Klas Telecom Voyager Security Target

16-3277-R-0030

Version: 1.0

2017-09-11

Prepared For:

Klas Telecom, Inc.

1101 30th Street NW
Suite 500
Washington, DC 20007

Prepared By:

Brad Mitchell

UL Verification Services

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:	Klas Telecom Voyager Security Target
ST Version Number:	Version 1.0
ST Author(s):	Brad Mitchell
ST Publication Date:	2017-09-11
Keywords	Network Device, VPN

1.2 Target of Evaluation Reference

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

TOE Developer	Klas Telecom, Inc.
	1101 30th Street NW, Suite 500,
	Washington, DC 20007
TOE Name:	Klas Voyager
TOE Version	1.0

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) that also provides Virtual Private Network Gateway services.

1.3.2 TOE Usage

The Klas Voyager running KlasOS firmware provides the ability to securely encrypt data over WAN links using IPsec and FIPS Approved algorithms. A real-time clock is present on all KlasOS devices and NTP server and client is also a feature of the firmware. Authentication can be provided locally or over a trusted channel using IPSec or SSH and all logs can be securely sent to a syslog server. Access lists (ACL's) can filter all types of IP, TCP and UDP traffic. KlasOS provides a Command Line Interface (CLI) for device configuration.

The Klas Voyager range of products provide expandable, enterprise-grade rugged mobility solutions. The Klas Voyager embedded module is used in a variety of these Klas Voyager products provide the ability to establish highly secure IPSec tunnels using FIPS approved algorithms.

The TOE provides gigabit Ethernet, PoE, and layer 3 aware high-speed switching and removable storage using the VIK. WLAN communication is protected by IPsec VPN tunnels utilizing FIPS Approved AES encryption.

1.3.3 TOE Major Security Features Summary

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

1.3.4 TOE IT environment hardware/software/firmware requirements

The environment must be capable of providing one or more IPsec peers supporting the following algorithms and parameters:

- IKEv1 or IKEv2
- X.509v3 authentication supporting ECDSA P-256, P-384, or Pre-shared Key
- Symmetric ciphers: AES-CBC-128, AES-CBC-256, AES-GCM-128 or AES-GCM-256
- Integrity algorithms: HMAC-SHA-256 or HMAC-SHA-384
- Diffie-Hellman groups: 14, 19, 20 or 24

The environment must provide an RFC 5424 compliant syslog server. This server must be or be protected by an IPsec peer as defined above.

Similarly, if an NTP server is being used for time syncing then an IPsec tunnel must be established with the NTP server.

An administrative local console must be present to configure the TOE. The local console must provide a DB-9 serial port and be capable of supporting a VT-100 compatible terminal or emulator.

An administrative remote console may be used to administer the TOE over SSH. The remote console must implement SSH conformant to RFCs 4251, 4252, 4253, and 4254 and providing the following algorithms and parameters:

- Password authentication.
    - Optionally, ECDSA public key authentication using NIST curves P-256 or P-384.
- AES-CBC-128 or AES-CBC-256 encryption.
- HMAC-SHA1, HMAC-SHA2-256, or HMAC-SHA2-512 for message authentication.

Key exchange using Diffie-Hellman Group 14, ECDH over NIST P-256, or ECDH over NIST P-384.

1.4 Target of Evaluation Description

1.4.1 Target of Evaluation Physical Boundaries

The TOE consists of the following hardware:

- VoyagerESm
- VoyagerSW14

Running:

- KlasOS v5.1.0rc7

On:

ARMv5TE Feroceon Rev0 v5l processors

The VoyagerESm contains 4 FastEthernet ports, 1 GigabitEthernet port, 2 USB ports, 1 VIK slot, 1 FXS port and a console port.

The Voyager SW14 contains 12 FastEthernet ports, 2 GigabitEthernet ports, 1 VIK slot and a console port.

The use of the USB ports, VIK port, and FXS port was not evaluated as part of the Common Criteria validation.

The guidance documentation 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 can transmit audit data to an external IT entity using IPsec protocol.

1.4.2.2 Cryptographic Operations

The TSF performs the following cryptographic operations:

- IKEv1, IKEv2 and ESP with AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256 for IPsec

- X509v3 with ECDSA P-256 or P-384 for IPsec
- SSHv2 using AES-CBC-128 or AES-CBC-256 for encryption; DH Group 14 or NIST P-256 / P-384 for key exchange; HMAC SHA1, HMAC-SHA2-256, or HMAC-SHA2-512 for message authentication; EC public key authentication using NIST P-256 or P-384.

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

1.4.2.4 Security Management

The TOE provides secure administrative services for management of general TOE configuration and the security functionality provided by the TOE. All TOE administration occurs via a local console connection. The TOE provides the ability to securely manage:

- All TOE administrative users
- All identification and authentication
- All audit functionality of the TOE
- All TOE cryptographic functionality
- Timestamps maintained by the TOE
- Update to the TOE
- TOE configuration files

Only one administrative user can be created on the TOE, and the administrative user can perform all of the above security relevant management functions. Administrators can create configurable login banners to be displayed at time of login and can also define an inactivity timeout to terminate sessions after a set period of inactivity.

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 digital signature mechanism prior to installing those updates.

1.4.2.6 *Packet Filtering*

The TOE filters packets received on the VLAN interfaces. The TOE can be configured to allow or deny the packet based on IP source address, IP destination address, TCP or UDP source port, TCP or UDP destination port

1.4.2.7 TOE Access

- The TOE, for local interactive sessions, shall 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.

1.4.2.8 Trusted Path/Channels

- The TOE uses IPsec and 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 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 #;" those taken from the VPN Gateway Extended Package are marked as "EP 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 underlined text**.** Selections within selections will be identified with double underline text.

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***~~.

Certain refinements – as mandated by the requirements of VPN GW EP v2.1 – are indicated with the presence of "***ST Author Notes***" below the affected SFRs.

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, February 27, 2015 [PP]. This Protection Profile will be referred to as cPP or PP for convenience throughout this Security Target. This [ST] conforms to the following NIAP-issued Technical Decisions:

- TD0228: NIT Technical Decision for CA certificates - basicConstraints validation
- TD0225: NIT Technical Decision for Make CBC cipher suites optional in IPsec[1]
- TD0224: NIT Technical Decision Making DH Group 14 optional in FCS_IPSEC_EXT.1.11[2]
- TD0223: NIT Technical Decision for "Expected" vs "unexpected" DNs for IPsec Communications
- TD0209: Additional DH Group added as selection for IKE Protocols[3]
- TD0199: NIT Technical Decision for Elliptic Curves for Signatures
- TD0189: NIT Technical Decision for SSH Server Encryption Algorithms
- TD0188: NIT Technical Decision for Optional use of X.509 certificates for digital signatures[4]
- TD0187: NIT Technical Decision for Clarifying FIA_X509_EXT.1 test 1[5]
- TD0186: NIT Technical Decision for Applicability of X.509 certificate testing to IPsec[6]
- TD0185: NIT Technical Decision for Channel for Secure Update.[7]
- TD0184: NIT Technical Decision for Mandatory use of X.509 certificates[8]
- TD0183: NIT Technical Decision for Use of the Supporting Document[9]
- TD0182: NIT Technical Decision for Handling of X.509 certificates related to ssh-rsa and remote comms.[10]
- TD0181: NIT Technical Decision for Self-testing of integrity of firmware and software.[11]
- TD0179: Management Capabilities in VPN GW EP 2.1

[1] This TD modifies the text of FCS_IPSEC_EXT.1.4 and .1.6; however, these SFRs are superseded by their counterparts in [EP] and do not appear to be modified in this [ST].

[2] This TD modifies the text of FCS_IPSEC_EXT.1.11; however, this SFR is superseded by its counterpart in [EP] and does not appear to be modified in this [ST].

[3] This TD adds additional selections that do not appear in this TOE and, therefore, are not used in this [ST].

[4] This TD does not modify any SFR text in this [ST].

[5] This TD does not modify any SFR text in this [ST].

[6] This TD does not modify any SFR text in this [ST].

[7] This TD does not modify any SFR text in this [ST].

[8] This TD does not modify any SFR text in this [ST].

[9] This TD does not modify any SFR text in this [ST].

[10] This TD does not modify any SFR text in this [ST].

[11] This TD does not modify any SFR text in this [ST].

- TD0169: NIT Technical Decision for Compliance to RFC5759 and RFC5280 for using CRLs
- TD0168: NIT Technical Decision for Mandatory requirement for CSR generation[12]
- TD0167: NIT Technical Decision for Testing SSH 2^28 packets
- TD0164: NIT Technical Decision for Negative testing for additional ciphers for SSH[13]
- TD0160: NIT Technical Decision for Transport mode and tunnel mode in IPSEC communications
- TD0154: NIT Technical Decision for Versions of TOE Software in the NDcPP v1.0 and FW cPP v1.0
- TD0153: NIT Technical Decision for Auditing of NTP Time Changes in the NDcPP v1.0 and FW cPP v1.0[14]
- TD0150: NIT Technical Decision for Removal of SSH re-key audit events in the NDcPP v1.0 and FW cPP v1.0
- TD0130: NIT Technical Decision for Requirements for Destruction of Cryptographic Keys
- 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
- TD0115: NIT Technical Decision for Transport mode and tunnel mode in IPsec communication in NDcPP and FWcPP[15]
- TD0114: NIT Technical Decision for Re-Use of FIPS test results in NDcPP and FWcPP[16]
- TD0111: NIT Technical Decision for third party libraries and FCS_CKM.1 in NDcPP and FWcPP[17]
- TD0095: NIT Technical Interpretations regarding audit, random bit generation, and entropy in NDcPP[18]
- TD0090: NIT Technical Decision for FMT_SMF.1.1 Requirement in NDcPP

2.3 Conformance to Security Packages

This Security Target extends the cPP security claims with the VPN Gateway Extended Package Version 2.1, dated March 8, 2017 [EP]. This Extended Package will be referred to as VPNEP or EP throughout this Security Target. This Security Target is VPNEP-conformant.

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

[12] This TD does not modify any SFR text in this [ST].

[13] This TD does not modify any SFR text in this [ST].

[14] This TD does not modify any SFR text in this [ST].

[15] This TD does not modify any SFR text in this [ST].

[16] This TD does not modify any SFR text in this [ST].

