM mode with key sizes 128 or 256 bits.
ECDH	CVL 1813, 1814	FCS_CKM.2	ECDH is used in TLS EC session key establishment.

ECDSA	1408	FCS_COP.1(2) FCS_CKM.1	The TOE performs ECDSA with NIST P-256 curves.
RSA	2863	FCS_COP.1(2) FCS_CKM.1	The TOE performs RSA with 2048-bit moduli.
HMAC	3547	FCS_COP.1(4)	The TOE performs HMAC using the following: HMAC-SHA1, with a key size of 160 bits and an output digest size of 160 bits. HMAC-SHA2-256 with a key size of 256 bits and an output digest size of 256 bits. HMAC-SHA2-384 with a key size of 384 bits and an output digest size of 384 bits. HMAC-SHA2-512 with a key size of 512 bits and an output digest size of 512 bits.
Cryptographic Hashing	4301	FCS_COP.1(3)	The TOE performs cryptographic hashing as follows: SHA1 - used in SSH, TLS HMAC. SHA-256 - used in SSH HMAC and TLS HMAC. SHA-384 - used in TLS and SSH HMAC. SHA-512 – used in TLS and SSH HMAC Appropriate hash algorithms are also used for digital signature verification and the key-derivation

			functions of SSH and TLS.
DRBG	2069	FCS_RBG_EXT.1	Random bit generation

The TOE uses OpenSSH 6.6.1p1 to provide all SSH functions, including the trusted path and trusted channel.

7.2.1 Cryptographic Key Generation

To support SSH for trusted path and trusted channel, the TOE cryptographic module implements RSA key generation with key sizes of 2048-bits and finite-field cryptography with modulus sizes of 2048 bits (Diffie-Hellman Group14). To meet the To support TLS, the TOE cryptographic module implements Elliptic-Curve key generation over NIST curve secp256r1 and RSA key generation using 2048-bit keys.

FCS_CKM.1

The TOE implements Elliptic Curve-based key establishment. Diffie-Hellman key establishment with DSA utilizing key sizes of 2048 bits is used in the SSH implementation, while elliptic curve-based key establishment with ECDH is used to establish the TLS tunnel. For Diffie-Hellman, the TOE meets RFC 3526 Section 3 by copying the parameters from the RFC and using them directly in prime generation.

FCS_CKM.2

Plaintext keys are temporary session keys for TLS or SSH, or persistent keys currently being used to perform cryptographic operations. Plaintext temporary session keys are generated on-the-fly as SSH and TLS sessions are created, and are not stored outside of RAM. Plaintext persistent keys are the decrypted contents of an encrypted persistent key file, which is kept in RAM while being used. The encrypted key is stored in the TOE underlying filesystem. Zeroization of keys in RAM is accomplished by overwriting with zeroes, followed by a read-verify. If the read-verification fails, the process is repeated.

Temporary keys are stored in RAM (volatile memory). These are zeroized when the associated session is ended.

Persistent keys are stored on SSDs. For SSD’s, the destruction of keys is executed by a block erase, followed by a read-verify. If this read verification fails, the process is repeated again.

The only private keys stored on SSDs are those used by sshd, the ssh tunnel client, and the TLS server. Zeroization of such keys is performed by executing the “zeroize-keyfile” command by the administrator when such keys are no longer required.

The host keys used by the sshd program are stored in /etc/ssh. The key type supported is RSA.

The key used for the SSH secure channel (for transporting audit data to a remote server) is stored in /opt/ssh-tunnel-client. The key type supported is RSA.

The key used by the TLS server is stored under /opt/TLS-server. The key type supported is RSA.

FCS_CKM.4

7.2.2 Cryptographic Operations

The TOE performs encryption and decryption in accordance with AES in CBC and GCM modes, using key sizes of 128 or 256 bits. These options are enforced in the OpenSSL configuration files, and are configured at the factory.

FCS_COP.1.1(1)

The TOE provides RSA digital signatures for RSA-based cryptographic functions and EC-based digital signatures for EC-based cryptographic functions. This is enforced in the OpenSSL configuration files, which are configured at the factory.

