QuintessenceLabs

Trusted Security Foundation 400, Version 3.2

Security Target

Version 1.0

2025-03-05

Prepared for:

Unit 11, 18 Brindabella Circuit
Brindabella Business Park
Canberra Airport, ACT 2609
AUSTRALIA

Prepared by:

Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive
Columbia, Maryland 21046

Table of Contents

List of Figures and Tables

1 Security Target Introduction

This section introduces the Target of Evaluation (TOE) and provides the Security Target (ST) and TOE references, TOE overview, and TOE description. It also contains the ST and TOE conformance claims, ST conventions, glossary, and list of abbreviations.

This ST includes the following additional sections:

- Security Problem Definition (Section 2)—describes the threats and assumptions that define the security problem to be addressed by the TOE and its environment
- Security Objectives (Section 3)—describes the security objectives for the TOE and its operational environment necessary to counter the threats and satisfy the assumptions that define the security problem
- IT Security Requirements (Section 4)—specifies the security functional requirements (SFRs) and security assurance requirements (SARs) to be met by the TOE
- TOE Summary Specification (Section 5)—describes the security functions of the TOE and how they satisfy the SFRs
- Protection Profile Claims (Section 6)—provides rationale supporting the claims for conformance of the ST and the TOE to [cPPND]
- Rationale (Section 7)—provides mappings and rationale for the security problem definition, security objectives, security requirements, and security functions to justify their completeness, consistency, and suitability.

1.1 Security Target, Target of Evaluation, and Common Criteria Identification

ST Title: QuintessenceLabs Trusted Security Foundation 400, Version 3.2 Security Target

ST Version: Version 1.0

ST Date: 2025-03-05

TOE Identification: QuintessenceLabs Trusted Security Foundation 400, Version 3.2

TOE Developer: QuintessenceLabs

Evaluation Sponsor: QuintessenceLabs

CC Identification: Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, April 2017.

1.2 Conformance Claims

This ST and the TOE it describes are conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, Version 3.1 Revision 5, April 2017
  - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Version 3.1 Revision 5, April 2017
  - Part 3 Conformant.

This ST and the TOE it describes are conformant to the following Protection Profile:

- *collaborative Protection Profile for Network Devices*, Version 2.2e, 23 March 2020 [cPPND], including the following optional and selection-based SFRs: FAU_STG.1, FCS_HTTPS_EXT.1; FCS_NTP_EXT.1; FCS_SSHC_EXT.1; FCS_SSHS_EXT.1; FCS_TLSC_EXT.1; FCS_TLSC_EXT.2; FCS_TLSS_EXT.1; FCS_TLSS_EXT.2; FIA_X509_EXT.1/Rev; FIA_X509_EXT.2; FIA_X509_EXT.3; and FMT_MTD.1/CryptoKeys.

The following NIAP Technical Decisions are applicable to the claimed Protection Profile:

- TD0527 – Updates to Certificate Revocation Testing (FIA_X509_EXT.1)
  - This TD is applicable to the TOE but relates solely to evaluation activities so it does not affect the ST.
- TD0528 – NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4
  - This TD is applicable to the TOE but relates solely to evaluation activities so it does not affect the ST.
- TD0536 – NIT Technical Decision for Update Verification Inconsistency
  - This TD is applicable to the TOE but relates solely to evaluation activities so it does not affect the ST.
- TD0537 – NIT Technical Decision for Incorrect reference to FCS_TLSC_EXT.2.3
  - This TD is applicable to the TOE but relates solely to an application note and therefore it does not affect the ST.
- TD0546 – NIT Technical Decision for DTLS - clarification of Application Note 63
  - This TD is not applicable to the TOE because DTLS is not used.
- TD0547 – NIT Technical Decision for Clarification on developer disclosure of AVA_VAN
  - This TD is applicable to the TOE.
- TD0555 – NIT Technical Decision for RFC Reference incorrect in TLSS Test
  - This TD is applicable to the TOE but relates solely to evaluation activities so it does not affect the ST.
- TD0556 – NIT Technical Decision for RFC 5077 question
  - This TD is applicable to the TOE but relates solely to evaluation activities so it does not affect the ST.
- TD0563 – NiT Technical Decision for Clarification of audit date information
  - This TD is applicable to the TOE.
- TD0564 – NiT Technical Decision for Vulnerability Analysis Search Criteria
  - This TD is applicable to the TOE but relates solely to evaluation activities so it does not affect the ST.
- TD0569 – NIT Technical Decision for Session ID Usage Conflict in FCS_DTLSS_EXT.1.7
  - This TD is applicable to the TOE but relates to application notes and evaluation activities so it does not affect the ST.
- TD0570 – NiT Technical Decision for Clarification about FIA_AFL.1
  - This TD is a clarification to the requirement and therefore is generally applicable to the TOE but does not change any of the requirements.

- TD0571 – NiT Technical Decision for Guidance on how to handle FIA_AFL.1
  - This TD is a clarification to the requirement and therefore is generally applicable to the TOE but does not change any of the requirements.
- TD0572 – NiT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers
  - This TD is a clarification to the FCS_TLSC_EXT and FTP_ITC.1 requirements and therefore is generally applicable to the TOE but does not change any of the requirements.
- TD0580 – NiT Technical Decision for clarification about use of DH14 in NDcPPv2.2e.
  - This TD is a clarification to the DH14 requirement and modifies the selections in the SFR. Therefore it is applicable to the ST.
- TD0581– NiT Technical Decision for Elliptic curve-based key establishment and NIST SP 800-56Arev3
  - This TD adds a selection for Elliptic curve-based key establishment based on NIST SP 800-56Arev3 and is therefore applicable to the ST as it has been selected.
- TD0591– NIT Technical Decision for Virtual TOEs and hypervisors
  - This TD modifies an assumption and therefore is applicable to the TOE.
- TD0592– NIT Technical Decision for Local Storage of Audit Records
  - This TD is applicable to the TOE, though it has no material effect on the ST.
- TD0631 – NIT Technical Decision for Clarification of public key authentication for SSH Server
  - This TD modifies FCS_SSHS_EXT.1 and FMT_SMF.1, and is applicable to the TOE.
- TD0632 – NIT Technical Decision for Consistency with Time Data for vNDs
  - This TD adds a selection but the ST does not make this selection. Therefore it is not applicable.
- TD0635 – NIT Technical Decision for TLS Server and Key Agreement Parameters
  - This TD is applicable to the TOE.
- TD0636 – NIT Technical Decision for Clarification of Public Key User Authentication for SSH
  - This TD modifies FCS_SSHC_EXT.1 and some of the EAs, and is applicable to the TOE.
- TD0638 – NIT Technical Decision for Key Pair Generation for Authentication
  - This TD is applicable to the TOE.
- TD0639 – NIT Technical Decision for Clarification for NTP MAC Keys
  - This TD is applicable to the TOE.
- TD0670 – NIT Technical Decision for Mutual and Non-Mutual Auth TLSC Testing
  - The TD clarifies/adds a test for FCS_TLSC_EXT.1.2 and therefore the TD is applicable to the TOE.
- TD0738 – NIT Technical Decision for Link to Allowed-With List
  - This TD moves the storage location of the allowed-with list to GitHub. It is not applicable to the TOE or to this ST.
- TD0790 – NIT Technical Decision: Clarification Required for testing IPv6

  - This TD modifies a test for FCS_TLSC_EXT.1.2 and therefore the TD is applicable to the TOE.
- TD0792 – NIT Technical Decision: FIA_PMG_EXT.1 - TSS EA not in line with SFR
  - This TD is applicable to the TOE.
- TD0800 – Updated NIT Technical Decision for IPsec IKE/SA Lifetimes Tolerance
  - IPsec is not used in the TOE and therefore this TD is not applicable to the TOE.

1.3 Conventions

The following conventions are used in this document:

- Security Functional Requirements—Part 1 of the CC defines the approved set of operations that may be applied to functional requirements: iteration; selection; assignment; and refinement.
  - Iteration—allows a component to be used more than once with varying operations. In this ST, the only iterated requirements are those reproduced from [cPPND], which uses descriptive strings to distinguish iterations of a requirement. For example, iterations of FCS_COP.1 are identified FCS_COP.1/DataEncryption, FCS_COP.1/SigGen, FCS_COP.1/Hash, and FCS_COP.1/KeyedHash.
  - Selection—allows the specification of one or more elements from a list. Selections completed in the ST are indicated using bold italics and are enclosed by brackets (e.g., [***selection***]).
  - Assignment—allows the specification of an identified parameter. Assignments completed in the ST are indicated using bold text and are enclosed by brackets (e.g., [**assignment**]). An assignment within a selection is identified in bold italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
  - Refinement—allows the addition of details. Refinements made in the ST of requirements drawn from [cPPND] would be indicated using bold for additions and strike-through for deletions (e.g., "… ~~some~~ **all** objects).
- Other sections of the ST—other sections of the ST use bolding and/or different fonts (such as `Courier`) to highlight text of special interest, such as captions, commands, or filenames specific to the TOE.
- These conventions are only applied to operations performed by the ST author.

1.4 Abbreviations and Acronyms

Table 1: Abbreviations and Acronyms

Abbreviation	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CSR	Certificate Signing Request
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm

Abbreviation	Definition
HMAC	Hash-based Message Authentication Code
HSM	Hardware Security Module
KMIP	Key Management Services
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentication Code
NTP	Network Time Protocol
QLabs	QuintessenceLabs
QRNG	Quantum Random Number Generator
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SKID	Secure Key ID
SSH	Secure Shell
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
	Trusted Security Foundation (the TOE)
USB	Universal Serial Bus

1.5 TOE Overview

The TOE is QuintessenceLabs' (QLabs) Trusted Security Foundation (TSF[1]) 400, Version 3.2. QLabs Trusted Security Foundation is a network-based appliance providing centralized and vendor-neutral key and policy management capabilities that seamlessly integrate into any organization's infrastructure. Trusted Security Foundation manages keys over their full life cycle in accordance with NIST SP 800-57 and KMIP protocol standards. The Trusted Security Foundation TOE is being evaluated as a network device and therefore the key and policy management capabilities were not evaluated.

The evaluation covers the secure communication channels for key management services (KMIP); database transactions and replication; and with LDAP, SMTP, backup and external syslog servers. Additionally, the evaluation covers all TOE functionality supporting the claims in the collaborative Protection Profile for Network Devices [cPPND]. The security functionality specified in [cPPND] includes protection of communications between the TOE and external IT entities as mentioned above, identification and authentication of administrators, auditing of security-relevant events, ability to verify the source and integrity of updates to the TOE, protection of TSF data, and use of NIST-validated cryptographic mechanisms.

1.6 TOE Description

The TOE is a standalone network device with a hardware security module (HSM) providing capabilities for key management and policy enforcement.

For the purpose of this evaluation, the TOE is treated as a network device offering NIST validated cryptographic functions, security auditing, secure administration, trusted updates, self-tests, and secure connections to other servers (e.g., to export audit records), protected using HTTPS/TLS and SSH.