[17] This TD does not modify any SFR text in this [ST].

[18] This TD does not modify any SFR text in this [ST].

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

o No additional OSPs have been defined in this ST.

- Assumptions
  - 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.
T.NETWORK_DISCLOSURE	Sensitive information on a protected network might be disclosed resulting from ingress- or egress-based actions.
T. NETWORK_ACCESS	Unauthorized access may be achieved to services on a protected network from outside that network, or alternately services outside a protected network from inside the protected network.
T.NETWORK_MISUSE	Access to services made available by a protected network might be used counter to Operational Environment policies.
T.REPLAY_ATTACK	If malicious or external IT entities are able to gain access to the network, they may have the ability to capture information traversing throughout the network and send them on to the intended receiver.
T.DATA_INTEGRITY	A malicious party attempts to change the data being sent – resulting in loss of integrity.

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

Table 3: Assumptions

Assumption	Description
	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 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.
A.CONNECTIONS	It is assumed that the TOE is connected to distinct networks in a manner that ensures that the TOE security policies will be enforced on all applicable network traffic flowing among the attached networks.

4. Security Objectives

4.1 Security Objectives for the TOE

Table 4: Security Objectives for the TOE

Objective	Description
O.ADDRESS_FILTERING	The TOE will provide the means to filter and log network packets based on source and destination addresses.
O.TOE_ADMINISTRATION	The TOE will provide the functions necessary for an administrator to configure the packet filtering rules, as well as the cryptographic aspects of the Ipsec protocol that are enforced by the TOE.
O.AUTHENTICATION	The TOE will provide a means to authenticate the user to ensure they are communicating with an authorized external IT entity.
O.CRYPTOGRAPHIC_FUNCTIONS	The TOE will provide means to encrypt and decrypt data as a means to maintain confidentiality and allow for detection and modification of TSF data that is transmitted outside of the TOE.
O.FAIL_SECURE	Upon a self-test failure, the TOE will shutdown to ensure data cannot be passed while not adhering to the security policies configured by the administrator.
O.PORT_FILTERING	The TOE will provide the means to filter and log network packets based on source and destination transport layer ports.
O.SYSTEM_MONITORING	The TOE will provide the means for administrators to configure packet filtering rules to 'log' when network traffic is found to match the configured rule.

4.2 Security Objectives for the Operational Environment

Table 5: 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.
OE.CONNECTIONS	TOE administrators will ensure that the TOE is installed in a manner that will allow the TOE to effectively enforce its policies on network traffic flowing among attached networks.

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 extended, 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 6: Security Functional Requirements

SFR	Description
FAU_GEN.1	Audit Data Generation
FAU_GEN.2	User Audit Association
FAU_STG_EXT.1	Protected Audit Event Storage
FCS_CKM.1(1)	Cryptographic Key Generation (for asymmetric keys)
FCS_CKM.1/IKE	Cryptographic Key Generation (for asymmetric keys)
FCS_CKM.2	Cryptographic Key Establishment (Refined)
FCS_CKM.4	Cryptographic Key Destruction
FCS_COP.1(1)	Cryptographic Operation (AES Data Encryption/Decryption) (for Data Encryption/Decryption)
FCS_COP.1(2)	Cryptographic Operation (Signature Generation and Verification) (for cryptographic signature)
FCS_COP.1(3)	Cryptographic Operation (Hash Algorithm)
FCS_COP.1(4)	Cryptographic Operation (Keyed Hash Algorithm)
FCS_IPSEC_EXT.1	Ipsec Protocol Ipsec Communications
FCS_RBG_EXT.1	Cryptographic Operation (Random Bit Generation)
FCS_RBG_EXT.1.2	Random Bit Generation
FCS_SSHS_EXT.1	SSH Server Protocol (selection based)
FIA_AFL.1	Authentication Failure Handling
FIA_PMG_EXT.1	Password Management
FIA_PSK_EXT.1	Pre-Shared Keys
FIA_UIA_EXT.1	User Identification and Authentication
FIA_UAU_EXT.2	Password-based Authentication Mechanism
FIA_UAU.7	Protected Authentication Feedback
FIA_X509_EXT.1	X.509 Certificate Validation
FIA_X509_EXT.2	X.509 Certificate Authentication
FIA_X509_EXT.3	X.509 Certificate Requests
FIA_EXT_EXT.4	X.509 Certificate Identity
FMT_MOF.1(1)/Trusted Update	Management of Security Functions Behaviour

Table 6: Security Functional Requirements

SFR	Description
FMT_MOF.1(1)/Audit	Management of Security Functions Behaviour (optional)
FMT_MOF.1(1)/AdminAct	Management of Security Functions Behaviour
FMT_MOF.1(2)/AdminAct	Management of Security Functions Behaviour
FMT_MTD.1	Management of TSF Data
FMT_MTD.1/AdminAct	Management of TSF Data
FMT_SMF.1	Specification of Management Functions
FMT_SMR.2	Restrictions on Security Roles
FPF_RUL_EXT.1	Packet Filtering
FPT_APW_EXT.1	Protection of Administrator Passwords
FPT_FLS.1	Fail Sucure
FPT_SKP_EXT.1	Protection of TSF Data (for reading all symmetric keys)
FPT_STM.1	Reliable Time Stamps
FPT_TST_EXT.1	TSF Testing
FPT_TST_EXT.2	TSF Testing Extended: TSF Testing
FPT_TUD_EXT.1	Trusted Update
FTA_SSL_EXT.1	TSF-initiated Session Locking
FTA_SSL.3	TSF-initiated Termination
FTA_SSL.4	User-initiated Termination
FTA_TAB.1	Default TOE Access Banners
FTP_ITC_EXT.1	Inter-TSF Trusted Channel
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)
- <u>No other actions;</u>

d) Specifically defined auditable events listed in Table.

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 1.

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.

SFR	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	None.
FAU_GEN.2	None.	None.
FAU_STG_EXT.1	None.	None.
FCS_CKM.1	None.	None.
FCS_CKM.2	None.	None.
FCS_CKM.4	None.	None.
FCS_COP.1(1)	None.	None.
FCS_COP.1(2)	None.	None.
FCS_COP.1(3)	None.	None.
FCS_COP.1(4)	None.	None.

SFR	Auditable Events	Additional Audit Record Contents
FCS_IPSEC_EXT.1 (selection based)	Failure to establish an Ipsec SA.	Reason for failure
FCS_IPSEC_EXT.1	Session Establishment with peer	VPN GW: Entire packet contents of packets transmitted / received during session establishment.
FCS_RBG_EXT.1	None.	None.
FCS_SSHS_EXT.1 (selection based)	Failure to establish an SSH session	Reason for failure.[19]
FIA_PMG_EXT.1	None.	None.
FIA_UIA_EXT.1	All use of the identification and authentication mechanism.	Provided user identity, origin of the attempt (e.g., IP address).
FIA_UAU_EXT.2	All use of the identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
FIA_UAU.7	None.	None.
FIA_X509_EXT.1	Unsuccessful attempt to validate a certificate.	Reason for Failure
FIA_X509_EXT.1	Establishing session with CA	VPN GW: Entire packet contents of packets transmitted / received during session establishment.
FIA_X509_EXT.2	None.	None.
FIA_X509_EXT.3	None.	None.
FIA_X509_EXT.4	None.	None.
FMT_MOF.1(1)/ TrustedUpdate	Any attempt to initiate a manual update.	None.
FMT_MOF.1(1)/ Audit (Optional)	Modification of the behaviour of the transmission of audit data to an external identity.	None.
FMT_MOF.1(1)/ AdminAct	Modification of the behavior of the TSF.	None.
FMT_MOF.1(2)/ AdminAct	Starting and stopping of services.	None.
FMT_MTD.1	All management activities of TSF data.	None.
FMT_MTD.1/ AdminAct	Modification, deletion, generation/import of cryptographic keys.	None.
FMT_SMF.1	None.	None.
FMT_SMR.2	None.	None.
FPF_RUL_EXT.1	Application of rules configured with the 'log' operation	Source and destination addresses Source and destination ports Transport Layer Protocol TOE Interface

[19] Modified by TD0150

SFR	Auditable Events	Additional Audit Record Contents
	Indication of packets dropped due to too much network traffic	TOE interface that is unable to process packets
FPT_SKP_EXT.1	None.	None.
FPT_APW_EXT.1	None.	None.
FPT_FLS.1/ LocSpace	None.	None.
FPT_TST_EXT.1	None.	None.
FPT_TST_EXT.2	None.	None.
FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure)	No additional information.
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).
FTA_SSL_EXT.1	Any attempts at unlocking of an interactive session.	None.
FTA_SSL.3	The termination of a remote session by the session locking mechanism.	None.
FTA_SSL.4	The termination of an interactive session.	None.
FTA_TAB.1	None.	None.
FTP_ITC_EXT_EXT.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.
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 3 and Table 4. 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

EP Application Note

For session establishment, the expectation is that the TOE is capable of auditing all of the packets associated with the establishment of a session; this would include the IKE phase 1 and phase 2 *negotiations. The TOE must be able to log all of the packets in a successful session establishment, and also have the ability to log any packets that were dropped or discarded.*

ST Author Note

FAU_GEN.1 has been refined in accordance with the requirements of NDcPP VPN GW EP v2.1, section 4.2.1.

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.

6.1.1.3 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_EXT_EXT.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: **Delete the current logfile and start a new one** 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'.

6.1.2 Class FCS: Cryptographic Support

6.1.2.1 FCS_CKM.1(1) Cryptographic Key Generation

FCS_CKM.1.1(1)

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

- ECC schemes using "NIST curves" P-256, P-384 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;
- FFC schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1

Application Note 7:

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 RSA key establishment scheme, the TOE does not need to implement RSA key generation.

6.1.2.2 FCS_CKM.1/IKE Cryptographic Key Generation (for IKE Peer Authentication)

FCS_CKM.1.1/IKE

The TSF shall generate asymmetric cryptographic keys used for IKE peer authentication in accordance with a:

- 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;
- FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4 for ECDSA schemes and implementing “NIST curves” P-256, P-384 and no other curves;.

EP Application Note

The ANSI X9.31-1998 option will be removed from the selection in a future publication of this document. Presently, the selection is not exclusively limited to the FIPS PUB 186-3 options in order to allow industry some further time to complete the transition to the modern FIPS PUB 186-3 standard.