FCS_COP.1.1(2)

The TOE performs SHA-1, SHA-256, SHA-384, and SHA-512 hashing. This is enforced in the OpenSSL configuration files, which are configured at the factory. These hashes are used in RSA digital signatures; the keyed-hash message authentication code in SSH as both client and server, and in TLS; to perform password hashing; and to support file integrity checking. The TOE generates hashes with message digest sizes of 160, 256, 384, or 512 bits.

FCS_COP.1.1(3)

The TOE performs keyed-hash message authentication using HMAC-SHA1 (public key authentication for SSH-RSA), HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-512 and using cryptographic key sizes of 160, 256, 384, 512. HMAC message digest sizes of 160, 256, 384, or 512 bits are used, with block sizes of 64 bytes for SHA-1 and SHA-256, and 128 bytes for SHA-384 and SHA-512, per RFC2104 and RFC4868. All of these options are administrator-configurable where not mandated by a specific protocol or configuration. For instance, certain SSH integrity algorithms are specified by the SSH SFRs. This is enforced in the OpenSSL configuration files, which are configured at the factory.

FCS_COP.1.1(4)

7.2.3 Random Bit Generation

Random bit generation is provided by the crypto module, RHEL 7.4 OpenSSL v1.0.1k-8, which utilizes CTR_DRBG as defined by NIST SP 800-90A. This is not configurable, and there are no other cryptographic engines provided in the TOE. It has been determined that the entropy source (RDRAND) is sufficiently entropic. Since RDRAND itself is not FIPS certified, OpenSSL uses RDRAND as a source of high quality seeds for generating random numbers. For seeding/reseeding, the seed length used by OpenSSL is at least as much as the output block size, incremented in blocks of 160 bits, in order to ensure sufficient input entropy. Thus for the AES256 CTR DRBG used by OpenSSL for generating random numbers, the seed length is 320 bits. For key generation in OpenSSL, at least as many random bits as the key length are used, thus ensuring that the keys generated are sufficiently entropic.

7.2.4 SSH Client & Server Protocols

The TOE implements OpenSSH 6.6.1 p1, which is conformant to RFC 4251, 4252, 4523, 4554, and 6668. This is not configurable, and there are no other SSH modules or applications provided in the TOE.

FCS_SSHC_EXT.1.1 & FAU_STG_EXT.1.3

The TOE implements and supports SSH-RSA as its only supported public key authentication algorithm. If the TOE is not configured for public key authentication for a specified user, password-based authentication is provided. This is enforced in the OpenSSH configuration files, which are configured at the factory.

FCS_SSHC_EXT.1.2 & FCS_SSHS_EXT.1.2

The TOE detects packets of size greater than 262144 (256*1024) bytes in the SSH traffic stream, and aborts the session when such a packet is discovered. Packet size is detected by reading the size of the payload from the IP packet header. This behaviour is enforced in the OpenSSH and cannot be changed.

FCS_SSHC_EXT.1.3 & FCS_SSHS_EXT.1.3

The TOE implements SSHv2 transport connections with AES128-CBC and AES256-CBC modes. SSH supports SSH-RSA public key algorithms and no others, rejecting all other public key algorithms. Supported integrity algorithms are HMAC-SHA2-256 and HMAC-SHA2-512, and key exchange is performed using DH-Group14-SHA1 only. This is enforced in the OpenSSH configuration files, which are configured at the factory.

FCS_SSHC_EXT.1.4, FCS_SSHC_EXT.1.5, FCS_SSHC_EXT.1.6, and FCS_SSHC_EXT.1.7
&
FCS_SSHS_EXT.1.4, FCS_SSHS_EXT.1.5, FCS_SSHS_EXT.1.6, and FCS_SSHS_EXT.1.7

The TOEs SSH configuration also supports a ReKeyLimit parameter. By default, the ReKeyLimit parameter is set by at the factory, and enforces rekeying after 1 hour or 1 GB of data exchanged with the same key.