Cryptographic functionality is performed by the TOE's HSM, Nettle, and OpenSSL in support of higher level protocols (TLS, SSH). The modules' FIPS-Approved cryptographic algorithms have obtained CAVP certificates.

The TOE audits security relevant events, stores audit records locally, and can be configured to forward its audit records to an external syslog server in the network environment. An administrator can configure the TOE to solicit time from an NTP server, and alternatively the administrator can manually set the TOE's time.

The TOE uses: TLS to protect syslog, LDAP, SMTP, KMIP and database traffic; offers a management UI protected by TLS/HTTPS; and provides a management Command Line Interface (CLI) protected by SSH.

Administrators are able to query the current version of the product firmware and manage the security functions of the TOE, including performing updates on the product. A published hash is used for protection of the update files.

The TOE provides self-tests to ensure the integrity and correct operation of the TOE.

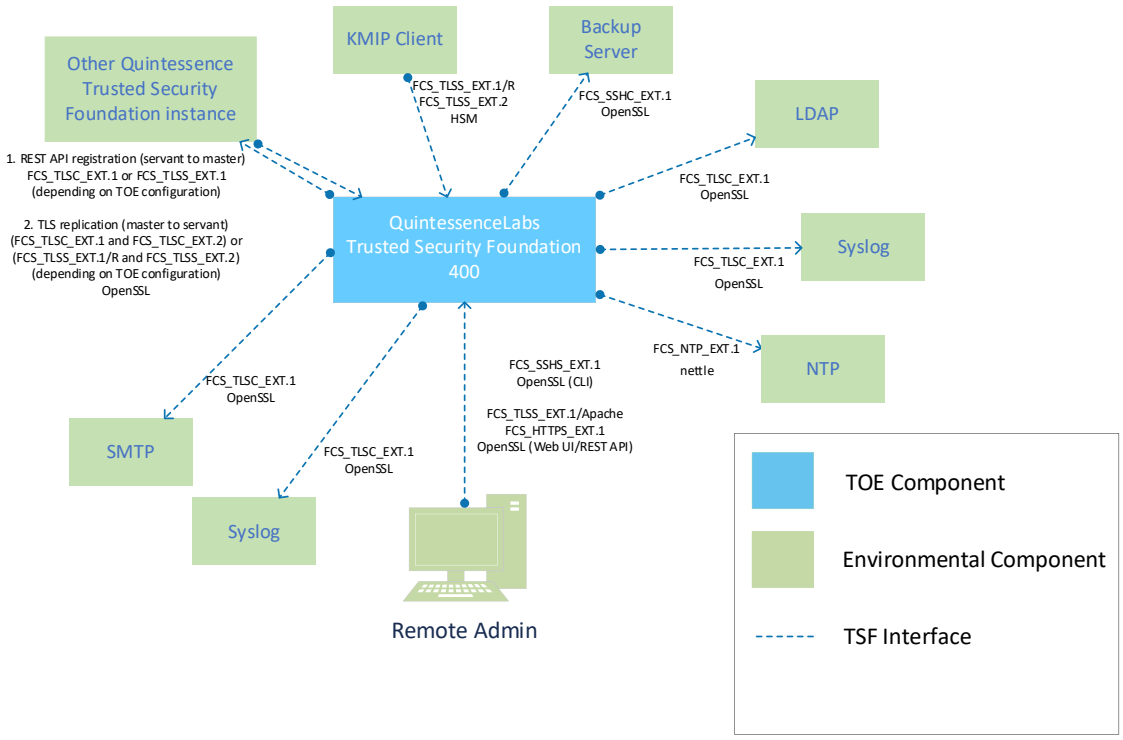
The TOE is operated in 'Common Criteria Mode' configured via the web UI. This mode places the TOE in the evaluated configuration by imposing cryptographic restrictions; disabling watchdogs and other excluded functions; and turning on password restrictions.

[1] QuintessenceLabs refers to the TOE as TSF in its product documentation

The TOE consists of the following component, as shown in Figure 1 below:

- QuintessenceLabs Trusted Security Foundation 400

Figure 1 - TOE Boundary



1.6.1 Physical Scope

The QLabs Trusted Security Foundation 400 TOE is deployed as a single physical appliance based on a Dell R6615. The TOE includes qCrypt Key Management Software v3.2; qStream 100 v3.1 (Quantum Random Number Generator (QRNG)) — in a PCIe-card form factor; and Hardware Security Module (HSM). The HSM contains a NXP® QorIQ T1042 with cryptographic acceleration (Power Architecture®) processor. The TOE also includes a purpose-built distribution of the Red Hat Enterprise Linux (RHEL) 8.10 operating system; and an AMD EPYC 9224 (Zen 4 microarchitecture) processor. Apache is used for the administrative web server providing access to the Web UI and RESTful API. See Table 2 below additional third party components and version numbers.

The Trusted Security Foundation 400 consists of the following physical interfaces:

Component	Description
Power	Dual Hot-plug, Redundant Power Supply (1+1) **PSU specification:** 750W, 50/60Hz, 100-240V AC autoranging, max. 2891 BTU/hr heat dissipation, IEC 60320 C14
Network interfaces	4x 10GbE RJ-45 port (CAT6 or CAT6a cables recommended)
Embedded Management	1x Dedicated iDRAC network port, RJ-45 (back) 1x iDRAC Direct micro port (front)
USB ports	2x USB 3.0 (back) 1x USB 2.0 (front)
Video	1x 15-pin VGA port (front), 1x 15-pin VGA (back)
Optical Drive	DVD-ROM
HSM connector	Mini-DIN connector

Local management is available by plugging a keyboard into a USB port and a monitor into a VGA port, giving access to the Trusted Security Foundation console, which provides a command line interface. SSH access via local network is also available for local management. This interface can be configured to use a public key for user authentication to prevent a situation where local access via password is unavailable due to authentication failure lockout.

Remote management is available from the Web UI (admin console); REST API, and SSH for command line access. Trusted Security Foundation protects interactive communication with remote administrators using SSH (for remote access to the CLI) and HTTPS (for remote access to the Web UI, and RESTful API).

Trusted Security Foundation protects communications with authorized external IT entities using TLS and SSH. Trusted Security Foundation uses TLS to protect communications with: external KMIP clients; other Trusted Security Foundation instances over a database channel; LDAP server; remote syslog server; and SMTP server. A Trusted Security Foundation administrator can use SSHFS to export a Trusted Security Foundation backup to an external backup server.

Additionally, the TOE includes the following third party components:

Table 2: Third Party Components

Component	**Version**
Apache	2.4.37-65.module+el8.10.0+22196+d82931da.2
Entrust (formerly nCipher) HSM	5s
Nuvoton NPCT7xx TPM	2.0
OpenSSL	1.1.1k-14.el8_6
Dell R6615 server	N/A
OpenSSH	8.0p1-25.el8_10
Ldap3	openldap - 2.4.46-20.el8_10
Chrony (NTP daemon)	4.5-2.el8_10

Rsyslog	8.2408.0-1.el8
Postgres	13.19-1PGDG.rhel8
Python (using the smtp module)	3.12.8-1.el8_10
Curl	7.61.1-34.el8_10.3

The TOE in its evaluated configuration requires the following components in its operational environment:

- A TLS-protected syslog server that receives audit events from the TOE
- NTP servers with which the TOE can synchronize its clock
- SMTP server for email notifications
- LDAP server to authenticate remote administrators
- Backup server supporting SSH Server functionality
- At least one other instance of the TOE for database transactions and replication
- KMIP client
- A client workstation for administrator access to the web UI and CLI with:
    - A supported browser:
        - Firefox® web browser, version 31 and later
        - Internet Explorer web browser, version 11
        - Microsoft Edge web browser
        - Safari® web browser, version 6 and later
        - Chrome™ web browser, version 31 and later.
    - An SSH Client for remote access to the CLI.

1.6.2 Logical Scope

This section summarizes the security functions provided by the TOE:

- Security audit
- Cryptographic support
- Identification and authentication
- Security management
- Protection of the TSF
- TOE access
- Trusted path/channels.

1.6.2.1 Security Audit

The TOE generates audit events associated with identification and authentication, management, updates, and user sessions. The TOE can store the events in a local log and export them to a syslog server using a TLS protected channel. The TOE protects stored audit records from unauthorized modification and deletion.

1.6.2.2 Cryptographic Support

The TOE provides CAVP certified cryptography in support of its SSH, TLS, and NTP implementations and for verifying TOE update packages. Cryptographic services include key management, random bit generation, symmetric encryption and decryption, digital signature, and secure hashing.

1.6.2.3 Identification and Authentication

The TOE requires users to be identified and authenticated before they can use functions mediated by the TOE, with the exception of reading the login banner. The TOE authenticates a user's credentials (password, key) using a local mechanism provided by the TOE and supports external LDAP authentication. The TOE also provides X.509 certificate checking for its TLS connections.

1.6.2.4 Security Management

The TOE provides CLI, web-based UI, and RESTful API management interfaces that an administrator can access remotely via a network port. The CLI can also be accessed locally by directly connecting to the appliances via USB port. Remote connections to the management interface are protected with SSH for the CLI and HTTPS for GUI and REST API. The management interface is limited to the authorized administrator.

1.6.2.5 Protection of the TSF

The TOE implements various self-protection mechanisms. The TOE performs self-tests that cover the correct operation of the TOE. It provides functions necessary to securely update the TOE. It relies upon either manually provided time or an NTP server in its environment to ensure reliable timestamps. It encrypts sensitive data such as passwords and cryptographic keys stored within it.

1.6.2.6 TOE Access

The TOE terminates local and remote interactive sessions after a configurable period of inactivity. The TOE additionally provides the capability for administrators to terminate their own interactive sessions. The TOE can be configured to display an advisory and consent warning message before establishing a user session.

1.6.2.7 Trusted Path/Channels

The TOE protects interactive communication with remote administrators using SSH (for remote access to the CLI) and HTTPS (for remote access to the Admin UI and RESTful API).

The TOE protects communications with authorized external IT entities using TLS and SSH. The TOE uses TLS to protect communications with external syslog, LDAP, and SMTP servers and external KMIP clients. The TOE can also transmit or receive replicated information from an external copy of the product in its operational environment over TLS. The TOE uses SSH to transmit backup data to a external backup server.