The keys that are required to be generated by the TOE through this requirement are intended to be used for the authentication of the VPN peers during the IKE (either v1 or v2) key exchange. While it is required that the public key be associated with an identity in an X509v3 certificate, this association is not required to be performed by the TOE, and instead is expected to be performed by a Certificate Authority in the Operational Environment.

As indicated in FCS_IPSEC_EXT.1, the TOE is required to implement support RSA or ECDSA (or both) for peer authentication.

The generated key strength of 2048-bit RSA keys need to be equivalent to, or greater than, a symmetric key strength of 112 bits. See NIST Special Publication 800-57, “Recommendation for Key Management” for information about equivalent key strengths.

6.1.2.3 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:

- 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”;
- Finite field-based key establishment schemes that meets the following: NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”

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.

The RSA-based key establishment schemes are described in Section 9 of NIST SP 800-56B; however, Section 9 relies on implementation of other sections in SP 800-56B. If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

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.

6.1.2.4 FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM.4.1 [20]

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;*

• *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 *instructs a part of the TSF to destroy the abstraction that represents the key*

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 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.

[20] Modified by TD0130

6.1.2.5 FCS_COP.1(1) Cryptographic Operation (for 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 and no other key sizes 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.

EP Application Note

This SFR has been modified from its definition in the NDcPP by mandating both GCM and CBC modes as well as both 128 and 256 bit key sizes at a minimum.

ST Author Note

FCS_COP.1(1) has been refined in accordance with the requirements of NDcPP VPN GW EP v2.1, section 4.2.2.

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

FCS_COP.1.1(2) [21,22]

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) **3072 bits.**
- Elliptic Curve Digital Signature Algorithm and cryptographic key sizes **256 bits or 384 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 RSASSAPKCS1v1_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, P-384, and no other curves; 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

[21] Modified by TD0199

[22] Modified by TD0116

other FCS requirements, especially when supporting elliptic curves.

6.1.2.7 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).

6.1.2.8 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 HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 and cryptographic key sizes **160-512 bits** and message digest sizes 160, 256, 384, 512 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.

6.1.2.9 FCS_IPSEC_EXT.1 Ipsec Protocol

FCS_IPSEC_EXT.1.1

The TSF shall implement the Ipsec architecture as specified in RFC 4301.

Application Note 52

RFC 4301 calls for an Ipsec implementation to protect IP traffic through the use of a Security Policy Database (SPD). The SPD is used to define how IP packets are to be handled: PROTECT the packet (e.g., encrypt the packet), BYPASS the Ipsec services (e.g., no encryption), or DISCARD the packet (e.g., drop the packet). The SPD can be implemented in various ways, including router access control lists, firewall rulesets, a "traditional" SPD, etc. Regardless of the implementation details, there is a notion of a "rule" that a packet is "matched" against and a resulting action that takes place.

While there must be a means to order the rules, a general approach to ordering is not mandated, as long as the SPD can distinguish the IP packets and apply the rules accordingly. There may be multiple SPDs (one for each network interface), but this is not required.

FCS_IPSEC_EXT.1.2

The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.3 [23]

The TSF shall implement transport mode, tunnel mode.

Application Note 57

The selection of supported modes shall be performed according to RFC 4301. The TSS shall provide details about the supported modes.

EP Application Note

Future versions of this EP will require that the TSF implement both tunnel mode and transport mode.

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) and AES-GCM-128 (specified in RFC 4106), AES-GCM-256 (specified in RFC 4106) together with a Secure Hash Algorithm (SHA)-based HMAC.

ST Author Note

FCS_IPSEC_EXT.1.4 has been refined in accordance with the requirements of NDcPP VPN GW EP v2.1, section 4.2.3. TD0225 modifies this requirement in the [PP], but the [EP] supersedes it and is used here.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol:

- IKEv1, using Main Mode for Phase 1 exchanges, as defined in RFCs 2407, 2408, 2409, RFC 4109, no other RFCs for extended sequence numbers, and RFC 4868 for hash functions;
- IKEv2 as defined in RFC 5996 and with mandatory support for NAT traversal as specified in RFC 5996, section 2.23), and RFC 4868 for hash functions.

Application Note 53

If the TOE implements SHA-2 hash algorithms for IKEv1 or IKEv2, the ST author selects RFC 4868. If the ST author selects IKEv1, FCS_IPSEC_EXT.1.15 must also be included in the ST. IKEv2 will be required for those TOEs entering evaluation after Quarter 3, 2016.

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the IKEv1, IKEv2 protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 3602 and AES-GCM-128, AES-GCM-256 as specified in RFC 5282.

Application Note 54

AES-GCM-128 and AES-GCM-256 may only be selected if IKEv2 is also selected, as there is no RFC defining AES-GCM for IKEv1.

FCS_IPSEC_EXT.1.7

The TSF shall ensure that:

- IKEv1 Phase 1 SA lifetimes can be configured by an Security Administrator based on :

[23] Modified by TD0160

- length of time, where the time values can configured within **1-24** hours;
- IKEv2 SA lifetimes can be configured by an Security Administrator based on:
  - length of time, where the time values can configured within **1-24** hours

Application Note 55

The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5). The ST author chooses either volume-based lifetimes or time-based lifetimes (or a combination). This requirement must be accomplished by providing Security Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE). Hardcoded limits do not meet this requirement. In general, instructions for setting the parameters of the implementation, including lifetime of the Sas, should be included in the guidance documentation generated for AGD_OPE.

FCS_IPSEC_EXT.1.8

The TSF shall ensure that:

- IKEv1 Phase 2 SA lifetimes can be configured by a Security Administrator based on:
  - Number of bytes;length of time, where the time values can be configured within 1-720 hours
- IKEv2 Child SA lifetimes can be configured by a Security Administrator based on:
  - Number of bytes;
  - length of time, where the time values can be configured within **1-720** hours

Application Note 56

The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5). The ST author chooses either volume-based lifetimes or time-based lifetimes (or a combination). This requirement must be accomplished by providing Security Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE). Hardcoded limits do not meet this requirement. In general, instructions for setting the parameters of the implementation, including lifetime of the Sas, should be included in the guidance documentation generated for AGD_OPE.

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange (“x” in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least **224, 256, or 384** bits.

Application Note 57

For DH groups 19 and 20, the “x” value is the point multiplier for the generator point G.

Since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the Sas, the assignment in FCS_IPSEC_EXT.1.9 may contain multiple values. For each DH group supported, the ST author consults Table 2 in NIST SP 800-57 “Recommendation for Key Management – Part 1: General” to determine the security strength (“bits of security”) associated with the DH group. Each unique value is then used to fill in the assignment for this element. For example, suppose the implementation *supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192.*

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKEv1, IKEv2 exchanges of length:

- at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash.

Application Note 58

The ST author must select the second option for nonce lengths if IKEv2 is also selected (as this is mandated in RFC 5996). The ST author may select either option for IKEv1. For the first option for nonce lengths, since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the Sas, the assignment in FCS_IPSEC_EXT.1.10 may contain multiple values. For each DH group supported, the ST author consults Table 2 in NIST SP 800-57 "Recommendation for Key Management–Part 1: General" to determine the security strength ("bits of security") associated with the DH group. Each unique value is then used to fill in the assignment for this element. For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192.

Because nonces may be exchanged before the DH group is negotiated, the nonce used should be large enough to support all TOE-chosen proposals in the exchange.

FCS_IPSEC_EXT.1.11

The TSF shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), 19 (256-bit Random ECP), 20 (384-bit Random ECP), and 24 (2048-bit MODP with 256-bit POS).

Application Note 59

The selection is used to specify additional DH groups supported. This applies to IKEv1 and IKEv2 exchanges. For products entering into evaluation after Quarter 3, 2015, DH Group 19 (256-bit Random ECP) and DH Group 20 (384-bit Random ECP) will be required. It should be noted that if any additional DH groups are specified, they must comply with the requirements (in terms of the ephemeral keys that are established) listed in FCS_CKM.1.

ST Author Note

FCS_IPSEC_EXT.1.11 has been refined in accordance with the requirements of NDcPP VPN GW EP v2.1, section 4.2.3. TD0224 modifies this SFR in [PP], but the version in [EP] TD0209 supersedes and is used here.

FCS_IPSEC_EXT.1.12

The TSF shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the IKEv1 Phase 1, IKEv2 IKE_SA connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the IKEv1 Phase 2, IKEv2 CHILD_SA connection.

Application Note 60

The ST author chooses either or both of the IKE selections based on what is implemented by the TOE. Obviously, the IKE version(s) chosen should be consistent not only in this element, but with other choices for other elements in this component. While it is acceptable for this capability to be configurable, the default configuration in the evaluated configuration (either "out of the box" or by configuration guidance in the AGD documentation) must enable this functionality.

FCS_IPSEC_EXT.1.13

The TSF shall ensure that all IKE protocols perform peer authentication using RSA, ECDSA that use X.509v3 certificates that conform to RFC 4945 and Pre-shared Keys.

Application Note 61

At least one public-key-based Peer Authentication method is required in order to conform to this PP; one or more of the public key schemes is chosen by the ST author to reflect what is implemented. The ST author also ensures that appropriate FCS requirements reflecting the algorithms used (and key generation capabilities, if provided) are listed to support those methods. Note that the TSS will elaborate on the way in which these algorithms are to be used (for example, RFC 2409 specifies three authentication methods using public keys; each one supported will be described in the TSS). Peer authentication using ECDSA X.509v3 certificates will be required for TOEs entering evaluation after Quarter 3, 2015.

FCS_IPSEC_EXT.1.14 [24]

The TSF shall only establish a trusted channel if the presented identifier in the received certificate matches the configured reference identifier, where the presented and reference identifiers are of the following types: Distinguished Name (DN) and no other reference identifier type.

Application Note 62

Supported peer certificate algorithms are the same as FCS_IPSEC_EXT.1.1.

6.1.2.10 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 CTR_DRBG (AES).

FCS_RBG_EXT.1.2

The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from **5** software based noise source, **1** hardware-based noise source with a minimum of 256 bits 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.

[24] Modified by TD0223

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.

6.1.2.11 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 5656, 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.