As a server, the ReKeyLimit parameter is enforced in the OpenSSH configuration files, which are configured at the factory. As a client, the ReKeyLimit parameter is enforced in the SSH configuration file in the underlying operating system, which governs all use of the SSH client.

FCS_SSHC_EXT.1.8 & FCS_SSHS_EXT.1.8

When acting as a client, the TOE authenticates the identity of the SSH server using a local database that associates each server hostname with its corresponding public key file as described in RFC4251 Section 4.1. This is accomplished by using the “known-hosts” file created by sshd and stored in the underlying filesystem. Should an offered public key not match the information stored in the known-hosts database, the SSH connection will be denied and an audit trail entry will be created to prompt the administrator to verify the information in the known-hosts file, the IP address and hostname of the desired SSH server, and perform any necessary updates to the known hosts file.

This behaviour is enforced in the OpenSSH configuration files, which are configured at the factory. An administrator may make authorized changes to the known-hosts file.

FCS_SSHC_EXT.1.9

7.2.5 TLS Server Protocol

The TOE implements TLSv1.2 according to RFC 5246, and supports the following ciphersuites:

- TLS_RSA_with_AES_128_CBC_SHA, defined in RFC 3268
- TLS_ECDHE_RSA_with_AES_256_GCM_SHA384, defined in RFC 5289

The TLS server provides a trusted path for performing administrative functions, during the course of which TLSv1.2 mechanisms are exercised.

This is enforced in the OpenSSL configuration files, which are configured at the factory.

FCS_TLSS_EXT.1.1

The TOE denies all connection requests from TLS version 1.1 or older, and SSLv3.0 and older. Only TLSv1.2 connections are supported. When a client requests an unsupported version of TLS, the TOE rejects the connection attempt by sending “handshake failure” and “invalid protocol version” error responses to the client. This behaviour is enforced in the OpenSSL configuration files, which are configured at the factory.

FCS_TLSS_EXT.1.2

For RSA-based key agreement in the first supported cipher, the TOE will negotiate key establishment using RSA with a key size of 2048 bits. For ECDHE key agreement in the second supported cipher, the TOE will negotiate key establishment using NIST curve secp256r1. This is enforced in the OpenSSL configuration files, which are configured at the factory.

FCS_TLSS_EXT.1.3

7.3 Identification and Authentication

7.3.1 Password Management

Passwords created for TOE authentication may be composed of all printable ASCII characters in UTF-8 formatting. For local console and SSH sessions, the administrative password length is configurable by changing the underlying RHEL 7.4 password policy, and may be configured to be 15 characters or greater. Such password policies are managed by the authenticated administrator, and are enforced by the login module. Password policies may have the following attributes:

- Minimum password lifetime
- Maximum password lifetime
- Minimum number of character types (i.e., Upper-case characters, lower case characters, etc.)
- Minimum password length
- Password history
- Priority (password policies are applied to user groups rather than individual user accounts; “priority” allows users who belong to multiple groups to fall under the purview of the strictest password policy applicable)
- Maximum consecutive failures before lockout
- Failure Delay interval
- Lockout time in seconds

Modifying the TOE password policy is done by changing the underlying RHEL 7.4 password policy. Authentication to the TOE is done by specifying a valid username and its associated password, which are identical between local and remote authentication.

FIA_PMG_EXT.1.1

7.3.2 User Identification and Authentication

The only supported authentication mechanisms are via the trusted path provided by SSH, TLS protected interface, or the local console.

For SSH remote administrative sessions, an administrator must connect to the TOE using their SSH client, which will negotiate a secure connection using the supported cryptographic operations.

For TLS remote administrative sessions, an administrator must connect to the TOE using a TLS client, which will negotiate a secure connection using the supported cryptographic operation.

During SSH or TLS remote administrative sessions, the administrator must authenticate using their username and password, or public key (for SSH sessions).