1.7 TOE Documentation

The TOE is supplied with the following guidance documentation that describes the installation process for the TOE and provides guidance for configuration and secure use of its security features:

- QLabs TSF Common Criteria Evaluated Configuration Guide, Version 1.0
- User Guide – TSF KMS 3.2
- Advanced Administration Guide – TSF KMS 3.2
- Common Criteria CLI Guide – TSF KMS 3.2
- Getting Started Guide – TSF KMS 3.2

1.8 Excluded Functionality

The Trusted Security Foundation is being evaluated as a network device and therefore the key management and policy enforcement capabilities were not evaluated. The list below identifies features or protocols that are not evaluated or must be disabled, and the rationale why. Note that this does not mean the features cannot be used in the evaluated configuration (unless explicitly stated so). It means that the features were not evaluated and/or validated by an independent third party and the functional correctness of the implementation is vendor assertion. Evaluated functionality is scoped exclusively to the security functional requirements specified in this Security Target. In particular, only the following protocols implemented by the TOE have been tested, and only to the extent specified by the security functional requirements: TLS, HTTPS, SSH, NTP authentication. The features below are out of scope.

Table 3: Excluded Functionality

Feature	Description
OASIS KMIP, PKCS#11 over KMIP	The TOE supports OASIS Key Management, the PKCS#11 over KMIP protocol for its key and policy management functions. The [cPPND] does not define requirements for these protocols and functions and therefore they have not been evaluated.
Watchdog port	This provides a mechanism to monitor internal Trusted Security Foundation operations. This connection is not protected and is disabled in the evaluated configuration via setting 'Common Criteria Mode'.
Entropy portal	This provides an entropy service to enable external entities to retrieve random bytes. This service is accessed over an HTTP REST API and is disabled by default.
Guest VM support	The TOE appliance can host multiple virtual machines. Support for Guest VMs has not been evaluated.
iDRAC	Provides a local interface to monitor and configure the Dell appliance. This interface is not used for any of the TOE's management functions and is therefore outside the scope of the TSF.
SNMP	SNMP provides an SNMP agent for use by network management tools. This connection is unprotected and is disabled in the evaluated configuration via setting 'Common Criteria Mode'.
Backups using SMB or NFS v4	The product supports SMB or NFS v4 to export system backups to a remote backup server. Backup data is encrypted but SMB and NFS are not approved protocols. SSHFS and SCP (based on SSH) are supported and should be used instead to export backups manually.
Access control mechanisms (e.g.: rate limiting, policy settings)	Access control mechanisms like rate limiting is outside the scope of the [cPPND] and therefore they have not been evaluated.

Feature	Description
High-availability and multi-master database transactions and replication	The TOE includes the TLS protected communication for database channels. The actual database transactions, replication and high-availability functions have not been evaluated since the [cPPND] does not define those functions.
Load balancing	The product implements hooks to integrate with external 3rd party load balancers. The hooks are provided over plain HTTP (not HTTPS) and are disabled when operating in Common Criteria mode.
Any features not associated with SFRs in claimed [cPPND]	[cPPND] forbids adding additional requirements to the Security Target (ST). If additional functionalities or products are mentioned in the ST, it is for completeness only.

2 Security Problem Definition

This ST includes by reference the Security Problem Definition (comprising threat statements, assumptions, and organizational security policies) from [cPPND]. A.COMPONENTS_RUNNING, A.VS_TRUSTED_ADMINISTRATOR, A.VS_REGULAR_UPDATES, A.VS_ISOLATION, and A.VS_CORRECT_CONFIGURATION do not apply to the TOE because the TOE does not have a distributed TOE or virtual network device configuration. The PP offers additional information about the threats, assumptions, and organizational security policies, but that has not been reproduced here and the PP should be consulted if there is interest in that material.

In general, the [cPPND] has presented a Security Problem Definition appropriate for network infrastructure devices, and as such is applicable to the TOE.

3 Security Objectives

The [cPPND] defines the following security objectives for the operational environment of the TOE. Note that for OE.COMPONENTS_RUNNING and OE.VM_CONFIGURATION do not apply to the TOE because the TOE does not have a distributed TOE or virtual network device configuration.

Table 4: Security Objectives for the Operational Environment

Objective	Description
OE.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is provided by the environment.
OE.NO_GENERAL_PURPOSE	There are no general-purpose computing capabilities (e.g., compilers or user applications) available on the TOE, other than those services necessary for the operation, administration and support of the TOE.
OE.NO_THRU_TRAFFIC_PROTECTION	The TOE does not provide any protection of traffic that traverses it. It is assumed that protection of this traffic will be covered by other security and assurance measures in the operational environment.
OE.TRUSTED_ADMIN	Security Administrators are trusted to follow and apply all guidance documentation in a trusted manner. For TOEs supporting X.509v3 certificate-based authentication, the Security Administrator(s) are assumed to monitor the revocation status of all certificates in the TOE's trust store and to remove any certificate from the TOE's trust store in case such certificate can no longer be trusted.
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.RESIDUAL_INFORMATION	The Security Administrator ensures that there is no unauthorized access possible for sensitive residual information (e.g. cryptographic keys, keying material, PINs, passwords etc.) on networking equipment when the equipment is discarded or removed from its operational environment.

4 IT Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the TOE and to scope the evaluation effort.

The SFRs have all been drawn from [cPPND]. As such, operations on SFRs already performed in that PP are not identified here. Rather, the SFRs have been copied from [cPPND] and any formatting used in that PP has been removed. Operations performed on SFRs in the writing of this ST are identified in accordance with the conventions described in Section 1.3.

The SARs are the set of SARs specified in [cPPND].

4.1 Extended Requirements

All of the extended requirements in this ST have been drawn from [cPPND]. The [cPPND] defines the following extended SFRs and since they are not redefined in this ST, the [cPPND] should be consulted for more information in regards to these CC extensions.

- FAU_STG_EXT.1—Protected Audit Event Storage
- FCS_HTTPS_EXT.1 – HTTPS Protocol
- FCS_NTP_EXT.1—NTP Protocol
- FCS_RBG_EXT.1—Random Bit Generation (iterated by ST author)
- FCS_SSHC_EXT.1—SSH Client Protocol
- FCS_SSHS_EXT.1—SSH Server Protocol
- FCS_TLSC_EXT.1—TLS Client Protocol (iterated by ST author)
- FCS_TLSC_EXT.2—TLS Client Support for Mutual Authentication
- FCS_TLSS_EXT.1—TLS Server Protocol (iterated by ST author)
- FCS_TLSS_EXT.2—TLS Server Support for Mutual Authentication
- FIA_PMG_EXT.1—Password Management
- FIA_UAU_EXT.2—Password-Based Authentication Mechanism
- FIA_UIA_EXT.1—User Identification and Authentication
- FIA_X509_EXT.1—X.509 Certificate Validation
- FIA_X509_EXT.2—X.509 Certificate Authentication
- FIA_X509_EXT.3—X.509 Certificate Requests
- FPT_APW_EXT.1—Protection of Administrator Passwords
- FPT_SKP_EXT.1—Protection of TSF Data
- FPT_STM_EXT.1—Reliable Time Stamps
- FPT_TST_EXT.1—TSF Testing
- FPT_TUD_EXT.1—Trusted Update
- FTA_SSL_EXT.1—TSF-Initiated Session Locking

4.2 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by the TOE.

Table 5: TOE Security Functional Components

Requirement Class	Requirement Component
FAU: Security audit	FAU_GEN.1—Audit Data Generation
	FAU_GEN.2—User Identity Association

Requirement Class	Requirement Component
	FAU_STG.1 – Protected Audit Trail Storage
	FAU_STG_EXT.1—Protected Audit Event Storage
FCS: Cryptographic support	FCS_CKM.1—Cryptographic Key Generation
	FCS_CKM.2—Cryptographic Key Establishment
	FCS_CKM.4—Cryptographic Key Destruction
	FCS_COP.1/DataEncryption—Cryptographic Operation (AES Data Encryption/Decryption)
	FCS_COP.1/SigGen—Cryptographic Operation (Signature Generation and Verification)
	FCS_COP.1/Hash—Cryptographic Operation (Hash Algorithm)
	FCS_COP.1/KeyedHash—Cryptographic Operation (Keyed Hash Algorithm)
	FCS_HTTPS_EXT.1 – HTTPS Protocol
	FCS_NTP_EXT.1—NTP Protocol
	FCS_RBG_EXT.1/OpenSSL —Random Bit Generation (OpenSSL)
	FCS_RBG_EXT.1/HSM —Random Bit Generation (HSM)
	FCS_SSHC_EXT.1 – SSH Client Protocol
	FCS_SSHS_EXT.1—SSH Server Protocol
	FCS_TLSC_EXT.1—TLS Client Protocol without Mutual Authentication
	FCS_TLSC_EXT.2—TLS Client Support for Mutual Authentication
	FCS_TLSS_EXT.1/Apache—TLS Server Protocol without Mutual Authentication (Apache)
	FCS_TLSS_EXT.1/R—TLS Server Protocol without Mutual Authentication (Replication and KMIP)
	FCS_TLSS_EXT.2—TLS Server Support for Mutual Authentication
FIA: Identification and authentication	FIA_AFL.1—Authentication Failure Management
	FIA_PMG_EXT.1—Password Management
	FIA_UAU_EXT.2—Password-Based Authentication Mechanism
	FIA_UAU.7—Protected Authentication Feedback
	FIA_UIA_EXT.1—User Identification and Authentication
	FIA_X509_EXT.1/Rev—X.509 Certificate Validation
	FIA_X509_EXT.2—X.509 Certificate Authentication
	FIA_X509_EXT.3—X.509 Certificate Requests
FMT: Security management	FMT_MOF.1/ManualUpdate—Management of Security Functions Behavior
	FMT_MTD.1/CoreData—Management of TSF Data
	FMT_MTD.1/CryptoKeys—Management of TSF Data
	FMT_SMF.1—Specification of Management Functions
	FMT_SMR.2—Restrictions on Security Roles

Requirement Class	Requirement Component
FPT: Protection of the TSF	FPT_APW_EXT.1—Protection of Administrator Passwords
	FPT_SKP_EXT.1—Protection of TSF Data (for reading of all pre-shared, symmetric and private keys)
	FPT_STM_EXT.1—Reliable Time Stamps
	FPT_TST_EXT.1—TSF Testing
	FPT_TUD_EXT.1—Trusted update
FTA: TOE access	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: Trusted path/channels	FTP_ITC.1—Inter-TSF Trusted Channel
	FTP_TRP.1/Admin—Trusted Path

4.2.1 Security Audit (FAU)

4.2.1.1 Audit Data Generation (FAU_GEN.1)

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).
- Changes to TSF data related to 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).
- [***no other actions***];

d) Specifically defined auditable events listed in Table 6.

FAU_GEN.1.2 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 cPP/ST, information specified in column three of Table 6.

Application Note: *The audit record information for the audit events listed in FAU_GEN.1.1 parts a-c above (i.e., those that are not present in Table 6) are the fields listed in FAU_GEN.1.2 part a.*

Table 6: Security Functional Requirements and Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	None.