FCS_SSHS_EXT.1.3

The TSF shall ensure that, as described in RFC 4253, packets greater than **35,000** bytes in an SSH transport connection are dropped.

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.

FCS_SSHS_EXT.1.4 [25]

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.

FCS_SSHS_EXT.1.5

The TSF shall ensure that the SSH transport implementation uses ecdsa-sha2-nistp256 and ecdsa-sha2-nistp384 as its public key algorithm(s) and rejects all other public key algorithms.

Application Note 72

[25] Modified by TD0189

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.

FCS_SSHS_EXT.1.6

The TSF shall ensure that the SSH transport implementation uses hmac-sha1, hmac-sha2-256, hmac-sha2-512 and no other MAC algorithms as its 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.

FCS_SSHS_EXT.1.7

The TSF shall ensure that diffie-hellman-group14-sha1, ecdh-sha2-nistp256 and ecdh-sha2-nistp384 are the only allowed key exchange methods used for the SSH protocol.

FCS_SSHS_EXT.1.8 [26]

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 onegigabyte 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.

ST Author Note:
This SFR text is copied exactly from TD0167, including typographical errors present.

6.1.3 Class FIA: Identification and Authentication

6.1.3.1 FIA_AFL.1 Authentication Failure Handling

FIA_AFL.1.1

[26] Modified by TD0167

The TSF shall detect when an Administrator configurable positive integer of successive unsuccessful authentication attempts occur related to administrators attempting to authenticate remotely.

FIA_AFL.1.2

When the defined number of unsuccessful authentication attempts has been met, the TSF shall prevent the offending remote administrator from successfully authenticating until **manual account unlocking** is taken by a local Administrator.

EP Application Note

This requirement does not apply to an administrator at the local console, since it does not make sense to lock a local administrator's account in this fashion. This could be addressed by (for example) requiring a separate account for local administrators or having the authentication mechanism implementation distinguish local and remote login attempts. The "action" taken by a local administrator is implementation specific and would be defined in the administrator guidance (for example, lockout reset or password reset). The ST author chooses one of the selections for handling of authentication failures depending on how the TOE has implemented this handler.

6.1.3.2 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: "!", "@", "#", "$", "%", "^", "&", "*", "(", ")";
b) Minimum password length shall be settable by the Security Administrator, and shall support passwords of **15-128 characters**.

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.

6.1.3.3 FIA_PSK_EXT.1 Password Management

FIA_PSK_EXT.1.1

The TSF shall be able to use pre-shared keys for IPsec and no other protocols.

FIA_PSK_EXT.1.2

The TSF shall be able to accept text-based pre-shared keys that:

- Are 22 characters and **up to 128 characters**;
- Are composed of any combination of upper and lower case letters, numbers, and special characters (that include: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")").

FIA_PSK_EXT.1.3

The TSF shall condition the text-based pre-shared keys by using SHA-256, **SHA-384.**

FIA_PSK_EXT.1.4

The TSF shall be able to accept bit-based pre-shared keys.

6.1.3.4 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 and DCHP requests, send and receive DNS Request, send and receive ARP requests, send and receive Spanning Tree Protocol packets, send and receive Cisco Discovery Protocol packets, send and receive NTP packets, send and receive SIP packets, send and receive SSH packets, send and receive IPsec packets.**

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_EXT.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.

6.1.3.5 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.

6.1.3.6 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.

6.1.3.7 FIA_X509_EXT.1 X.509 Certificate Validation

FIA_X509_EXT.1.1 [27]

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 5759 Section 5.
- 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[28]

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'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).

FIA_X509_EXT.1.2

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.

[27] Modified by TD0169

[28] Modified by TD0117

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.

6.1.3.8 FIA_X509_EXT.2 X.509 Certificate Authentication

FIA_X509_EXT.2.1

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

Application Note 20

The ST author's selection matches the selection of FTP_ITC_EXT.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.

6.1.3.9 FIA_X509_EXT.3 X.509 Certificate Requests

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 device-specific information, Common Name, Organization, Organizational Unit, Country.

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.

6.1.3.10 FIA_X509_EXT.4 X.509 Certificate Identity

FIA_X509_EXT.4.1

The TSF shall not establish an SA if the distinguished name (DN) contained in a certificate does not match the expected DN for the entity attempting to establish a connection.

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.

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

FMT_MOF.1.1(1)/Audit

The TSF shall restrict the ability to determine the behaviour of, modify the behaviour of the functions transmission of audit data to an external IT entity to Security Administrators.

Application Note 43

FMT_MOF.1(1)/Audit should always be chosen if the transmission protocol for transmission of audit data to an external IT entity as defined in FAU_STG_EXT.1.1 is configurable.

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

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.

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

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.

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.

6.1.4.6 FMT_MTD.1/AdminAct Management of TSF Data

FMT_MTD.1.1/AdminAct

The TSF shall restrict the ability to modify, delete, generate/import the cryptographic keys and certificates used for VPN operation 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.

ST Author Note

This SFR has been refined in accordance with the VPN GW EP v2.1

6.1.4.7 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 [29,30,31]

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 digital signature and <u>no other</u> capability prior to installing those updates;*
- *<u>Ability to configure the list of TOE-provided services available before an entitity is identified and authenticated, as specified in FIA_UIA_EXT.1;</u>*
- *Ability to configure firewall rules;*
- *Ability to configure the cryptographic functionality,*
- *Ability to configure the IPsec functionality,*
- *Ability to import X.509v3 certificates,*
- *Ability to enable, disable, determine and modify the behavior of all the security functions of the TOE identified in the VPN GW EP to the Administrator,*
- *Ability to configure thresholds for SSH rekeying.*
- *Ability to configure all security management functions identified in other sections of the VPN GW EP.*
    - *<u>No other capabilities</u>.*

ST Author Note

FMT_SMF.1 was refined in accordance with the NDcPP VPN GW EP v2.1 section 4.2.6

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

[29] Modified by TD0179

[30] Modified by TD0167

[31] Modified by TD0090

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."

EP Application Note

In order to prevent redundancy, an ST claiming conformance to this EP should not select "Ability to configure the cryptographic functionality" as defined in the NDcPP when completing FMT_SMF.1 since it is already mandated by this EP.

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).

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.

6.1.5.2 FPT_FLS.1/SelfTest Fail Secure

FPT_FLS.1.1

The TSF shall shutdown when the following types of failures occur: failure of the power-on self-tests, failure of integrity check of the TSF executable image, failure of noise source health tests.

EP Application Note

The failures relevant to this requirement are the FPT_TST_EXT.1.1 requirement in the NDcPP, and the FPT_TST_EXT.1.2 requirement specified in this EP.

6.1.5.3 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.

6.1.5.4 FPT_TST_EXT.1 TSF Testing

FPT_TST_EXT.1.1

The TSF shall run a suite of the following self-tests <u>during initial start-up (on power on)</u> to demonstrate the correct operation of the TSF: **digital signature verification of the TOE firmware, cryptographic known answer tests for all cryptographic algorithms.**

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.

6.1.5.5 FPT_TST_EXT.2 Extended: TSF Testing

FPT_TST_EXT.2.1

The TSF shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the TSF-provided cryptographic service specified in FCS_COP.1(2).

EP Application Note

The NDPP contains one element for this component, which simply requires a suite of self-tests to demonstrate correct operation of the TSF. This element is added to that component to comply with the EP.

6.1.5.6 FPT_TUD_EXT.1 Trusted Update

FPT_TUD_EXT.1.1 [32]

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

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.

FPT_TUD_EXT.1.2

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

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.

FPT_TUD_EXT.1.3

The TSF shall provide means to authenticate firmware/software updates to the TOE using a digital signature mechanism and no other functions prior to installing those updates.

ST Author Note

FPT_TUD_EXT.1.3 was refined in accordance with the NDcPP VPN GW EP v2.1 section 4.2.8

EP Application Note

The NDcPP provides an option of which method of verification the ST author wishes to specify. For compliance with this EP, a digital signature mechanism (one of those specified in FCS_COP.1(2) must be employed. Note that the ST author should include the other two elements of the 15 NDcPP FPT_TUD_EXT.1 in the ST without modification. This may also trigger the inclusion of the NDcPP's selection-based SFR

[32] Modified by TD0154

FPT_TUD_EXT.2 as specified in the NDcPP if "code signing for system software updates" is selected in FIA_X509_EXT.2 of the NDcPP.

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.

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).

6.1.5.7 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.

6.1.6 Class FPF: Packet Filtering

6.1.6.1 FPF_RUL_EXT.1 Packet Filtering

FPF_RUL_EXT.1.1

The TSF shall perform Packet Filtering on network packets processed by the TOE.

FPF_RUL_EXT.1.2

The TSF shall process the following network traffic protocols:

- Internet Protocol (IPv4)
- Internet Protocol version 6 (IPv6)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

and be capable of inspecting network packet header fields defined by the following RFCs to the extent mandated in the other elements of this SFR

- RFC 791 (IPv4)
- RFC 2460 (IPv6)
- RFC 793 (TCP)
- RFC 768 (UDP).

EP Application Note

This element identifies the protocols and references the protocol definitions that serve to define to what extent the network traffic can be interpreted by the TOE when importing (receiving network traffic or ingress) and exporting (sending – or forming to be sent - network traffic or egress).

While the protocol formatting specified in the RFCs is still used, many RFCs define behaviors which are no longer considered safe to follow. For example, RFC792 defined the "Redirect" ICMP type, which is not considered safe to honor when it might come from an adversary; the "source quench" message, which is insecure because its source cannot be validated.

FPF_RUL_EXT.1.3

The TSF shall allow the definition of Packet Filtering rules using the following network protocol fields:

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

EP Application Note

This element identifies the various attributes that are applicable when constructing rules to be enforced by this requirement – the applicable interface is a property of the TOE and the rest of the identified attributes are defined in the associated RFCs. Note that the Protocol is the IPv4 field (in IPv6 this field is called the "next header" that identifies the applicable protocol, such as TCP, UDP, ICMP, etc.. Also, 'Interface' identified above is the external port where the applicable network traffic was received or alternately will be sent.

FPF_RUL_EXT.1.4

The TSF shall allow the following operations to be associated with Packet Traffic Filtering rules: permit, discard, and log.

EP Application Note

This element defines the operations that can be associated with rules used to match network traffic.