A successful login will be denoted by obtaining a command prompt; unsuccessful logons will result in the connection being dropped by the TOE. Administrators authenticate by providing their username and password, or by use of SSH-RSA public keys if configured. For password authentication, the provided password is hashed according to the underlying RHEL authentication mechanism, and the resultant hash is compared against the stored value for the user’s hashed password. If the hashes are the same, authentication for that user is successful. If the hash values are different, authentication fails.

If public key authentication is proposed, the TOE will initiate ssh-rsa authentication.

For Local administrative sessions, administrators authenticate by providing their username and password at the logon prompt. Successful authentication is denoted by obtaining a command prompt, while unsuccessful authentication results in a prompt to reauthenticate.

Before being required to authenticate to the local or remote console, a non-TOE entity may view the warning and consent banner in accordance with FTA_TAB.1, Respond to ICMP Echo Request, respond to ARP requests with ARP replies, make DNS Requests, respond to TLS ClientHello messages with TLS ServerHello messages on TCP port 27777, and respond to SSH login messages on TCP port 22.

FIA_UIA_EXT.1

7.3.3 Password-based Authentication Mechanism

The TOE provides a local password-based authentication mechanism to identify and verify users before allowing them to perform actions or execute commands on the TOE for both local and remote administrative sessions. This is provided by the underlying RHEL 7.4 user authentication component, which stores authentication data for remote and local console sessions in non-plaintext form in the /etc/shadow file in the underlying filesystem.

FIA_UAU_EXT.2.1

7.3.4 Protected Authentication Feedback

During authentication, the TOE obscures passwords by failing to echo any information back to the screen. This protects the administrator authentication data by revealing neither the content nor any related data regarding the administrator credentials (such as length or complexity).

FIA_UAU.7.1

7.3.5 X.509 Certificate Validation

The TOE validates x509v3 certificates according to the validation rules in RFC 5280, terminating with a trusted CA certificate. Certificates presented for validation may be validated via CRL as specified in RFC 5280. Such certificate validations are carried out using the OpenSSL module, and this behaviour is not configurable by the administrator. Certificates are checked for validity when the certificate is presented for TLS authentication. While only PEM encoding is supported for certificates, both PEM and DER encodings are supported for CRLs.

FIA_X509_EXT.1

7.3.6 X.509 Certificate Authentication

The TOE allows administrators to install x509v3 certificates for use in negotiating TLS connections, and [T1] specifies how such certificates are installed. Only one certificate may be installed at any time.

When certificates presented for TLS authentication cannot be validated, the TOE will deny the connection.

FIA_X509_EXT.2

7.3.7 X.509 Certificate Requests

An administrator may generate a Certificate Request Message as specified by RFC 2986, providing the following information in the request: public key, Common Name, Organization, Organizational Unit, and Country.

FIA_X509_EXT.3

7.4 Security Management

7.4.1 Management of Security Functions Behaviour

The TOE does not allow any but authorized and authenticated administrators to perform a manual update of the TOE firmware. Unless logged in to a local or remote administrative session, the commands required to initiate such updates are not available. In order to initiate the update process, the administrator must copy the candidate image to a specific file location within the TOE. Once the copy is completed, the administrator may issue the command to initiate the update process. The update process is more fully described in [ST] Section 7.5.4.

FMT_MOF.1.1(1)/TrustedUpdate

The TOE does not allow any but authorized and authenticated administrators to view or modify settings related to the handling of Audit data. Unless logged in to a local or remote administrative session, the commands required to modify audit functionality or interact in any way with the audit logs are unavailable. After authentication, the administrator may view or modify the current configuration of the auditing function, such as viewing or changing the destination syslog server in the operational environment.

FMT_MOF.1.1(2)/Audit

The TOE does not allow any but authorized and authenticated administrators to view or modify any security related settings, or in fact any settings at all. There are no options to manage the TOE or modify its behavior in any way prior to authenticating as an administrator.

After authentication, the administrator may modify the behaviour of the TSF, such as shutting down or rebooting the TOE, replacing the persistent cryptographic keys in use by the TOE, and/or changing the allowed or enabled ciphers for SSH and/or TLS.