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.2	None.	None.
FAU_STG.1	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/DataEncryption	None.	None.
FCS_COP.1/SigGen	None.	None.
FCS_COP.1/Hash	None.	None.
FCS_COP.1/KeyedHash	None.	None.
FCS_HTTPS_EXT.1	Failure to establish a HTTPS session	Reason for failure
FCS_NTP_EXT.1	• Configuration of a new time server • Removal of configured time server	Identity of new/removed time server
FCS_RBG_EXT.1	None.	None.
FCS_SSHC_EXT.1	Failure to establish an SSH session	Reason for failure
FCS_SSHS_EXT.1	Failure to establish an SSH session	Reason for failure
FCS_TLSC_EXT.1	Failure to establish a TLS session	Reason for failure
FCS_TLSC_EXT.2	None.	None.
FCS_TLSS_EXT.1	Failure to establish a TLS session	Reason for failure
FCS_TLSS_EXT.2	Failure to authenticate the client	Reason for failure
FIA_AFL.1	Unsuccessful login attempts limit is met or exceeded.	Origin of the attempt (e.g., IP address).
FIA_PMG_EXT.1	None.	None.
FIA_UIA_EXT.1	All use of identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
FIA_UAU_EXT.2	All use of identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
FIA_UAU.7	None.	None.
FIA_X509_EXT.1/Rev	• Unsuccessful attempt to validate a certificate • Any addition, replacement or removal of trust anchors in the TOE's trust store	• Reason for failure of certificate validation • Identification of certificates added, replaced or removed as trust anchor in the TOE's trust store
FIA_X509_EXT.2	None.	None.
FIA_X509_EXT.3	None.	None.
FMT_MOF.1/ManualUpdate	Any attempt to initiate a manual update.	None.
FMT_MTD.1/CoreData	None.	None.
FMT_MTD.1/CryptoKeys	None.	None.

Requirement	Auditable Events	Additional Audit Record Contents
FMT_SMF.1	All management activities of TSF data.	None.
FMT_SMR.2	None.	None.
FPT_SKP_EXT.1	None.	None.
FPT_APW_EXT.1	None.	None.
FPT_TST_EXT.1	None.	None.
FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure)	None.
FPT_STM_EXT.1	Discontinuous changes to time – either Administrator actuated or changed via an automated process. (Note that no continuous changes to time need to be logged. See also application note on FPT_STM_EXT.1).	For discontinuous 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 (if "terminate the session" is selected)	The termination of a local session by the session locking mechanism.	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.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/Admin	• Initiation of the trusted path. • Termination of the trusted path. • Failure of the trusted path functions.	None.

4.2.1.2 User Identity Association (FAU_GEN.2)

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.

4.2.1.3 Protected Audit Trail Storage (FAU_STG.1)

FAU_STG.1.1 The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

FAU_STG.1.2 The TSF shall be able to prevent unauthorized modifications to the stored audit records in the audit trail.

4.2.1.4 Protected Audit Event Storage (FAU_STG_EXT.1)

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 to FTP_ITC.1.

FAU_STG_EXT.1.2 The TSF shall be able to store generated audit data on the TOE itself. In addition [***The TOE shall consist of a single standalone component that stores audit data locally***].

FAU_STG_EXT.1.3 The TSF shall [***overwrite previous audit records according to the following rule: [the oldest archive audit file is deleted, the current audit file is rotated to be an archive audit file, and a new current audit file is created]***] when the local storage space for audit data is full.

4.2.2 Cryptographic Support (FCS)

4.2.2.1 Cryptographic Key Generation (FCS_CKM.1)

FCS_CKM.1.1 The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm: [

- ***RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3;***
- ***ECC schemes using "NIST curves" [P-256, P-384, P-521] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;***
- ***FFC Schemes using 'safe-prime' groups that meet the following: "NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" and [RFC 3526]***

].

4.2.2.2 Cryptographic Key Establishment (FCS_CKM.2)

FCS_CKM.2.1[2] 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 Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography";***
- ***FFC Schemes using "safe-prime" groups that meet the following: 'NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" and [groups listed in RFC 3526[3]]***

].

4.2.2.3 Cryptographic Key Destruction (FCS_CKM.4)

FCS_CKM.4.1 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 [***logically***

[2] Modified by TD 0581 to allow for selection of Elliptic curve-based key establishment schemes that meet NIST Special Publication 800-56A Revision 3.

[3] Modified by TD0580 to clarify requirements for RFCs are restricted to groups.

addresses the storage location of the key and performs a single overwrite consisting of [zeroes, a new value of the key]];
that meets the following: No Standard.

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

FCS_COP.1.1/DataEncryption The TSF shall perform encryption/decryption in accordance with a specified cryptographic algorithm AES used in [***CBC, CTR, GCM***] mode and cryptographic key sizes [***128 bits, 256 bits***] that meet the following: AES as specified in ISO 18033-3, [***CBC as specified in ISO 10116, CTR as specified in ISO 10116, GCM as specified in 19772***].

Application Note: *Specifically, the TSF supports 256-bit AES-GCM for TLS, 128-bit and 256-bit AES-CTR for SSH, and 256-bit AES-CBC for protection of data at rest.*

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

FCS_COP.1.1/SigGen The TSF shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm [

- ***RSA Digital Signature Algorithm and cryptographic key sizes (modulus) [2048 bits, 3072 bits 4096 bits],***
- ***Elliptic Curve Digital Signature Algorithm and cryptographic key sizes [256 bits, 384 bits, 521 bits]***

]
that meet the following: [

- ***For RSA schemes: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3,***
- ***For ECDSA schemes: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" [P-256, P-384, P-521]; ISO/IEC 14888-3, Section 6.4***

].

Application Note: *4096-bit RSA is used only in the context of signature verification as part of the HSM's power-on self-test.*

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

FCS_COP.1.1/Hash The TSF shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-1***, ***SHA-256***, ***SHA-384***, ***SHA-512***] and message digest sizes [***160***, ***256***, ***384***, ***512***] bits that meet the following: ISO/IEC 10118-3:2004.

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

FCS_COP.1.1/KeyedHash 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 bits, 256 bits, 384 bits, 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".

4.2.2.8 HTTPS Protocol FCS_HTTPS_EXT.1

FCS_HTTPS_EXT.1.1 The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2 The TSF shall implement HTTPS using TLS.

FCS_HTTPS_EXT.1.3 If a peer certificate is presented, the TSF shall [***not require client authentication***] if the peer certificate is deemed invalid.

Application Note: *The TOE uses HTTPS as a Server to protect access to the Web UI and RESTful API.*

4.2.2.9 NTP Protocol (FCS_NTP_EXT.1)

FCS_NTP_EXT.1.1 The TSF shall use only the following NTP version(s) [***NTP v4 (RFC 5905)***].

FCS_NTP_EXT.1.2 The TSF shall update its system time using [

- ***Authentication using [SHA256] as the message digest algorithm(s);***

].

FCS_NTP_EXT.1.3 The TSF shall not update NTP timestamp from broadcast and/or multicast addresses.

FCS_NTP_EXT.1.4 The TSF shall support configuration of at least three (3) NTP time sources in the Operational Environment.

4.2.2.10 Random Bit Generation (OpenSSL) (FCS_RBG_EXT.1/OpenSSL)

FCS_RBG_EXT.1.1/OpenSSL 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/OpenSSL The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from [***[1] platform-based noise sources***] 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: *This is the OpenSSL DRBG which is used to perform all random bit generation functions except for those related to the KMIP server and X.509 key pair generation for CSRs.*

4.2.2.11 Random Bit Generation (HSM) (FCS_RBG_EXT.1/HSM)

FCS_RBG_EXT.1.1/HSM The TSF shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using [***Hash_DRBG (any)***].

FCS_RBG_EXT.1.2/HSM The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from [***[1] platform-based noise sources***] 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: *This is the HSM DRBG which is used to perform random bit generation functions related to the KMIP server and X.509 key pair generation for CSRs.*

4.2.2.12 SSH Client Protocol (FCS_SSHC_EXT.1)

FCS_SSHC_EXT.1.1 The TSF shall implement the SSH protocol in accordance with: RFCs 4251, 4252, 4253, 4254, [***4256, 4344, 5656, 6668, 8268***].

FCS_SSHC_EXT.1.2[4] The TSF shall ensure that the SSH protocol implementation supports the following user authentication methods as described in RFC 4252: public key-based, [***no other method***].

FCS_SSHC_EXT.1.3 The TSF shall ensure that, as described in RFC 4253, packets greater than [**32768**] bytes in an SSH transport connection are dropped.

FCS_SSHC_EXT.1.4 The TSF shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: [***aes128-ctr, aes256-ctr***].

FCS_SSHC_EXT.1.5 The TSF shall ensure that the SSH public-key based authentication implementation uses [***ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521***] as its public key algorithm(s) and rejects all other public key algorithms.

FCS_SSHC_EXT.1.6 The TSF shall ensure that the SSH transport implementation uses [***hmac-sha1, hmac-sha2-256, hmac-sha2-512***] as its data integrity MAC algorithm(s) and rejects all other MAC algorithm(s).

FCS_SSHC_EXT.1.7 The TSF shall ensure that [***diffie-hellman-group14-sha1, ecdh-sha2-nistp256***] and [***diffie-hellman-group14-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, ecdh-sha2-nistp384, ecdh-sha2-nistp521***] are the only allowed key exchange methods used for the SSH protocol.

FCS_SSHC_EXT.1.8 The TSF shall ensure that within SSH connections the same session keys are used for a threshold of no longer than one hour, and each encryption key is used to protect no more than one gigabyte of data. After any of the thresholds are reached, a rekey needs to be performed.

FCS_SSHC_EXT.1.9 The TSF shall ensure that the SSH client authenticates the identity of the SSH server using a local database associating each host name with its corresponding public key and [**no other methods**] as described in RFC 4251 section 4.1.

Application Note: *The TOE acts as an SSH Client with backup servers.*

[4] Modified by TD0636

4.2.2.13 SSH Server Protocol (FCS_SSHS_EXT.1)

FCS_SSHS_EXT.1.1 The TSF shall implement the SSH protocol in accordance with: RFCs 4251, 4252, 4253, 4254, [***4256, 4344, 5656, 6668, 8268***].

FCS_SSHS_EXT.1.2[5] The TSF shall ensure that the SSH protocol implementation supports the following user 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 [**32768**] bytes in an SSH transport connection are dropped.

FCS_SSHS_EXT.1.4 The TSF shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: [***aes128-ctr, aes256-ctr***].

FCS_SSHS_EXT.1.5 The TSF shall ensure that the SSH public-key based authentication implementation uses [***ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521***] as its public key algorithm(s) and rejects all other public key algorithms.

FCS_SSHS_EXT.1.6 The TSF shall ensure that the SSH transport implementation uses [***hmac-sha1, hmac-sha2-256, hmac-sha2-512***] as its data integrity MAC algorithm(s) and rejects all other MAC algorithm(s).

FCS_SSHS_EXT.1.7 The TSF shall ensure that [***Diffie-hellman-group14-sha1, ecdh-sha2-nistp256***] and [***Diffie-hellman-group14-sha256, Diffie-hellman-group16-sha512, Diffie-hellman-group18-sha512, ecdh-sha2-nistp384, ecdh-sha2-nistp521***] are the only allowed key exchange methods used for the SSH protocol.