FPF_RUL_EXT.1.5

The TSF shall allow the Packet Traffic Filtering rules to be assigned to each distinct network interface.

EP Application Note

This element identifies where rules can be assigned. Specifically, a conforming TOE must be able to assign filtering rules specific to each of its available and identifiable distinct network interfaces that handle layer 3 and 4 network traffic. Identifiable means the interface is unique and identifiable within the TOE, and does not necessarily require the interface to be visible from the network perspective (e.g., does not need to have an IP address assigned to it). A distinct network interface is one or more physical connections that share a common logical path into the TOE. For example, the TOE might have a small form-factor pluggable (SFP) port supporting SFP modules that expose a number of physical network ports, but since a common driver is used for all external ports they can be treated as a single distinct network interface.

Note that there could be a separate ruleset for each interface or alternately a shared ruleset that somehow associates rules with specific interfaces.

FPF_RUL_EXT.1.6

The TSF shall process the applicable Packet Filtering rules (as determined in accordance with FPF_RUL_EXT.1.5) in the following order: Administrator-defined.

EP Application Note

This element requires that an administrator is able to define the order in which configured filtering rules are processed for matches.

FPF_RUL_EXT.1.7

The TSF shall drop traffic if a matching rule is not identified.

EP Application Note

This element requires that the behavior is always to deny network traffic when no rules apply.

6.1.7 Class FTA: TOE Access

6.1.7.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

FTA_SSL_EXT.1.1

The TSF shall, for local interactive sessions,

- <u>terminate the session</u>

after a Security Administrator-specified time period of inactivity.

6.1.7.2 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.

6.1.7.3 FTA_SSL.4 User-initiated Termination

FTA_SSL.4.1

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

6.1.7.4 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.

6.1.8 Class FTP: Trusted Path/Channels

6.1.8.1 FTP_ITC_EXT.1 Inter-TSF Trusted Channel

FTP_ITC_EXT.1.1

The TSF shall use IPsec and <u>no other protocols</u> to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: audit server, VPN communications, **<u>NTP client, NTP server</u>** 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.

ST Author Note

FTP_ITC_EXT.1.1 has been refined in accordance with NDcPP VPN GW EP v2.1 section 4.2.9

EP Application Note

The NDcPP allows trusted channels other than IPsec to be available for communication with external IT entities but defers to this EP to specify VPN Gateway functionality. To be compliant with this EP, the selection is made such that the TOE must provide the IPsec protocol for its VPN Gateway functionality. Protection (by at least one of the listed protocols) is required at least for communications with the server that collects the audit information (per the NDcPP). For communication with any other authorized IT entity, the ST author makes the appropriate selections/assignments and includes the related requirements from Annex C corresponding to their selections.

FTP_ITC_EXT.1.2

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

FTP_ITC_EXT.1.3

The TSF shall initiate communication via the trusted channel for **NTP, Audit server (syslog)**.

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_EXT.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_EXT.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.

6.1.8.2 FTP_TRP.1 Trusted Path

FTP_TRP.1.1

The TSF shall be capable of using IPsec, SSH 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.*

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[33]: 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.

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.

[33] 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.

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[34].

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.

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.

Assurance Activity

[34] 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 shall generate network packets that cycle through all of the values for attributes, Type, Code, and Transport Layer Protocol, that are undefined by the RFC for each of the protocols, ICMPv4, ICMPv6, IPv4, and IPv6. For example, ICMPv4 has an eight-byte field for Type and an eight-byte field for Type has a Code associated with it, the number of RFC defined Codes varies based on the Type. The evaluator is required to construct packets that exercise each possible value not defined in the RFC (the defined values are already tested in FPF_RUL_EXT.1.10) of Type and Code (including all possible combinations) and target each distinct interface type to determine that the TOE handles these packets appropriately. Since none of these packets will match a rule, or belong to an allowed session the packets should be dropped. Since there are no requirements that the firewall audit a packet being dropped under these circumstances, the evaluator shall ensure the firewall does not allow these packets to flow through the TOE.

In addition to the undefined attribute testing required above, the evaluator shall perform intelligent fuzz testing of the remaining fields in the required protocol headers (excluding FTP). The intent of intelligent fuzzing is that a packet that is otherwise correctly constructed, such that it will be denied when the ruleset is applied, has random values inserted into each of the protocol header fields. The evaluator ensures a statistically significant sample size, which will vary depending on the protocol field length, is used and is justified in their report.

The evaluator should consult whatever diagnostics (e.g., logging, process status, interface errors) the TOE offers to determine if the TOE was adversely impacted by the processing of such packets.

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
- Packet Filtering
- TOE Access
- Trusted Path/Channels

7.1 Security Audit

7.1.1 Audit Generation

The TOE saves audit logs in a local audit log file store. Each audit log file is rotated at approximately 10MB in size, but due to the lag between the appending to the log and the rotation of the log, the size may grow larger than this. Each log will never grow larger than 20MB in size. On rotation of the log file, the relevant log file is deleted and a new log file is created. No log files other than the current log file are kept by the TOE.

Any individual packet filter rule (“firewall rule”) may be configured to log instances where that rule is applied including the name of the rule, the allow or deny status, the protocol, the source and destination ports, and the source and destination IP addresses.

The Voyager ESm provides four 100 Mbit Ethernet ports and one 1GBit Ethernet port. When the incoming traffic exceeds the bandwidth limitations of the ports, the TOE discards incoming packets beyond its capacity and logs a message to the audit log that packets have been dropped on the associated interface. Although the SW14 has more ports, the behavior is the same on the VoyagerSW14.

FAU_GEN.1

FAU_GEN.2

7.1.2 Audit Storage

The TOE manages a number of log files as follows:

- Audit log: Stores CLI commands entered by the user.
- System log: Stores general system log messages.
- IPSec log: Stores IPSec log messages.
- Firewall log: Stores packet filtering log messages.

Each log file is rotated at approximately 10MB in size but due to the lag between the appending to the log and the rotation of the log, the size may grow larger than this. Each log will never grow larger than 20MB in size. The log files cannot be modified, but may be read by an authorized security administrator. Log files are stored in volatile memory, and are not preserved between reboots - an authorized security administrator may issue a reboot command to delete all log files.

The TOE transmits audit logs to an RFC 5424 compliant syslog server over a trusted channel as described in [ST] Section 6.1.8.1.

The TOE log files store up to approximately 10 megabytes of data locally in each log file. When a log file reaches 10 megabytes in size, the TOE deletes the existing file and creates a new one. Note that due to the lag between the appending to the log and the rotation of the log, the size may grow larger than this. But each log will never grow larger than 20MB in size. While the TOE is configured to send data to an external RFC 5424-compliant syslog server, all log messages are sent to the syslog server over the trusted channel as described in [ST] Section 6.1.8.1

FAU_STG_EXT.1

7.2 Cryptographic Support

7.2.1 Cryptographic Key Generation

The TOE generates Elliptic Curve 256- and 384-bit ephemeral SSH session keys, 2048-bit DH ephemeral SSH session keys, Elliptic Curve 256- and 384-bit ephemeral IPsec SA keys, and 2048-bit DH ephemeral IPsec SA keys.

FFC keypairs (Diffie-Hellman, DSA) are generated with Extra Random Bits as specified in FIPS 186-4 Appendix B.2.1

EC keypairs (ECDSA, ECDH) are generated with Extra Random Bits as specified in FIPS pub 186-4 Appendix B.4.1.

DH keys for RSA based IKE key exchange and IKE peer authentication are generated according to FIPS 186-4 Appendix B.3.3 for probable primes.

When generating persistent keys, the TOE generates EC 256- or 384-bit keys over curves NIST P-256 or NIST P-384, or RSA-3072-bit keys for the IPsec public/private keypair. Choice of curve and associated key size is determined by the security administrator.

When generating its own public / private keypair for SSH, the TOE generates EC 256-bit or 384-bit keys over curves NIST P-256 or NIST P-384. Choice of curve and associated key size is determined by the security administrator.

The TOE complies with NIST SP 800-56A and for the following purposes:

- ECDH KeyGen
- FFC (Diffie-Helman except group14) KeyGen

The TSF forms DH Group14 exchanges, the implementation is SP800-56A compliant with the exception of the domain parameters (p and g from RFC 3526 and q=(p-1) / 2) instead of the domain parameters required by FIPS 186-4 Appendix B.1.

FCS_CKM.1(1)

FCS_CKM.1/IKE

The TOE utilizes FFC-based DH for non-EC SSH session keys and ECDH for EC-based SSH session keys. The TOE utilizes FFC-based DH for Group 14 IPsec key agreement and ECDH for other IPsec key agreement schemes.

FCS_CKM.2

The TOE does not store plaintext keys persistently. Plaintext ephemeral keys are generated for each unique IPsec or SSH session and are stored in RAM. Persistent keys that are in use are stored in plaintext in RAM.

Ephemeral keys associated with SSH sessions are zeroized when the associated session ends. Ephemeral keys associated with IPsec connections are zeroized when the associated SA is torn down. Pre-shared keys are zeroized at the command of an authorized security administrator. Persistent private keys are zeroized when the "crypto key zeroize" command is issued by an authorized security administrator. The Crypto Key Zeroize command may zeroize just one key, or all stored keys.

Cryptographic keys stored in RAM (ephemeral session keys, plaintext persistent keys, and pre-shared keys) are zeroized by being overwritten by zeroes. A read-verify is then performed to ensure that the contents have been changed to all zeroes. Keys stored in flash memory (private keys) are zeroized with a block-erase operation. A read-verify is performed on the partition to ensure all contents have been erased.

FCS_CKM.4

7.2.2 Cryptographic Operations

All ECDSA, RSA, and DSA operations performed by the TOE are compliant with FIPS 186-4.