FMT_MOF.1.1(1)/AdminAct

The TOE does not allow any but authorized and authenticated administrators to view or modify any services or functions. There are no options to manage the TOE or configure, start, or stop services in any way prior to authenticating as an administrator.

After authentication, the administrator may enable or disable any services offered by the TSF, such as the TLS service, the SSH service, the syslog service, and any background processes running in the underlying RHEL operating system. The administrator may also shutdown or reboot the TOE.

FMT_MOF.1.1(2)/AdminAct

7.4.2 Management of TSF Data

No administrative actions or functions are available prior to log in. Only those pre-authentication functions described below are permitted before forcing the user or non-TOE entity to authenticate:

- Display the warning banner in accordance with FTA_TAB.1;
- Respond to ICMP Echo Request, respond to ARP requests with ARP replies, make DNS Requests, respond to TLS ClientHello messages with TLS ServerHello messages on TCP port 27777, respond to SSH login messages on TCP port 22
- Respond to TLS ClientHello messages

FIA_UIA_EXT.1

The TOE does not allow any but authorized and authenticated administrators to view or modify the TSF data. There is no mechanism to interact with any TSF data at all prior to authenticating as an administrator.

FMT_MTD.1

The TOE does allow authorized and authenticated administrators to create, modify, import, and delete cryptographic keys. All security-related functions are performed by the authorized administrator(s). All actions related to cryptographic keys require first authenticating as an authorized administrator, and there are no mechanisms to interact with these keys prior to authenticating.

FMT_MTD.1.1/AdminAct

7.4.3 Specification of Management Functions

When authenticated as an authorized administrator via a local session, the administrator has the ability to perform the following functions:

- Configure the access banner
- Configure the session inactivity time before session termination
- Update the TOE, and to verify the updates using digital signature capability prior to installing those updates
- Configure the cryptographic functionality
- Start, stop, and restart services
- Configure Audit functionality

When authenticated as an authorized administrator via a remote session, the administrator has the ability to perform the following functions:

- Configure the access banner
- Configure the session inactivity time before session termination or locking
- Configure the cryptographic functionality
- Start, stop, and restart services
- Configure audit functionality

FMT_SMF.1

7.4.4 Restrictions on Security Roles

The TOE maintains the role of “administrator”. There are no user-level accounts on the TOE, and all accounts are administrator accounts. All administrators are security administrators for the purposes of this PP.

FMT_SMR.2.1

Since there is no cause to create a non-administrative user of the TOE, there are no other roles available. All users of the TOE’s administrative and management functions are authorized administrators, and only those personnel designated as authorized administrators may have user accounts on the TOE.

FMT_SMR.2.2

All administrators may administer the TOE locally and remotely.

FMT_SMR.2.3

7.5 Protection of the TSF

7.5.1 Protection of Administrator Passwords

All administrator passwords are subject to the requirements of FPT_APW_EXT.1. All passwords are stored in SHA512 hashed form, and there is no mechanism provided to read or display administrative password or authentication material.

FPT_APW_EXT.1

7.5.2 Protection of TSF Data (for reading of all symmetric keys)

Symmetric keys, private keys, and other CSPs are stored as protected files using the security permissions of the underlying file structure. There are no mechanisms that would allow an administrator to view these keys directly.

Administrative login passwords are stored as hashes that are not readable by anyone. During the login process, the plaintext password entered by the user is hashed, and the resulting hash compared to the stored hash in the password storage file in the underlying file structure. A successful match grants login, while and unsuccessful match results in authentication failure.

FPT_SKP_EXT.1

7.5.3 TSF Testing

The TOE performs a suite of self-tests upon start up or power on, and periodically during normal operation, and at the request of an authorized administrator. The TOE verifies that:

- The memory/RAM is functioning by writing known values to each register and reading these values back.
- RDRAND is responding as expected by performing the entropy health checking as prescribed in NIST SP 800-90B section 4.
- Software Integrity is valid, by computing the hash of the static system binaries and configuration files and comparing them to a stored value. If the values are identical, then none of the system binaries have changed and software integrity is intact.
- The cryptographic module is performing as expected, by executing Known Answer Tests for each algorithm. These tests are accomplished by encrypting a known value with each cipher, then decrypting it to verify that the decrypted value is as expected.