FCS_SSHS_EXT.1.8 The TSF shall ensure that within SSH connections the same session keys are used for a threshold of no longer than one hour, and each encryption key is used to protect no more than one gigabyte of data. After any of the thresholds are reached, a rekey needs to be performed.

Application Note: *The TOE acts as an SSH Server for remote administration.*

4.2.2.14 TLS Client Protocol without Mutual Authentication

FCS_TLSC_EXT.1.1 The TSF shall implement [***TLS 1.2 (RFC 5246)***] and reject all other TLS and SSL versions. The TLS implementation will support the following ciphersuites:
[
- ***TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

].

[5] Modified by TD0631

FCS_TLSC_EXT.1.2 The TSF shall verify that the presented identifier matches [***the reference identifier per RFC 6125 section 6, IPv4 address in CN or SAN, IPv6 address in the CN or SAN***].

FCS_TLSC_EXT.1.3 When establishing a trusted channel, by default the TSF shall not establish a trusted channel if the server certificate is invalid. The TSF shall also [

- ***Not implement any administrator override mechanism***

].

FCS_TLSC_EXT.1.4 The TSF shall [***present the Supported Elliptic Curves/Supported Groups Extension with the following curves/groups: [secp384r1] and no other curves/groups***] in the Client Hello.

Application Note: *The TOE acts as a TLS client when communicating with external syslog, LDAP, and SMTP servers, when replicating database communications to an environmental instance of the KMS, or when registering to an environmental instance of the KMS to receive replicated database communications.*

4.2.2.15 TLS Client Support for Mutual Authentication (FCS_TLSC_EXT.2)

FCS_TLSC_EXT.2.1 The TSF shall support TLS communication with mutual authentication using X.509v3 certificates.

Application Note: *The TOE only presents a client certificate for TLS mutual authentication when acting as the master node in a database replication configuration (where the subordinate node is in the operational environment).*

4.2.2.16 TLS Server Protocol without Mutual Authentication (Apache) (FCS_TLSS_EXT.1/Apache)

FCS_TLSS_EXT.1.1/Apache The TSF shall implement [***TLS 1.2 (RFC 5246)***] and reject all other TLS and SSL versions. The TLS implementation will support the following ciphersuites: [

- ***TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

].

FCS_TLSS_EXT.1.2/Apache The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and [***TLS 1.1***].

FCS_TLSS_EXT.1.3/Apache The TSF shall perform key establishment for TLS using [

- ***Diffie-Hellman parameters with size [4096 bits],***
- ***ECDHE curves [secp384r1] and no other curves***].

FCS_TLSS_EXT.1.4/Apache The TSF shall support [***session resumption based on session IDs according to RFC5246 (TLS 1.2)***].

Application Note: *This iteration of the SFR applies to the TOE's web server that is used for remote administration. This interface supports TLS session resumption.*

4.2.2.17 TLS Server Protocol without Mutual Authentication (Replication and KMIP) (FCS_TLSS_EXT.1/R)

FCS_TLSS_EXT.1.1/R The TSF shall implement [***TLS 1.2 (RFC 5246)***] and reject all other TLS and SSL versions. The TLS implementation will support the following ciphersuites:
[
- ***TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***

].

FCS_TLSS_EXT.1.2/R The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and [***TLS 1.1***].

FCS_TLSS_EXT.1.3/R The TSF shall perform key establishment for TLS using [
- ***Diffie-Hellman parameters with size [4096 bits],***
- ***ECDHE curves [secp384r1] and no other curves***].

FCS_TLSS_EXT.1.4/R The TSF shall support [***no session resumption or session tickets***].

Application Note: *This iteration of the SFR applies to the TOE's KMIP server, the web server that is used for registration of a database replication channel, and the interface that is used to receive replicated data from an environmental instance of the KMS. This interface does not TLS session resumption.*

4.2.2.18 TLS Server Support for Mutual Authentication (FCS_TLSS_EXT.2)

FCS_TLSS_EXT.2.1 The TSF shall support TLS communication with mutual authentication of TLS clients using X.509v3 certificates.

FCS_TLSS_EXT.2.2 When establishing a trusted channel, by default the TSF shall not establish a trusted channel if the client certificate is invalid. The TSF shall also [
- ***Not implement any administrator override mechanism***

].

FCS_TLSS_EXT.2.3 The TSF shall not establish a trusted channel if the identifier contained in a certificate does not match an expected identifier for the client. If the identifier is a Fully Qualified Domain Name (FQDN), then the TSF shall match the identifiers according to RFC 6125, otherwise the TSF shall parse the identifier from the certificate and match the identifier against the expected identifier of the client as described in the TSS.

Application Note: *Mutual authentication is supported when the TOE is acting as the TLS server in KMIP or database replication communications (i.e. FCS_TLSS_EXT.1/R is mutually-authenticated and FCS_TLSS_EXT.1/Apache is not).*

4.2.3 Identification and Authentication (FIA)

4.2.3.1 Authentication Failure Management (FIA_AFL.1)

FIA_AFL.1.1 The TSF shall detect when an Administrator configurable positive integer within [***1 and 50 for the web UI and between 1 and 65,535 for remote CLI***] unsuccessful authentication attempts occur related to Administrators attempting to authenticate remotely using a password.

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been met, the TSF shall [***prevent the offending Administrator from successfully establishing a remote session using any authentication method that involves a password until [unlock] is taken by an Administrator; prevent the offending Administrator from successfully establishing remote session using any authentication method that involves a password until an Administrator defined time period has elapsed***].

4.2.3.2 Password Management (FIA_PMG_EXT.1)

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: [***"!", "@", "#", "$", "%", "^", "&", "*", "(", ")", ["'"], "'", "+", ",", "-", ".", "/", ":", ";", "<", "=", ">", "?", "[", "\", "]", "_", " " (i.e. empty space character), "`", "{", "|", "}", "~"***];

b) Minimum password length shall be configurable to between [**8 (Web UI users), 6 (CLI users)**] and [**15**] characters.

4.2.3.3 User Identification and Authentication (FIA_UIA_EXT.1)

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 ping requests, accept license terms]***].

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.

4.2.3.4 Password-Based Authentication Mechanism (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1 The TSF shall provide a local [***password-based, SSH public key-based***] authentication mechanism to perform local administrative user authentication.

4.2.3.5 Protected Authentication Feedback (FIA_UAU.7)

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.

4.2.3.6 X.509 Certificate Validation (FIA_X509_EXT.1/Rev)

FIA_X509_EXT.1.1/Rev The TSF shall validate certificates in accordance with the following rules:

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

FIA_X509_EXT.1.2/Rev 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.

4.2.3.7 X.509 Certificate Authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1 The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [***HTTPS, TLS***], and [***no additional uses***].

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***].

4.2.3.8 X.509 Certificate Requests (FIA_X509_EXT.3)

FIA_X509_EXT.3.1 The TSF shall generate a Certificate Request as specified by RFC 2986 and be able to provide the following information in the request: public key and [***Common Name, Organization, Organizational Unit, Country***].

FIA_X509_EXT.3.2 The TSF shall validate the chain of certificates from the Root CA upon receiving the CA Certificate Response.

4.2.4 Security Management (FMT)

4.2.4.1 Management of Security Functions Behavior (FMT_MOF.1/ManualUpdate)

FMT_MOF.1.1/ManualUpdate The TSF shall restrict the ability to enable the functions to perform manual updates to Security Administrators.

4.2.4.2 Management of TSF Data (FMT_MTD.1/CoreData)

FMT_MTD.1.1/CoreData The TSF shall restrict the ability to manage the TSF data to Security Administrators.

4.2.4.3 Management of TSF Data (FMT_MTD.1/CryptoKeys)

FMT_MTD.1.1/CryptoKeys The TSF shall restrict the ability to manage the cryptographic keys to Security Administrators.

4.2.4.4 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1[6] 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 [***hash comparison***] capability prior to installing those updates;
- Ability to configure the authentication failure parameters for FIA_AFL.1; [
  - ***Ability to manage the cryptographic keys*;**
  - ***Ability to re-enable an Administrator account;***
  - ***Ability to set the time which is used for time-stamps*;**
  - ***Ability to configure NTP*;**
  - ***Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors*;**
  - ***Ability to import X.509v3 certificates to the TOE's trust stor*e;**
  - ***Ability to manage the trusted public keys database*;**

  ].

4.2.4.5 Restrictions on Security Roles (FMT_SMR.2)

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.

4.2.5 Protection of the TSF (FPT)

4.2.5.1 Protection of Administrator Passwords (FPT_APW_EXT.1)

FPT_APW_EXT.1.1 The TSF shall store administrative passwords in non-plaintext form.

FPT_APW_EXT.1.2 The TSF shall prevent the reading of plaintext administrative passwords.

[6] Modified by TD0631

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

FPT_SKP_EXT.1.1 The TSF shall prevent reading of all pre-shared keys, symmetric keys, and private keys.

4.2.5.3 Reliable Time Stamps (FPT_STM_EXT.1)

FPT_STM_EXT.1.1 The TSF shall be able to provide reliable time stamps for its own use.

FPT_STM_EXT.1.2 The TSF shall [***synchronise time with an NTP server***].

4.2.5.4 TSF Testing (FPT_TST_EXT.1)

FPT_TST_EXT.1.1 The TSF shall run a suite of the following self-tests [***during initial start-up (on power on)***] to demonstrate the correct operation of the TSF: [**cryptographic algorithm Known Answer Tests (KATs), QRNG start-up tests, HSM Operational Mode Check, software integrity test**].

4.2.5.5 Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1 The TSF shall provide Security Administrators the ability to query the currently executing version of the TOE firmware/software and [***no other TOE firmware/software version***].

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***].

FPT_TUD_EXT.1.3 The TSF shall provide means to authenticate firmware/software updates to the TOE using a [***published hash***] prior to installing those updates.

4.2.6 TOE Access (FTA)

4.2.6.1 TSF-Initiated Session Locking (FTA_SSL_EXT.1)

FTA_SSL_EXT.1.1 The TSF shall, for local interactive sessions, [***terminate the session***] after a Security Administrator-specified time period of inactivity.

4.2.6.2 TSF-Initiated Termination (FTA_SSL.3)

FTA_SSL.3.1 The TSF shall terminate a remote interactive session after a Security Administrator-configurable time interval of session inactivity.

4.2.6.3 User-Initiated Termination (FTA_SSL.4)

FTA_SSL.4.1 The TSF shall allow Administrator-initiated termination of the Administrator's own interactive session.

4.2.6.4 Default TOE Access Banners (FTA_TAB.1)

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.