Function	Certificate Number	SFR	Comments
AES - Encryption / Decryption	4131 implemented via Klas OpenSSL FIPS Object Module 2.0.9 and 4132 implemented via Klas Linux Kernel Crypto API Encryption performed as part of the IPsec IKEv1 Phase 2 / IKEv2 Child SA is done by the Linux Kernel Crypto API using certificate 4132. All other hashing (for example, as part of SSH, IKEv1 phase 1 / IKEv2 SA) is performed by the Klas OpenSSL FIPS object module 2.0.9 using certificate 4131	FCS_COP.1(1)	The TOE performs AES in the CBC mode with key sizes 128 or 256 bits, in GCM mode with key sizes 128 or 256 bits.
ECDH	1173 implemented via Klas OpenSSL FIPS Object Module 2.0.9	FCS_CKM.1(1) FCS_CKM.2	ECDH is used in EC SSH session key establishment. EC key pairs are generated for IPsec and

			SSH public/private authentication.
ECDSA	946 implemented via Klas OpenSSL FIPS Object Module 2.0.9	FCS_CKM.1(1) FCS_CKM.1/IKE FCS_COP.1(2)	The TOE performs ECDSA with NIST P-256 and NIST P-384 curves.
RSA	2249 implemented via Klas OpenSSL FIPS Object Module 2.0.9	FCS_CKM.1/IKE FCS_COP.1(2)	The TOE performs RSA with 3072-bit moduli.
DSA	1193 implemented via Klas OpenSSL FIPS Object Module 2.0.9	FCS_CKM.1(1)	
DH	1173 implemented via Klas OpenSSL FIPS Object Module 2.0.9	FCS_CKM.1(1) FCS_CKM.2.1	DH is used in non-EC SSH session key establishment, and certain IPsec key negotiation modes (DH Group 14).
ESP	N/A		
HMAC	2703 implemented via Klas OpenSSL FIPS Object Module 2.0.9 and 2704 implemented via Klas Linux Kernel Crypto API. Hashing performed as part of the IPsec IKEv1 Phase 2 / IKEv2 Child SA is done by the Linux Kernel Crypto API using certificate 2704. All other hashing (for example, as part of SSH, IKEv1 phase 1 / IKEv2 SA) is performed by the Klas OpenSSL FIPS object module 2.0.9 using certificate 2703	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-SHA1 uses a 64 bit block size. HMAC-SHA2-256 with a key size of 256 bits and an output digest size of 256 bits. HMAC-SHA2-256 uses a 512 bit block size. HMAC-SHA2-384 with a key size of 384 bits and an output digest size of 384 bits. HMAC-SHA2-384 uses a 1024 bit block size. HMAC-SHA2-512 with a key size of 512 bits and an output digest size of 512 bits. HMAC-SHA2-

			512 uses a 1024 bit block size. In IPsec, the TOE generates a 160-bit SHA1 digest. Only the first 96 bits are transmitted.
Cryptographic Hashing	3400 implemented via Klas OpenSSL FIPS Object Module 2.0.9 and 3401 implemented via Klas Linux Kernel Crytpo API Hashing performed as part of the IPsec IKEv1 Phase 2 / IKEv2 Child SA is done by the Linux Kernel Crypto API using certificate 3401. All other hashing (for example, as part of SSH, IKEv1 phase 1 / IKEv2 SA) is performed by the Klas OpenSSL FIPS object module 2.0.9 using certificate 3400	FCS_COP.1(3)	The TOE performs cryptographic hashing as follows: SHA1 - used in SSH HMAC. SHA-256 - used in SSH HMAC and IPsec HMAC, and in digital signature generation. SHA-384 - used in IPsec HMAC. Appropriate hash algorithms are also used for digital signature generation and verification, and the key-derivation functions of SSH and IPsec.
DRBG	1250 implemented via Klas OpenSSL FIPS Object Module 2.0.9	FCS_RBG_EXT.1	

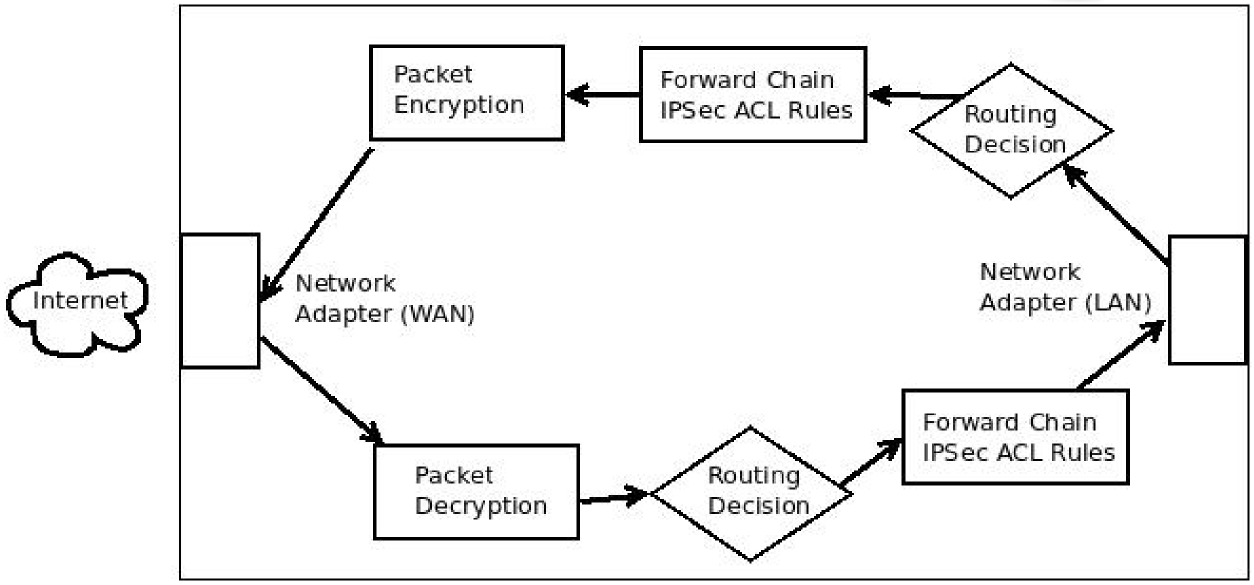
The TOE contains a deterministic random bit generator that complies with ISO 18031:2011 and utilizes the AES-based CTR_DRBG. The operation of this generator is described in complete detail in the non-public Entropy Assessment Report [ENT] that was provided to NIAP/IAD.

FCS_RBG_EXT.1

7.2.3 IPsec Protocol

The administrator creates an access-control list (ACL) specifying the traffic types and sources/destinations (peers) which are to be protected by IPsec encryption. The Access-Control List may also specify certain traffic types or peers to which packets are to be sent in the clear and may also specify the rules for dropping packets. The ACL performs all SPD functions: PROTECT, BYPASS, and DISCARD.

The above description applies to both transport and tunnel mode connections.



For an established session, the TOE receives an IPsec protected packet and attempts decryption. If decryption fails, the packet is dropped. The decrypted packet is routed - if the destination cannot be routed to, the packet is dropped. The routed packet is compared to the ACL rules to determine whether or not the packet is allowed.

The access list rules are associated with an access list number and this access list number is associated with the IPSec configuration. The rules are applied in a top-down order and the traffic is processed according to the first rule that is matched in the list.

The TOE implements the IPSec architecture as specified in RFC 4301. The administrator configures the IPsec service with the list of allowed and excluded endpoints (peers). This is configured using SPD access list rules.

IPSec is configured on the TOE by applying the appropriate IPSec configuration to an interface. IKE negotiation is triggered when traffic matching a ‘protect’ SPD access list rule arrives at the ToE. IKE negotiation will also get triggered if the peer end initiates the connection. If the SA negotiation is successful, the traffic will get encrypted/decrypted and routed accordingly. If the SA fails, the traffic will get dropped immediately.

If an SA already exists to protect specific traffic configured in a SPD access list rule, then any traffic arriving from these endpoints will get encrypted or decrypted by the TOE before being routed to their destination.

If an SA doesn’t exist for traffic arriving at the TOE, then that traffic will be processed according to the SPD access list rules. If there is a bypass rule for that traffic, then that traffic can be routed through the device unencrypted. If there is no SPD rule for the traffic, then the traffic is dropped using a default deny all rule.

If traffic arrives during SA establishment, that traffic will get dropped until the SA is established.

FCS_IPSEC_EXT.1.1

The TOE contains a nominal, final “Deny All” firewall rule. If a packet is not matched to an explicit administrator-defined firewall rule, the packet is dropped.

FCS_IPSEC_EXT.1.2

The TOE will establish IPsec connections with peers in either tunnel or transport mode.

FCS_IPSEC_EXT.1.3

The TOE supports AES-CBC-128 and AES-CBC-256 for ESP when it is configured for IKEv1. The ToE supports AES-CBC-128, AES-CBC-256, AES-GCM-128, and AES-GCM-256 for ESP when it is configured for IKEv2. The TOE supports the use of HMAC-SHA-256 and HMAC-SHA-384 for message integrity.

FCS_IPSEC_EXT.1.4

In IKEv1, the TOE only supports main mode. The TOE does not support aggressive mode for IKEv1 Phase 1 Exchanges. The TOE supports NAT Traversal as specified in RFC 5996 section 2.23 and RFC 4868 for hash functions.

FCS_IPSEC_EXT.1.5

The TOE supports AES-CBC-128 and AES-CBC-256 for both IKEv1 and IKEv2. The TOE also supports AES-GCM-128 and AES-GCM-256 for IKEv2.

FCS_IPSEC_EXT.1.6

For both IKEv1 and IKEv2 IPsec connections, the TOE supports negotiation of Phase 1 / IKE_SA lifetimes in elapsed seconds. The lifetime of an IKE phase negotiation can be at most 86,400 seconds (24 hours). Phase 2 / Child_SA lifetimes may be negotiated in terms of elapsed time or number of bytes of data transferred through the connection. For elapsed-time lifetimes, the TOE negotiates lifetimes of up to 720 hours.

In the Phase 2 / ESP phase only, the TOE allows the administrator to configure a data-transfer lifetime to allow between 2560 and 2147483647 bytes to be transferred before the tunnel expires.

For elapsed-time lifetimes, the TOE makes use of the NTP-corrected real-time clock or system real-time clock to determine the exact starting time of an SA and calculate its end time based on the currently configured lifetime limitation. For data-transfer lifetimes, the TOE IPsec implementation counts the number of bytes sent through a given SA.

For IKEv1 and IKEv2, the TOE will re-key the connection 30 seconds before the configured lifetimes elapse. If volume-based re-keys are configured, expiry will occur when the number of bytes configured has been reached.

FCS_IPSEC_EXT.1.7

FCS_IPSEC_EXT.1.8

The TOE shall generate the secret value x used in the DH key exchange using the random bit generator specified in FCS_RBG_EXT.1 and having a length of at least: 224 bits for Group 14 (2048 bits), 256 bits for Group 19 (256 bits), 384 bits for Group 20 (384 bits) and 256 bits for Group 24 (256 bits).