All tests are performed at startup, and the cryptographic tests may be requested during runtime by the TOE itself or at the request of an authorized administrator. The TSF may initiate periodic runtime cryptographic testing before performing any cryptographic operation (as required by the OpenSSL module when operating in FIPS mode).

Software integrity is checked continuously during runtime by the AIDE package (Advanced Intrusion Detection Environment, which is a file and directory integrity checker provided by the underlying RHEL OS). AIDE is initialized by computing the cryptographic hashes of the TOE configuration and executable files and storing these in a local database. During runtime, the AIDE package performs continuous cryptographic hash computation and comparison of these files against the initialized values stored in the local database.

In total, these tests ensure that the TSF is operating correctly at all times (having demonstrated that memory is operating as expected, the cryptographic module is operating correctly, none of the executable or configuration files have been modified, and that the entropy source is operating correctly). There are no other security-relevant TOE components to test. Should any of these tests fail, the administrator will be notified via alerts (syslog messages). The administrator should review the audit records for the status (Pass or Fail) of the self-tests that are auditable after successful boot-up of the TOE (note: RDRAND and memory self-tests are not auditable).

FPT_TST_EXT.1

7.5.4 Trusted Update

The TOE allows administrators to initiate the update process by downloading the update file from Bivio and following the specific instructions in the administrative guidance. No other mechanism is provided to update the TOE. The TOE also allows the administrator to query the currently running version of software at start up time. The administrator is instructed in [T1] to verify the published hash of the downloaded update file before installation. Verification of the published hash is done by executing a command to cause the TOE to generate the hash of the update candidate image stored in the underlying filesystem. This hash is displayed to the administrator for comparison against the published hash available on the Bivio website.

When the administrator verifies that the hashes are correct, the update process proceeds. If the administrator indicates that the hashes are not correct, the administrator must contact Bivio support to resolve the error.

When querying the running version of the firmware, the TOE reports the major version and the identification of the latest patch which was installed. Patches have an incrementing sequence number, and must be installed in that sequence.

FPT_TUD_EXT.1

7.5.5 Reliable Time Stamps

The TOE provides reliable time stamps for all operations. The TOE also allows the time to be adjusted by the administrator, who may set the time.

TSF functions that rely on accurate time are SSH and TLS negotiations (for verifying validity dates of keys or certificates, or the generating of time-based nonces), audit log timestamps, session timeouts, and X.509v3 Certificate verification (for verifying validity of certificates, ie, the certificate has not expired). All time stamps are provided by the system clock in the underlying RHEL 7.4 operating system.

Time is maintained and considered reliable by the hardware clock, which has been measured to provide less than 1.5 seconds of drift per calendar year. The administrator is instructed to configure the time at least once per year.

FPT_STM.1

7.6 TOE Access

7.6.1 TSF-initiated Session Locking

Local administrative sessions are terminated after an administrator-specified period of inactivity. This is configured using the TMOUT global environment variable, and is configurable by an administrator. To enforce the timeout, a count-up timer is started when an administrator logs in. After every action taken in an administrative session, the timer is reset. When the timer reaches the stored timeout value in seconds, the session is terminated. Administrative sessions are never locked, always terminated.

FTA_SSL_EXT.1

7.6.2 TSF-initiated Termination

Remote administrative sessions are terminated after an administrator-specified period of inactivity. This is enforced in the OpenSSH configuration files, using the "ClientAliveInterval" and "ClientAliveCountMax" settings in sshd.conf which is configurable by an administrator. To enforce the timeout, a count-up timer is started when an administrator logs in. After every action taken in an administrative session, the timer is reset. When the timer reaches the stored timeout value in seconds, the session is terminated. Administrative sessions are never locked, always terminated.

FTA_SSL.3

7.6.3 User-initiated Termination

Users may end their own sessions at any time by use of the "logout" command or by closing their SSH client or by ending their SSH session.