4.2.7 Trusted Path/Channels (FTP)

4.2.7.1 Inter-TSF Trusted Channel (FTP_ITC.1)

FTP_ITC.1.1 The TSF shall be capable of using [***SSH, TLS, HTTPS***] to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: audit server, [***authentication server, [key management services, database transaction and replication, sending email notifications via SMTP, export a TSF backup to an external backup server, replication actions]***] that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

FTP_ITC.1.2 The TSF shall permit the TSF or the authorized IT entities to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for [**export of audit records to external syslog server, LDAP authentication, database transaction and replication, sending email notifications via SMTP, export a TSF backup to an external backup server, replication management actions**].

4.2.7.2 Trusted Path (FTP_TRP.1/Admin)

FTP_TRP.1.1/Admin The TSF shall be capable of using [***SSH, HTTPS***] 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/Admin The TSF shall permit remote Administrators to initiate communication via the trusted path.

FTP_TRP.1.3/Admin The TSF shall require the use of the trusted path for initial Administrator authentication and all remote administration actions.

4.3 TOE Security Assurance Requirements

The security assurance requirements for the TOE are included by reference from the [cPPND].

Table 7: Assurance Components

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1 Basic functional specification
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.1 Labelling of the TOE
	ALC_CMS.1 TOE CM coverage
ATE: Tests	ATE_IND.1 Independent testing – conformance
AVA: Vulnerability assessment	AVA_VAN.1 Vulnerability survey

5 TOE Summary Specification

This section describes the following security functions implemented by the TOE to satisfy the SFRs claimed in Section 4.2:

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

5.1 Security Audit

5.1.1 Audit Data Generation

The TOE generates audit records for: start-up and shut-down of the audit functions; and the administrative actions:

- Administrative login and logout.
- Changes to TSF data related to configuration changes (in addition to the information that a change occurred, the TOE logs 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 is logged).
- Resetting passwords (name of related user account is logged).

Additionally, the TOE logs the specifically defined auditable events listed in Table 6.

The TOE records the following information within each audit record: date and time of the event, type of event, subject identity, the outcome (success or failure) of the event; and for each audit event type, based on the auditable event definitions of the functional components included in the cPP/ST, information specified in column three of Table 6.

All auditable events that involve configuration of the TOE include the action requested, the success or failure, and the identity of the user that made the request. The TOE uses the following keys: SSH host keys, SSH client keys, TLS private keys and TLS public keys / certificates. Once in Common Criteria mode, creation of SSH keys is recorded in /var/log/messages, and the keys are identified by their key fingerprint. TLS private keys are each associated with a TLS certificate or CSR. Operations involving the TLS certificates or CSRs (creation, import, destruction) are recorded in the audit log.

Each CSR or certificate has an internal name, which is unique within the TOE, and is included in the audit messages. In addition:

- Each audit message about a certificate includes the Secure Key ID (SKID) of the certificate.
- Each audit message about a CSR includes the Common Name of the CSR.

This aspect of the Security Audit security function satisfies FAU_GEN.1 and FAU_GEN.2.

5.1.2 Audit Storage and Audit Record Export

The TOE is a single standalone appliance that stores audit logs locally in the log file (/var/log/messages) and provides the administrator the ability to configure the real-time export of syslog records protected

with TLS. To enable sending to a remote Syslog server, log into the web UI, and navigate to Auditing / Remote Logging. First, configure the TLS certificate to use (by clicking *set tls certificate*) then click *add destination* and configure the details of the remote log server.
The TOE implements a log rotation policy to manage the log files it generates. The size of log files are configurable via the Web UI (using page Auditing / Log Rotation). Rotations are size-based where the value selected must be at least 1 MB. The upper limit on configurable size can be no more than 1,000,000 MB, however a smaller cap might apply: if configured to hold *n* rotated (*Lower limit 1, upper limit 99)* versions of the log, the largest acceptable size limit is $\frac{A-250}{n+1}$ MB, where A is the amount of space allocated to the /var/log directory. The size of the /var/log directory is determined by the free space available on the disk of the installed system. After installation, this space is ~= 1.7 TB. The free space will change depending on system usage, e.g., storing a large number of keys in the database, or storing multiple database backups. Consequently, the limit is not a fixed limit. The value of n is chosen by the operator; defines the number of rotated versions to maintain; and can be configured as an integer between 1 and 99. For example, if configured to keep 7 copies of a log file, and if /var/log has 500GB available, the maximum size of the log can be set up to a limit of $\frac{500\ x\ 1024-250}{8}$ = 63,968 MB. When the current log file reaches its rotation threshold, it is closed and rotated to an archive, and a new current log file is created. If the maximum number of archive files already exists, the oldest one is deleted.

The TOE does not provide any interfaces to modify the stored audit logs and can only be viewed by a Security Administrator. Audit logs for TOE functionality cannot be cleared via the web UI, but can be deleted manually via CLI, by a user with the root password.

This aspect of the Security Audit security function satisfies FAU_STG.1 and FAU_STG_EXT.1.

5.2 Cryptographic Support

Cryptographic functionality is performed by HSM's nShield 5 Algorithm Library-nCore v13.4.5, OpenSSL 1.1.1, and Nettle Cryptographic Library 3.4.1 from RHEL 8.10. The SHA-256 cryptographic function provided by Nettle is used for NTP authentication. The TOE uses the OpenSSL library: for TLS client functionality, Postgres, HTTPS TLS server functionality, and certificate validation in all cases except for the KMIP interface. The OpenSSH library (running on top of OpenSSL) is used for SSH functionality. The TOE uses the Entrust 5s HSM's TLS functionality for KMIP TLS servers, as well as certificate functionality including X.509 key pair generation, validation, signing of CSRs, and for the KMIP certificate, encryption, and decryption. The TOE's qStream 100 v3.1 module is used as an entropy source only.

Table 8: Validated Algorithm Implementations

Functions	Standards	Certificates
Asymmetric Key Generation (FCS_CKM.1)		
RSA (2048 bits, 3072 bits, 4096 bits)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3	OpenSSL: RSA #A6764 HSM: RSA #A3707

Functions	Standards	Certificates
ECDSA (P-256, P-384, P-521 curves)	FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4	OpenSSL: ECDSA #A6764 HSM: ECDSA #A3707
FFC Schemes using ‘safe-prime’ groups and RFC 3526 (MODP 2048, MODP 4096, MODP 8192)	“NIST Special Publication 800-56A Revision 3 and RFC 3526	OpenSSL: Safe Primes Key Gen/Ver #A6764 HSM: Safe Primes Key Gen/Ver #A3707
Key establishment (FCS_CKM.2)		
Elliptic curve-based scheme (P-256, P-384, P-521 curves)	NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”	OpenSSL KAS-ECC-SSC Sp800-56Ar3 #A6764 HSM: KAS-ECC Sp800-56Ar3 #A3707
FFC Schemes using “safe-prime” groups (MODP 2048, MODP 4096, MODP 8192)	NIST Special Publication 800-56A Revision 3 and RFC 3526 groups	OpenSSL: FFC KAS-FFC-SSC Sp800-56Ar3 #A6764 HSM: KAS-FFC Sp800-56Ar3 #A3707
Data encryption (FCS_COP.1/DataEncryption)		
AES in CBC mode (256 bits) AES in CTR mode (128, 256 bits) AES in GCM mode (256 bits)	ISO 18033-3 (AES) ISO 10116 (CBC mode) ISO 10116 (CTR mode) ISO 19772 (GCM mode) *Note: OpenSSL implements AES-CTR for SSH and AES-GCM for TLs. HSM implements AES-CBC for protection of data at rest and GCM for SSH.*	OpenSSL: AES CTR, GCM #A6764 HSM: AES CBC, GCM #A3707

Functions	Standards	Certificates
Digital signature generation and verification (FCS_COP.1/SigGen)		
RSA Digital Signature Algorithm (2048, 3072, 4096 bit)	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	OpenSSL: RSA SigGen (2048/3072), RSA SigVer (2048/3072) #A6764 HSM: RSA RSA SigGen (2048/3072), RSA SigVer (2048/3072/4096) #A3707
ECDSA Elliptic Curve Digital Signature Algorithm (P-256, P-384, P-521 curves)	FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 6 and Appendix D, Implementing “NIST curves” [P-256, P-384, P-521]; ISO/IEC 14888-3, Section 6.4	OpenSSL: ECDSA SigGen/SigVer #A6764 HSM: ECDSA SigGen/SigVer #A3707
Cryptographic hashing (FCS_COP.1/Hash)		
SHA-1 (digest size 160 bits) SHA-256 (digest size 256 bits) SHA-384 (digest size 384 bits) SHA-512 (digest size 512 bits)	ISO/IEC 10118-3:2004	OpenSSL: SHA-1, SHA2-256/384/512 #A6764 Nettle: SHA2-256 #A5443 HSM: SHA-1, SHA2-256/384/512 #A3707

Functions	Standards	Certificates
Keyed-hash message authentication (FCS_COP.1/KeyedHash)		
HMAC-SHA-1 (key sizes: 256-2048 bits in 128 bit increments, digest size 160 bits) HMAC-SHA-256 (key sizes: 256, 448, 512, 1536, 2048 bits, digest size 256 bits) HMAC-SHA-384 (key sizes: 192, 320, 1024, 1920, 2048 bits, digest size 384 bits) HMAC-SHA-512 (key sizes: 256, 448, 1024, 1536, 2048 bits, digest size 512 bits)	ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"	OpenSSL: HMAC-SHA-1, HMAC-SHA2-256/384/512 #A6764 HSM: HMAC-SHA-1, HMAC-SHA2-256/384/512 #A3707
Deterministic random bit generation (FCS_RBG_EXT.1)		
AES-256 CTR_DRBG	ISO/IEC 18031:2011	OpenSSL: Counter DRBG #A6764
SHA-256 Hash DRBG		HSM: Hash DRBG #A3707

5.2.1 Cryptographic Operations

The TOE uses most cryptographic algorithms in support of TLS and SSH; but also in support of NTP Authentication (FCS_NTP_EXT.1), X.509 CSR generation (FIA_X509_EXT.3), TOE integrity checking (FPT_TST_EXT.1), and Update integrity checking (FPT_TUD_EXT.1).

The TOE uses SHA hashing during digital signature calculation (hashing of the message); for self-test integrity, for integrity as part of HMAC-SHA operations within TLS (SHA384) and SSH (SHA1, SHA256, SHA512); for the persistence of administrator passwords (CLI: SHA512, Web UI: SHA256); for checking integrity of trusted updates (SHA256); and also for authentication of NTP servers (SHA256).

The TOE implements AES-GCM (both 128 and 256-bit) for TLS; 128/256 AES-CTR for SSH and 128/256 AES-CTR for the DRBG. The TOEs HSM implements AES-256-CBC for encrypting the TLS private keys for KMIP and the master HSM key for storing on the HSM.