FCS_IPSEC_EXT.1.9

The TOE generates nonces used in IKEv1 and IKEv2 of at least 128 bites in size (half the output size of the negotiated pseudorandom function hash). The TOE supports PRF hash functions SHA256 and SHA384. The TOE generates nonces of 256 bits.

FCS_IPSEC_EXT.1.10

The TOE supports DH group 14, 19, 20, and 24 Diffie-Hellman key exchange. The specific groups supported may be configured by the security administrator, and negotiation of which DH group to use for key exchange is performed by choosing the strongest DH group supported by both the TOE and the peer in their handshake messages. This behavior is consistent with RFC5996 section 2.7, "Cryptographic Algorithm Negotiation."

FCS_IPSEC_EXT.1.11

The TOE ensures that the strength of the symmetric algorithm used to protect the Phase 1 / IKE_SA / IKE phase of an IPsec connection is greater than or equal to the strength of the symmetric algorithm used to protect the Phase 2 / Child_Sa / ESP phase of an IPsec connection.

FCS_IPSEC_EXT.1.12

The TOE supports IPsec authentication using ECDSA and RSA peer authentication protocols using X.509v3 certificates in accordance with RFC 4945.

The TOE cannot generate pre-shared keys, but an administrator-provided ASCII or bit-based pre-shared key may be entered and used for IPsec authentication.

FCS_IPSEC_EXT.1.13

The TOE allows the administrator to configure an expected subject name. If a peer certificate does not match this expected DN, the connection is rejected.

FCS_IPSEC_EXT.1.14

7.2.4 Random Bit Generation

The TOE produces random bits in accordance with the NIST recommendations. The TOE utilizes a combination of multiple software and hardware sources to generate entropy.

FCS_RBG_EXT.1

7.2.5 SSH Server Protocol

The TOEs SSH implementation is conformant to RFCs 4251, 4252, 4253, 4254, 5656, and 6668. The TOE provides SSH services for remote administration.

FCS_SSHS_EXT.1.1

The TOE implements both password-based and public key based authentication mechanisms. The TOE supports ECDSA-based public key authentication using NISTp256 or p384 curves.

FCS_SSHS_EXT.1.2

The TOE SSH implementation contains a counter for received packet size. The TOE implements a counter for each SSH session. As a packet is being received is, this counter increments for each byte received. If the counter exceeds 35,000 bytes in a single packet (inclusive of padding data, header data, etc.) the packet is dropped.

FCS_SSHS_EXT.1.3

The TOE supports AES-CBC-128 and AES-CBC-256 encryption algorithms to protect the content of the SSH session. No optional characteristics are specified.

FCS_SSHS_EXT.1.4

The TOE supports SSH public key authentication using ECDSA-SHA2-NISTP256 and ECDSA-SHA2-NISTP384 keypairs. All other public key algorithms are rejected. If an administrator does not wish to configure ECDSA public key authentication, password-based authentication is also supported.

FCS_SSHS_EXT.1.5

The TOE supports the use of HMAC-SHA1, HMAC-SHA2-256, and HMAC-SHA2-512 as the only integrity algorithms for SSH sessions.

FCS_SSHS_EXT.1.6

The TOE allows only the following key exchange methods: DH-Group14 with SHA1, ECDH over NIST p256 with SHA2, and ECDH over NIST p384 with SHA2.

FCS_SSHS_EXT.1.7

The TOE implements a counter to keep track of the number of packets transmitted through an SSH session. The TOE ensures that the same session keys are used for a threshold of no longer than one hour and no more than 1 GB of transmitted data.

FCS_SSHS_EXT.1.8

7.3 Identification and Authentication

7.3.1 Authentication and Failure Handling

FIA_AFL.1.1

The TOE allows an administrator to configure the number of successive failed authentication attempts that are permissible before account locking occurs. Each time a user attempts authentication, a counter is implemented which tracks the number of successive failed authentication attempts. When a user fails to authenticate a number of times equal to the configured limit, the TOE locks the claimed user identity (if it exists). The account remains locked until such time as a local administrator manually unlocks the account.

While the account is locked, the TOE will refuse all authentication attempts claiming the locked account as the user identity.

FIA_AFL.1.2

7.3.2 Password Management

The TOE allows administrators to set passwords of between 15 and 128 characters. Minimum password length can be configured by the administrator, and the administrator can configure the TOE to require passwords of at least 15 characters.

Configured passwords can use special characters - these characters include the following: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)”.

FIA_PMG_EXT.1.1

7.3.3 Pre-Shared Key Management

The TOE allows administrators to configure text-based and bit-based pre-shared keys for IPsec. Pre-Shared keys for IPsec may be 22 characters in length, and must be between 1 and 128 characters long. Supported characters are the complete set of ASCII upper and lower case letters, the numerals 0-9, and the following special characters: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”.

PSK’s are always stored in Base64 format in the underlying file system, but may be displayed in cleartext, type7, or Base64 in the running configuration file with administrator configuration. For use by the IPsec service, the TOE utilizes the stored base64 encoded key.

The TOE may accept pre-shared keys generated by an administrator but does not generate its own.

Text-based pre-shared keys are converted to bitstrings by being conditioned by SHA-256 or SHA-384, as specified by the administrator.

FIA_PSK_EXT.1

7.3.4 User Identification and Authentication

The TOE supports local authentication via the local serial console. Authentication is performed by providing the username and password. Successful authentication will be indicated by presenting the console prompt; unsuccessful authentication is indicated by an error message and a prompt to re-authenticate.

The TOE supports remote authentication via SSH. Password based authentication is performed by providing the username and password. Successful authentication will be indicated by presenting the console prompt; unsuccessful authentication is indicated by an error message and a prompt to re-authenticate. Public-Key based authentication is performed by configuring an SSH client to send the correct public key to the TOE for comparison with the stored authentication data. Successful authentication will be indicated by presenting the console prompt; unsuccessful authentication is indicated by an error message and a prompt to provide a password for the claimed user identity. If the password authentication fails, the user will receive an error message and the SSH session will fail.

FIA_UIA_EXT.1

7.3.5 Password-based Authentication Mechanism

The TOE provides a local authentication mechanism that is password protected for administrative functions. This is the local serial console, which requires authentication as an authorized security administrator before administration of the TOE takes place.

FIA_UAU_EXT.2.1

7.3.6 Protected Authentication Feedback

The TOE does not echo back authentication data during authentication attempts. Password prompts provide neither the authentication data nor obfuscated data (such as asterisks).

FIA_UAU.7.1

7.3.7 X.509 Certificate Validation

The TSF validates 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 validates 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 can validate the revocation status of the certificate using a Certificate Revocation List (CRL) as specified in RFC 5759.

The TOE validates a certificate that is installed on the TOE (such as its own certificate, Root CA certificates, and Intermediate CA certificates) at import time (when the certificate is installed onto the device) and also on use (ie, when an IPSec session is being established). The TOE validates peer certificates and peer CA certificates on use (ie, when an IPSec session is being established). The TOE will reject expired certificates, certificates that are not valid (certificates which have failed one or more validity checks as described in FIA_X509_EXT.1), and certificates that have been revoked via a CRL.

The TOE validates peer certificates using the distinguished name (DN) whenever the peer attempts to connect to the TOE. If the DN configured on the TOE does not match the DN contained in the peer certificate, the connection will fail to establish.

The certificate validation process includes a number of steps:

- The TOE checks to ensure that the certificate has been signed.
- Inspection of certificate extensions to determine whether or not the CA flag is present and True, present and False, or not present.
- Path validation of the certificate by checking for the presence and validity of the next certificate in the trust chain (i.e., the certificate that signed the certificate in question). This step is iterated as many times as needed to get to a root CA, and all certificates between the root CA and the certificate in question are part of the "trust chain".

If the certificate validation path does not terminate with a trusted root CA that is itself valid, validation of the certificate fails. If any part of the validation path fails validation, validation of the certificate fails. The TOE may use CRLs distributed via web servers. CRLs contained in the peer certificates are checked when the peer certificate is presented for validation. CRL's contained within the TOE certificates are checked when the certificate is imported onto the device and also on use (ie, during IPSec establishment). All CRL's must be signed by a CA certificate with the CRLsign purpose or the CRL will not be processed by the TOE. If the connection to the CRL Distribution Point is down or the CRLDP cannot be reached, the TOE will reject the certificate. The administrator must follow the instructions in Guidance for this.

The administrator can define what certificates are used by applying the relevant imported TOE and CA certificates to the IPSec configuration.

FIA_X509_EXT.1

7.3.8 X.509 Certificate Authentication

The TOE allows multiple certificate trustpoints to be configured. CA certificates, Intermediate CA certificates, and end entity certificates are associated with the trustpoint. A trustpoint is then applied to the IPSec configuration. On IPSec establishment, the associated trustpoint is then validated. Instructions for configuring the TOE to use the correct trustpoint are provided in the administrative guidance.

When the TOE cannot establish a connection to determine the validity of a certificate, the TOE rejects the connection. Specific instructions on how to configure this functionality is provided in the administrative guidance.

FIA_X509_EXT.2

7.3.9 X.509 Certificate Requests

In addition to the distinguished name fields, an administrator may optionally enter the device serial number and associated IP address in the CSR. This data is encoded as part of the subject name. The administrator may also create a subject alternative name that includes the IP address or DNS name of the device.

FIA_X509_EXT.3

7.3.10 X.509 Certificate Identity

The TOE provides the ability for the administrator to configure a peer subject name as the expected DN for X.509v3 certificate validation. If the peer certificate presented for authentication does not match the configured subject name, the TOE will deny the establishment of a SA.

The TOE stores certificates in the CLI configuration, which is in turn stored in RAM after bootup. To save the configuration, including imported certificates, the running configuration is saved to a startup configuration file in the TOE flash memory.

FIA_X509_EXT.4

7.4 Security Management

7.4.1 Management of Security Functions Behaviour

The TOE allows authorized and authenticated security administrators to perform the following functions, which are restricted only to security administrators. Non-security administrators are prevented from performing any of the following functions.