FTA_SSL.4

7.6.4 Default TOE Access Banners

The TOE allows TLS and SSH administration and local console sessions for administrative access. Whenever a user attempts to initiate an administrative session, they will be shown the administrator-configurable TOE Access and Warning Banner.

FTA_TAB.1

7.7 Trusted Path/Channels

7.7.1 Inter-TSF Trusted Channel

The TOE supports and enforces trusted channels that protect the communications between the TOE and a remote audit server from unauthorized disclosure or modification. It also supports trusted paths between itself and remote administrators so that the contents of administrative sessions are protected against unauthorized disclosure or modification.

The TOE achieves trusted channels by use of the SSHv2 Protocol which ensures the confidentiality and integrity of communication with the remote audit server. The TOE will initiate the connection, and mutual identification of the endpoints is guaranteed by using public key and or password based authentication for SSH. The SSHv2 protocol ensures that the data transmitted over a SSH session cannot be disclosed or altered by using the encryption and integrity mechanisms of the protocol. Full details regarding the allowed protocol options are discussed in Section 7.2.4 of this document. The TOE will act as both the SSH client and SSH server.

The TOE implements trusted paths by use of the SSHv2 protocol and/or TLSv1.2, which ensures the confidentiality and integrity of administrative sessions. The encrypted communication path between the TSF and a remote administrator is provided by the use of a SSH or TLSv1.2 session. Remote administrators initiate the connection to the TOE for both SSH or TLS connections. Assured identification of the endpoints is provided by using public key and or password based authentication for SSH. For TLSv1.2, the administrator authenticates the TOE via X.509v3 certificates, while the TOE authenticates the administrator with a password. The SSHv2 protocol ensures that data transmitted over a SSH session cannot be disclosed or altered by using the encryption and integrity mechanisms of the protocol. The TLSv1.2 protocol ensures that data transmitted over the TLS session cannot be disclosed or altered by using the encryption and integrity mechanisms of the supported ciphersuites as part of the TLSv1.2 protocol. For Trusted Paths, the TOE Is always the client of the protocol connection.

Local console access is gained using the Bivio-provided console cable between the console port on the TOE and the administrator workstation with an RS-232 port.

FTP_ITC.1 & TCP_TRP.1

8. Terms and Definitions

Table 9: TOE Abbreviations and Acronyms

Abbreviations/ Acronyms	Description
3DES-CBC	Triple DES in Chaining Block Cipher mode
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AES-NI	Intel Advanced Encryption Standard New Instructions
AIDE	Advanced Intrusion Detection Environment
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
CA	Certificate Authority
CAST	Carlisle Adams and Stafford Tavares
CAVP	Cryptographic Algorithm Validation Program
CBC	Chaining Block Cipher
CCRA	Common Criteria Recognition Arrangement
CLI	Command Line Interface
CMVP	Cryptographic Module Validation Program
CP	Central Processor
cPP	Collaborative Protecting Profile
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSP	Critical Security Parameter
CST	Computer Science and Technology
CVL	Component Validation List
DES	Data Encryption Standard
DH	Diffie-Hellman
DHE	Diffie-Hellman Exchange
DNS	Domain Name Service
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
E.G.	*Exempli Gratia*, “For the sake of example”
EC	Elliptic Curve
ECDHE	Elliptic Curve Diffie-Hellman Exchange
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electronically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standard
GB	Gigabytes
GCM	Galois/Counter Mode
HMAC	(Keyed-) Hash Message Authenticating Code