The TOE will generate ephemeral 2048/4096/8192 MODP Safe-Prime (2048/8192 MODP are used in SSH only) or ECDSA P-256/P-384/P-521 ephemeral elliptic curve parameters for key establishment in TLS/SSH. P-256 and P-521 are only used for SSH. The TOE will use ECDSA P-256/P-384/P-521 or 2048/3072-bit RSA signature generation and verification operations as part of peer authentication. ECDSA SigGen/SigVer is only used for SSH.

The TOE performs SHA-1/256/384/512 cryptographic hashing with message digest sizes 160, 256, 384, 512, in bits that meets ISO/IEC 10118-3:2004. The TOE performs HMAC-SHA-384 for TLS HMAC-SHA-1, HMAC-SHA-256,or HMAC-SHA-512 (depending on negotiated cipher suite) for integrity of SSH protected data using SHA384 (for TLS) and SHA-1/SHA256/SHA512 (for SSH) with key sizes 256-2048 bits to produce a 160/256/384/512 output MAC. SHA-256 is also used for digital signatures and integrity checks during startup self-tests. The HMAC-SHA-1 and HMAC-SHA-256 algorithms have a block size of 512 bits, while

the HMAC-SHA-384 and HMAC-SHA-512 algorithms have a block size of 1024 bits. The keyed-hash message authentication cryptographic algorithms have the following cryptographic key sizes: for HMAC-SHA-1: 256-2048 bits in 128 bit increments; for HMAC-SHA2-256: 256, 448, 512, 1536, 2048 bits; for HMAC-SHA2-384: 192, 320, 1024, 1920, 2048 bits; and for HMAC-SHA2-512: 256, 448, 1024, 1536, 2048 bits. The TOE's HSM uses SHA-256 for its DRBG.

This aspect of the Cryptographic Support security function satisfies FCS_COP.1/DataEncryption, FCS_COP.1/SigGen, FCS_COP.1/Hash, and FCS_COP.1/KeyedHash.

5.2.2 Random Bit Generation

The TOE performs deterministic random bit cryptographic operations using either AES-256 CTR_DRBG provided by OpenSSL or the HSM's SHA-256 Hash_DRBG.

The TOE seeds an AES-256 CTR_DRBG provided by OpenSSL by reading 2 128-bit segments from the qStream 100 QRNG. The qStream 100 QRNG provides 128 bits of entropy in every 128-bit block it outputs and these are concatenated to provide the 256 bits of entropy needed to seed the AES-256 CTR_DRBG. The QRNG is physical noise source that measures a quantum random process (tunneling noise source) to generate entropy.

The HSM entropy source has gone through Entropy Source Validation (E38, https://csrc.nist.gov/projects/cryptographic-module-validation-program/entropy-validations/certificate/38). It is used solely by the HSM to seed its own DRBG to support internal key generation. As described in the proprietary EAR, the HSM's Physical True Random Number Generator (PTRNG) entropy source is a physical noise source providing 0.89 bits of min-entropy per output bit seeding the HSM's CAVP validated SHA-256 Hash_DRBG.

Details of the calculation and the physical platform-based entropy sources are provided in the proprietary EAR.

When configured appropriately using the guidance documentation, entropy sources for different Trusted Security Foundation functionality are as follows:

Entropy Source	**Used for**
QRNG card	TLS handshake for KMIP over TLS TLS handshake for protocols other than KMIP (HTTPS, LDAP, SMTP, Postgres, syslog) SSH host & session keys HTTPS private keys Locally created private keys with 'client' or 'web server' usage designation TLS private keys for both replication and non-replication connections to Postgres databases.
HSM	Locally created private keys with 'internal' usage designation (used for KMIP TLS connection)

	Private keys for locally generated certificate authorities TLS private keys used for Postgres replication

This aspect of the Cryptographic Support security function satisfies FCS_RBG_EXT.1/OpenSSL, FCS_RBG_EXT.1/HSM.

5.2.3 Cryptographic Key Generation and Establishment

The TOE supports generating key pairs, both for authentication and for key exchange for SSH and TLS. When generating authentication key pairs, the TOE can generate a CSR with RSA 2048 and 3072 bit key pairs; and ECDSA P-256, P-384, P-521 key pairs.

For key establishment, the TOE generates 2048/4096/8192 MODP Safe-Prime (2048 and 8192 are only used in SSH), or ECDSA P-256/P-384/P-521 (P-256 and P-521 are only used in SSH) ephemeral elliptic curve parameters for key establishment in TLS/SSH (depending on the TLS interface and negotiated cipher suite) for TLS, and SSH. The TOE acts as a TLS client when exporting audit records to an external audit server; for both database transaction and replication services (and management of); and when communicating with external LDAP and SMTP servers. The TOE acts as a TLS server supporting inbound administrative sessions; replication services; and for communications with KMIP clients.

The TOE's SSH implementation uses only the following key exchange methods: Diffie-hellman-group14-sha1, Diffie-hellman-group14-sha256, Diffie-hellman-group16-sha512, Diffie-hellman-group18-sha512, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521.

This aspect of the Cryptographic Support security function satisfies FCS_CKM.1 and FCS_CKM.2.

5.2.4 Cryptographic Key Destruction

The TOE clears keys from volatile memory by overwriting the memory locations in RAM with zeroes. For plaintext keys in non-volatile storage, the TOE takes two approaches.

The ephemeral TPM key exists only within the TPM and is only accessible using a random secret stored on the hard drive. The TOE generates this random secret during installation, which is passed to the TPM at startup. The TPM uses the random secret in combination with its own internal seed to generate a new key – the key that the TPM generates is the same each time, and this is the KEK used to protect the TPM key vault. Upon executing the administrator command /usr/local/bin/mount_vault (via the command line), this secret is overwritten with zeroes.

For the HSM, the administrator can delete the security world via the cli command: rm -rf /opt/nfast/kmdata that deletes all keys within the HSM. Refer to Table 9 below for the keys and CSPs stored by the TOE along with their method of storage and destruction.

Table 9: Key Clearing

Key	Storage Location	How Stored	Usage and destruction
SSH Host Private Key	File system	Encrypted, AES-256-XTS, key stored in TPM.	Used by SSH for client authentication of the SSH server, which is the TOE. It

			is replaced with a new key at request of the administrator. This key would implicitly be erased by zeroizing the TPM key that encrypts it. SSH host keys are also rotated when entering Common Criteria mode via the web interface.
SSH Client private key for user authentication	TPM vault	Encrypted, AES-256-XTS	Used for user authentication in SSH connections (in particular the private key for the SSH Client to remote backup servers).
SSH Session Key	RAM	plaintext	Used by SSH to encrypt a session and is destroyed when the SSH session is closed.
TLS Session Key	RAM	Plaintext	Used by TLS to encrypt a session and is destroyed when the TLS session is closed.
Diffie-Hellman Shared Secret	RAM	Plaintext	Used by the SSH and TLS protocols and is destroyed at the completion of the DH exchange step of those protocols
Syslog Certificate Private Key	TPM vault	Encrypted, AES-256-XTS	Used for connections with external syslog servers. Can be replaced by the user via the web interface. The KEK is the TPM key and can be cleared by the root user with the cli command: /usr/local/bin/mount_vault -d
Web Server (Apache) Certificate Private Key	TPM	Encrypted, AES-256-XTS, key stored in TPM.	Used for web server. Generated on system install, can be replaced by the user via the web interface. This key would implicitly be erased by zeroizing the TPM key that encrypts it.
KMIP Server Certificate Private Key	HSM	Encrypted, AES-256-CBC	Used for KMIP server. Generated on system install, can be replaced by the user via the web interface. The KEK is the DEK (HSM master key) which is itself encrypted (AES-256-CBC).

Database TLS Certificate Private Keys	TPM (All database related keys are stored encrypted in the db_credentials folder in the TSM vault area.)	Encrypted, AES-256-XTS	Used for Database server. Generated on system install, can be replaced by the user via the web interface. The KEK is the TPM key and can be cleared by the root user with the cli command: /usr/local/bin/mount_vault -d
TPM random secret	TPM	Plaintext	Used to generate the TPM key that is used to unlock secrets protected by the TPM. The secret is stored in the TPM and cannot be retrieved from the TPM. It is initialized when the TPM is setup and is destroyed via the tpm2_clear command. As described above, the TPM key is the kek and is epehemeral (not persistently stored but is generated dynamically by the TPM). This key is derived from the seed and the internal secret stored in the TPM. The root user can use the CLI /usr/local/bin/mount_vault -d via the command line is used to overwrite the secret with zeroes. This implicitly overwrites the TPM key.
TPM seed	File System	Plaintext	The command /usr/local/bin/mount_vault -d wipes the seed (overwrites with zeros) and also clears the TPM using the tpm2_clear command. The tpm2_clear command resets the TPM. This means that even if the seed and LUKS container have not been destroyed (ie saved in a backup), the LUKS container can no longer be decrypted as it is no longer possible to obtain the key from the TPM even if the seed value is known.

			This implicitly overwrites the TPM key.
DEK (Master HSM key)	HSM	Encrypted, AES-256-CBC	Data that requires encryption with the DEK is encrypted by the HSM. Data encrypted by the DEK must be decrypted by the HSM. Such encryption and decryption are only possible if the operator card is inserted in the card reader and the password to the operator card is known. The DEK cannot leave the HSM boundary as plain text. All keys within the HSM can be deleted by deleting the security world via the cli command: rm -rf /opt/nfast/kmdata. As the data is encrypted by the HSM no special wiping for the DEK is necessary.

The TOE uses OpenSSL to provide the cryptographic services required by TLS and SSH connections. There are two modes in which the TOE uses OpenSSL.

In the first mode, the TOE uses the built-in OpenSSL implementations of cryptographic algorithms (from Table 8). This mode is used to perform all HTTPS operations, to carry out all TLS operations (aside from key generation) for all TLS links except for KMIP, and for all SSH operations.

The SSH server keys are stored on disk, encrypted using AES-256 in XTS mode using a key that is stored inside the server's TPM chip. The SSH server code reads these server keys into volatile memory at startup, decrypting them on the fly using the dm-crypt subsystem within the Linux kernel. During processing of these keys, intermediate memory which stored private key material is zeroed (via calls to library routine explicit_bzero) before being released. During normal server operations, once an SSH client has established a channel with the server, the server code will zeroize and release the server keys in volatile memory as part of setting up a child process to handle the connection.

In the second mode, the TOE uses an OpenSSL Engine that delegates implementation of cryptographic algorithms to a built-in HSM. This mode is used to perform TLS operations for KMIP links, and to generate X509 certificates for securing all TLS links.

Certificates are stored on the file system of the TOE. Private keys for each certificate are stored on the file system encrypted using AES-256 in XTS mode, using a key stored within the TPM. Deletion of the certificate and encrypted key is performed through the web interface. This will trigger a file system *unlink* operation to remove references to the encrypted key and certificate from the file system.

When securing the TLS KMIP link, the HSM is loaded with the private key corresponding to the certificate used by the KMIP TLS server. This private key remains within the FIPS boundary of the HSM while unencrypted. Entropy for the TLS handshake is provided from the QRNG entropy source, provided via an OpenSSL callback to the OpenSSL handshake logic.