Administrator-configurable functions:

- Initiate manual updates of the firmware image.
- Change the behavior of the TSF with regard to transmission of audit data to external syslog servers.
- Enable, disable, start or stop any of the following services:
  - DHCP server
  - SNMP server
  - TFTP server
  - VoIP and SIP services
  - OSPF and RIP
  - 802.1x and RADIUS
  - CDP
  - NTP server and client
  - SSH server
  - DNS server
  - Multicast PIM
  - IGMP snooping
  - IPSec
  - Remote syslog

FMT_MOF.1

FMT_MOF.1.1(1)/TrustedUpdate

FMT_MOF.1.1(1)/Audit

FMT_MOF.1.1(1)/AdminAct

FMT_MOF.1.1(2)/AdminAct

7.4.2 Management of TSF Data

The TOE prevents non-security administrators from modifying any TSF element or security function. The only interfaces available to an unauthenticated user are the login prompts to the TOE. Only authorized security administrators may authenticate to the TOE and interact with the TSF data.

FMT_MTD.1

The TOE prevents all non-security-administrators from modifying, deleting, generating, or importing cryptographic keys or certificates. No ability to perform these functions is present prior to authentication as a security administrator.

FMT_MTD.1.1/AdminAct

7.4.3 Specification of Management Functions

The TOE allows the authorized security administrator to:

- Perform administrative activities locally and remotely
- Configure the pre-login access banner in accordance with FTA_TAB
- Configure an inactivity timeout period
- Perform updates to the TOE firmware
- Configure the operation of the cryptographic functions.
- Configure IPsec functionality
- Import X.509v3 certificates
- Enable, Disable, View, and Edit the behavior of the security functions
- Configure the security management functions of the TOE
- Configure firewall / packet filtering rules

FMT_SMF.1

7.4.4 Restrictions on Security Roles

The TSF allows one security administrator account to be created on the TOE. All administrator accounts are security administrators.

FMT_SMR.2

7.5 Protection of the TSF

7.5.1 Protection of Administrator Passwords

Local and SSH password authentication data is stored as an MD5 hash in the underlying operating system and running configuration. Public keys used for SSH authentication to the TOE are stored in the underlying operating system and in the running configuration in plaintext form. Non-security administrators are not permitted to view the public keys.

FPT_APW_EXT.1

7.5.2 Fail Secure

The TOE performs self-testing on startup. If the entropy or cryptographic self-tests fail, the TOE immediately reboots and will not become operational until these tests all pass. The TOE does not allow traffic to flow before becoming fully operational, so a reboot due to failure of the self-tests does not allow traffic to flow. If the firmware image integrity check fails, the TOE will bootup but all cryptographic functionality will be disabled and an error message will be logged in the audit log.

Noise Source health testing is implemented as part of the start-up self testing. If the noise source fails health testing, the TOE reboots immediately.

FPT_FLS.1

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

The TOE stores private portions of keypairs in individual micropartitions of the local flash storage. Pre-shared keys are read from the configuration file stored in the underlying filesystem at runtime and are stored in RAM. There is no standard interface for reading the stored private keys or pre-shared keys. Pre-shared keys and private keys may be destroyed (zeroized) or replaced (zeroized and overwritten with new data), but not read.

FPT_SKP_EXT.1

7.5.4 TSF Testing

The TOE performs an integrity check of the installed firmware by comparing the 256-bit ECDSA digital signature of the complete firmware image during the bootup process, before any configuration is loaded or any interfaces are enabled. If signature verification fails, the TOE fails the self-testing process. After verifying the image, the TOE performs cryptographic known-answer tests on all cryptographic algorithms. The correct operation of the TOE is ensured by verifying that the installed firmware image is unmodified and that the cryptographic modules are performing as expected.

FPT_TST_EXT.1, FPT_TST_EXT.2

7.5.5 Trusted Update

The TOE allows security administrators to initiate manual updates by copying the update candidate to the TOE using non-TOE provided methods (SCP or TFTP). The candidate updates are first obtained from the vendor support channel. The update candidate is stored in local flash memory, and the administrator is instructed to verify the digital signature of the update candidate manually to verify that the digital signature matches the expected result. If signature verification fails, the administrator is notified that the firmware update cannot continue, and the update candidate is automatically deleted by the TOE. If signature verification succeeds, the administrator is instructed that the update may continue, and the TOE proceeds to apply the update.

FPT_TUD_EXT.1

7.5.6 Reliable Time Stamps

The TOE updates the local real-time clock based on an NTP server configured by the administrator. The time is used for many functions (a pseudorandom number for a non-security relevant function, X.509v3 certificate validation, etc.), as well as the auditing and timestamping process. The time interval (counter) is used for IPsec, session timers, audit log, and PKI.

The hardware real-time clock has a drift estimate of less than 1.5 minutes per month, which does not represent a security concern in the context of any of the stated security functions. In combination with a configured NTP server, this time value is considered reliable.

FPT_STM.1

7.6 Packet Filtering

7.6.1 Packet Filtering

During startup, the TOE disables all physical interfaces and does not allow traffic to flow. The TOE enables the physical ports and logical packet processing function only after successfully completing the self-tests and initializing the firewall / ACL rulesets.

Firewall packet processing is performed on an individual VLAN. The TOE allows the administrator to create one or more VLANs spanning one, many, or all physical interfaces. Packet filtering is bundled with the routing and packet processing function, a process that runs within the overall KlasOS. If this process hangs, fails, or is terminated all packet flow will cease. The TOE performs resource management to mitigate potential resource exhaustion failures, and if such a failure should occur (such as all memory being allocated), the TOE ceases to process packets until sufficient resources are available.

FPF_RUL_EXT.1.1

The TOE supports the following protocols for packet filtering:

- RFC 791 (IPv4)
- RFC 2460 (IPv6)
- RFC 793 (TCP)
- RFC 768 (UDP)

Applicability of the TOE to the above-claimed standards is demonstrated via protocol compliance and interoperability between other compliant devices.

FPF_RUL_EXT.1.2

The TOE packet filter allows the administrator to configure rules based on the following attributes:

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

Any rule configured by the security administrator can permit or deny the specified traffic, and log or not log the action of the rule.

The TOE packet filter applies to any or multiple VLAN interface(s). The administrator determines which interfaces are controlled by which ACLs.

FPF_RUL_EXT.1.5

The TOE processes incoming packets by parsing the packet data and comparing this to the configured access control lists. The administrator may configure any number of access control lists in sequence. If the packet matches a deny rule, it is discarded. If it matches an allow rule, it is forwarded according to the ACL ruleset (i.e., “encrypt the traffic to an IPsec peer” or “pass the packet in cleartext”). If the packet reaches the end of the ruleset without being matched, the TOE applies a final default-deny rule to drop the packet.

Established sessions are not treated any differently than new sessions.

FPF_RUL_EXT.1.6

The TOE applies a final, default-deny rule that matches all packets that are otherwise unmatched and discards them.

FPF_RUL_EXT.1.7

7.7 TOE Access

7.7.1 TSF-initiated Session Locking

The TOE terminates local administrative sessions when the session inactivity timer expires.

FTA_SSL_EXT.1

7.7.2 TSF-initiated Termination

The TOE terminates remote administrative sessions when the session inactivity timer expires.

The TOE terminates inactive VPN SAs via dead peer detection. When the TOE determines that a peer is inactive, a timer is started. When the timer meets a hard-coded inactivity period, the SA is terminated and a new SA must be established. IKEv1 timeout value is 150 seconds; IKEv2 is 180 seconds.

FTA_SSL.3

7.7.3 User-initiated Termination

The TOE allows the administrator to terminate their own session for both remote and local sessions.

FTA_SSL.4

7.7.4 Default TOE Access Banners

The TOE provides a pre-authentication access banner that is configurable by the administrator and is displayed in local and remote authentication sessions prior to allowing the user to authenticate.

FTA_TAB.1

7.8 Trusted Path/Channels

7.8.1 Inter-TSF Trusted Channel

The TOE utilizes IPsec to provide a trusted communication channel between itself and authorized IT entities that is logically distinct from other communication channels and provides assured identification of the endpoints. The Trusted Channel prevents detection or disclosure of the channel data.

The TOE uses the trusted channel to protect communication between itself and an NTP server or NTP client configured by the administrator and the audit / syslog server.

The TOE shall initiate communication via the trusted channel for NTP and syslog.

FTP_ITC_EXT.1

7.8.2 Trusted Path

The TOE provides IPsec and SSH as the remote administration mechanisms between itself and authorized administrators to provide assured identification of the endpoints and protection of the channel data against detection or disclosure.

FTP_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
CSfC	Commercial Solutions for Classified
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

Table 9: TOE Abbreviations and Acronyms

Abbreviations/ Acronyms	Description
HMAC	(Keyed-) Hash Message Authenticating Code
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

Table 9: TOE Abbreviations and Acronyms

Abbreviations/ Acronyms	Description
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
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
TACACS	Terminal Access Controller Access Control System
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
CAC	Common Access Card
CAP	Composed Assurance Package
CC	Common Criteria
CCRA	Arrangement on the Recognition of Common Criteria Certificates in the field of IT Security
DAC	Discretionary Access Control
DOD	Department of Defense
DOD	See DOD
EAL	Evaluation Assurance Level
IT	Information Technology
OSP	Organizational Security Policy
PP	Protection Profile
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SFP	Security Function Policy
SPD	Security Policy Database
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface

9. References

Table 11: TOE Guidance Documentation

Reference	Description	Version	Date
[ESMHW]	VOYAGERESm Hardware Reference Guide	1.0	November 2014
[SW14HM]	VOYAGERSW14 Hardware Reference Guide	2.1	August 2014
[SCG]	KlasOS Software Configuration Guide	4.3.2	
[AQ]	Klas Voyager Assurance Questionnaire (625-8315)	1.2	August 2017
[ENT]	VoyagerESM Entropy Design and Analysis	1.6	February 2017
[AGD]	Klas Voyager Common Criteria Operational User Guidance	1.7	September 2017

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
[PP]	Collaborative Protection Profile for Network Devices	1.0	February 27, 2015
[SD]	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP	1.0	February 2015
[EP]	Collaborative Protection Profile for Network Devices (NDcPP) Extended Package for VPN Gateways	2.1	December 1, 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 and P-384, 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 and P-384, 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) 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) 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 pseudo randomly 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 pseudo randomly 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.