Table 9: TOE Abbreviations and Acronyms

Abbreviations/ Acronyms	Description
HTTP	Hyper Text Transfer Protocol
HTTPS	Secure HTTP
I&A	Identity and Authentication
IBM	International Business Machines
ICMP	Internet Control Message Protocol
IDEA	International Data Encryption Algorithm
IEC	International Electrotechnical Commission
IP	Internet Protocol
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
IV	Initialization Vector
J-PAKE	Password Authenticated Key Exchange by Juggling
KAT	Known Answer Test
KEK	Key Encrypting Key
MAC	Message Authentication Code
MD2	Message Digest Algorithm 2
MD4	Message Digest Algorithm 4
MD5	Message Digest Algorithm 5
MDC2	Modification Detection Code 2 (aka “Meyer-Schilling”)
NDcPP	Collaborative Protection Profile for Network Devices
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
NV	Non-Volatile
NVLAP	National Voluntary Laboratory Accreditation Program
OCSP	Online Certificate Status Protocol
OS	Operating System
PKCS	Public Key Cryptography Standard
POST	Power-On Self Tests
PSS	Probabilistic Signature Scheme
PUB	Publication
RACE	Row-based ASCII Compatible Encoding
RADIUS	Remote Authentication Dial-In User Service
RAM	Random Access Memory
RC2	Rivest Cipher 2
RC4	Rivest Cipher 4
RC5	Rivest Cipher 5
RFC	Request For Comment
RHEL	Red Hat Enterprise Linux
RIPEMD	RACE Integrity Primitives Evaluation Message Digest

Table 9: TOE Abbreviations and Acronyms

Abbreviations/ Acronyms	Description
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
RSASSA	RSA Signature Scheme with Appendix
SHA	Secure Hash Algorithm
SP	Special Publication
SSD	Solid State Disk
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
USB	Universal Serial Bus
UTF	Unicode Transformation Format

Table 10: CC Abbreviations and Acronyms

Abbreviations/ Acronyms	Description
AGD	Administrative Guidance Documentation
CC	Common Criteria
CCRA	Arrangement on the Recognition of Common Criteria Certificates in the field of IT Security
CEM	Common Criteria Evaluation Methodology
IT	Information Technology
NIAP	National Information Assurance Partnership
OSP	Organizational Security Policy
PP	Protection Profile
SAR	Security Assurance Requirement
SD	Supplemental Documentation
SFP	Security Function Policy
SFR	Security Functional Requirement
SPD	Security Policy Database
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification

9. References

Table 11: TOE Guidance Documentation

Reference	Description	Date
[AGD]	Bivio 6310-NC Common Criteria Administrative Guidance v1.3	March 8, 2018

Table 12: Common Criteria v3.1 References

Reference	Description	Version	Date
[C1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model CCMB-2009-07-001	V3.1 R4	July 2009
[C2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components CCMB-2009-07-002	V3.1 R4	July 2009
[C3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components CCMB-2009-07-003	V3.1 R4	July 2009
[C4]	Common Criteria for Information Technology Security Evaluation Evaluation Methodology CCMB-2009-07-004	V3.1 R4	July 2009

Table 13: Supporting Documentation

Reference	Description	Version	Date
[S1]	Collaborative Protection Profile for Network Devices	1.0	February 27, 2015
[S2]	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP	1.0	February 2015

Annex A Algorithm Validation Requirements

FCS_CKM.1.1

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.

Key Generation for FIPS PUB 186-4 RSA Schemes

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

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

a) Random Primes:
   - Provable primes
   - Probable primes
b) Primes with Conditions:
   - Primes p1, p2, q1, q2, p and q shall all be provable primes
   - Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
   - Primes p1, p2, q1, q2, p and q shall all be probable primes

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

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

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

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

and two ways to generate the cryptographic group generator g:

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

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

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

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

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

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

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

for each FFC parameter set and key pair.

FCS_CKM.2.1

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

SP800-56B Key Establishment Scheme Testing

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following assurance activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following assurance activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

b) The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

FCS_COP.1.1(1)

AES-CBC Known Answer Tests

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

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

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

```
# Input: PT, IV, Key
for i = 1 to 1000:
        if i == 1:
                CT[1] = AES-CBC-Encrypt(Key, IV, PT)
                PT = IV
else:
                CT[i] = AES-CBC-Encrypt(Key, PT)
                PT = CT[i-1]
```

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

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

AES-GCM Test

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

128 bit and 256 bit keys

a) Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
b) 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.
c) Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

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

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

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

FCS_COP.1.1(2)

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

FCS_COP.1.1(3)

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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