All keys, key material, and authentication credentials are protected from unauthorized disclosure.

This aspect of the Cryptographic Support security function satisfies FCS_CKM.4.

5.2.5 Cryptographic Protocols

The TOE implements the following cryptographic protocols to protect communications between itself and non-TOE entities:

- TLS as a client—the TOE acts as a TLS client when exporting audit records to an external audit server; for database transaction and replication services; and when communicating with external LDAP and SMTP servers.
    - Mutual authentication is supported for the TLS client for Database Transaction and Replication channels.
- TLS as a server—the TOE acts as a TLS server supporting inbound administrative sessions; replication services; and for communications with KMIP clients.
    - Mutual authentication is supported for the TLS Server for the following: KMIP communications; and Database Transaction and Replication channels.
- HTTPS as a client and a server – the TOE acts as both an HTTPS client and a server for the REST API management for database replication (replication actions). The TOE acts as an HTTPS server for connections with remote administrators.
- SSH—the TOE acts as an SSH Client for connections with remote backup servers and as an SSH server supporting inbound administrative sessions.
- NTP—the TOE can synchronize its system clock with an NTP server.

5.2.5.1 SSH Client and Server Protocol

The SSH client and server implementations are identical except where otherwise specified.

The TOE's SSH server and SSH client both implement the SSH protocol in accordance with RFCs 4251, 4252, 4253, 4254, 4256, 4344, 5656, 6668, 8268. The TOEs SSH implementation does not support any optional characteristics.

The TOE's SSH server supports both SSH public key-based and password-based authentication methods as described in RFC 4252. The TOE's SSH client supports public key authentication only. For both interfaces, ECDSA is the key pair algorithm supported for the user key, "large packets" (in this case greater than 32768 bytes) in an SSH transport connection are dropped as specified by RFC 4253.

For its SSH transport implementation, the TOE supports aes128-ctr and aes256-ctr and rejects all other encryption algorithms.

The SSH transport implementation uses ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 as it supported host key algorithms.

The SSH transport implementation uses hmac-sha1, hmac-sha2-256, and hmac-sha2-512 as its data integrity MAC algorithms and rejects all other MAC algorithms.

The TOE's SSH implementation uses Diffie-hellman-group14-sha1, ecdh-sha2-nistp256, Diffie-hellman-group14-sha256, Diffie-hellman-group16-sha512, Diffie-hellman-group18-sha512, ecdh-sha2-nistp384, and ecdh-sha2-nistp521 for its key exchange methods.

Within SSH connections the same session keys are used for a threshold of no longer than one hour, and each encryption key is used to protect no more than one gigabyte of data. After any of the thresholds are reached, the TOE performs a rekey.

This aspect of the Cryptographic Support security function satisfies FCS_SSHC_EXT.1 and FCS_SSHS_EXT.1.

5.2.5.2 TLS Client Protocol

The TOE's TLS client, used for connections with syslog, SMTP, LDAP, and database replication endpoints, supports TLS 1.2 with the following TLS ciphersuites:

- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

The TOE allows the TLS client (used for audit export) to specify the syslog server by hostname or IPv4/IPv6 in the CN; or DNS in the SAN. IP addresses in the SAN are not supported. Certificate validation is done for validity (e.g. cryptographically valid, not expired, not revoked). The TOE's TLS client implementation establishes its reference identifiers from the administrator-configured reference identifiers as follows. When the SAN extension is present in a certificate, the CN is ignored and verification is performed using the DNS name in the Subject Alternative Name. When the SAN is not present in the certificate, verification is performed in accordance with Section 6 of RFC 6125, using the hostname or IP address as a reference identifier and checking that the syslog server's certificate includes the specified identifier. This interface is not mutually authenticated; the TOE does not present a client certificate to the remote peer.

The TOE allows the TLS client (used for authenticating the user) to specify the remote LDAP server by hostname in the CN; DNS in the SAN; or IPv4/IPv6 addresses in the CN or SAN. Certificate validation is done for validity (e.g. cryptographically valid, not expired, not revoked). When the SAN extension is present in a certificate, the CN is ignored and verification is performed using the DNS name and IPAddress in the Subject Alternative Name. When the SAN is not present in the certificate, verification is performed in accordance with Section 6 of RFC 6125, using the hostname or IP address as a reference identifier and checking that the LDAP server's certificate includes the specified identifier. This interface is not mutually authenticated; the TOE does not present a client certificate to the remote peer.

The TOE allows the TLS client (used for sending emails) to specify the remote SMTP server by hostname in the CN; DNS in the SAN; or IPv4/IPv6 addresses in the CN or SAN. Certificate validation is done for validity (e.g. cryptographically valid, not expired, not revoked). When the SAN extension is present in a certificate, the CN is ignored and verification is performed using the DNS name and IPAddress in the Subject Alternative Name. When the SAN is not present in the certificate, verification is performed in accordance with Section 6 of RFC 6125, using the hostname or IP address as a reference identifier and checking that the SMTP server's certificate includes the specified identifier. This interface is not mutually authenticated; the TOE does not present a client certificate to the remote peer.

The TOE acts as a TLS client when transmitting database information to the operational environment via the database replication channel. The remote server is specified by hostname or IPv4/IPv6 in the CN; or DNS in the SAN. IP addresses in the SAN are not supported. Presented certificates are validated (cryptographically valid, not expired, not revoked). When the SAN extension is present in a certificate, the

CN is ignored and verification is performed using the DNS name and IPAddress in the Subject Alternative Name. When the SAN is not present in the certificate, verification is performed in accordance with Section 6 of RFC 6125, using the hostname or IP address as a reference identifier and checking that the peer's certificate includes the specified identifier. This interface is mutually authenticated, so the TOE presents a client certificate to the remote peer.

The TOE acts as a TLS client when registering to an environmental instance of the KMS to receive inbound communications for database replication. The remote service is specified by hostname in the CN; DNS in the SAN; or IPv4/IPv6 addresses in the CN or SAN. When the SAN extension is present in a certificate, the CN is ignored and verification is performed using the DNS name and IPAddress in the Subject Alternative Name. When the SAN is not present in the certificate, verification is performed in accordance with Section 6 of RFC 6125, using the hostname or IP address as a reference identifier and checking that the peer's certificate includes the specified identifier. This interface is not mutually authenticated; the TOE does not present a client certificate to the remote peer.

In all cases where IP address is used for reference identification, the TOE converts the text representation of the IP address in the CN to a binary representation of the IP address in network byte order; and canonical format is enforced according to RFC 5952 for IPv6, and according to RFC 3986 for IPv4.

When an elliptic curve cipher suite is negotiated, the TLS client supports the Elliptic Curves Extension (specifying only P-384) in its Client Hello, and the TOE does not require (nor allow) administrative configuration of this extension; the TOE always sends it.

For all interfaces, the TOE's TLS client does not support wildcards. When establishing a trusted channel, by default the Trusted Security Foundation does not establish a trusted channel if the server certificate is invalid and there are no administrator override mechanisms to change this behavior.

This aspect of the Cryptographic Support security function satisfies FCS_TLSC_EXT.1, FCS_TLSC_EXT.2, and FCS_HTTPS_EXT.1.

5.2.5.3 TLS Server Protocol

The TOE implements TLS server functionality for inbound management requests over HTTPS; for KMIP communications; for inbound registration of a database replication channel from an environmental instance of the KMS over HTTPS; and for inbound database communications with an environmental instance of the KMS over TLS. The HTTPS management interface supports both Web UI and RESTful API communications. The registration channel for database registration is an HTTPS connection that uses a REST API to accept inbound registration requests from the operational environment. The TOE's HTTPS implementation conforms to RFC 2818. The TOEs TLS server supports TLS 1.2.

The TOE's TLS server supports the following ciphersuites:

- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

The TOE denies versions of TLS older than TLS 1.2 and only the Apache server supports session resumption based on session IDs according to RFC5246 (TLS 1.2). The TOE uses 4096-bit DHE or P-384 for TLS key exchange using P-384, depending on the negotiated ciphersuite.

Mutual authentication is supported for inbound KMIP communications and for replication of database information. The TOE does not support any fallback authentication methods to authenticate TLS clients that do not present a certificate. If a valid certificate is not presented, the connection is not allowed. The presented client certificate is checked against a database of pre-approved clients. This database is set up

by the TOE administrator – each entry contains the serial number of a valid certificate and the CN of the CA that issued that certificate. If this is validated, the IP address of the client is checked against an expected IP address identified in the SAN. If the certificate is not found in the database or the identifier in the certificate does not match the client's, the TOE will not establish a trusted channel with the client. There is no administrator override mechanism to change this behavior.

This aspect of the Cryptographic Support security function satisfies FCS_HTTPS_EXT.1, FCS_TLSS_EXT.1/Apache, FCS_TLSS_EXT.1/R, and FCS_TLSS_EXT.2.

5.2.5.4 HTTPS Protocol

The TOE supports HTTPS in a server capacity to secure the sessions for remote administration over TLSv1.2. The TOE also acts as both an HTTPS client and a server for the REST API management for database replication (replication actions). The TOE's HTTPS protocol complies with RFC 2818 and uses the TLS functionality described above. RFC 2818 (HTTP Over TLS) describes how to use TLS to secure HTTP connections over the Internet. The details of the TOE's behavior with regards to the following requirements listed in RFC 2818 are as follows:

- Section 2.1, Connection Initiation:
  - o The TOE's HTTP client also acts as the TLS client
  - o The TOE sends all HTTP data as TLS application data
  - o The TOE meets described behavior; no deviation
- Section 2.2, Connection Closure:
  - o The TOE sends TLS closure alert when terminating an HTTPS connection
  - o The TOE supports session reuse
  - o The TOE meets described behavior; no deviation
- Section 2.2.1, Client Behavior:
  - o The TOE treats premature closes as errors and the data received as potentially truncated
  - o The TOE sends a closure alert before closing the connection
  - o The TOE meets described behavior; no deviation
- Section 2.2.2, Server Behavior:
  - o The TOE supports TLS session resumption based on session IDs according to RFC5246
  - o The TOE will attempt to initiate an exchange of closure alerts with the client before closing the connection
- Section 2.3, Port Number:
  - o The TOE utilizes TCP port 443 to listen for incoming HTTPS connections
- Section 2.4, URI Format:
  - o The TOE supports and requires the "https://" URI protocol identifier prefix for incoming HTTPS requests
- Section 3.1, Server Identity:
  - o The TOE checks the server's hostname against the server's identity as presented in the server's Certificate message
  - o The TOE allows the REST API server to be specified by hostname or IPv4/IPv6 in the CN; or DNS in the SAN. IP addresses in the SAN are not supported
  - o The TOE meets described behavior; no deviation
- Section 3.2, Client Identity:
  - o The TOE does not support mutual authentication with regards to its HTTPS connections
  - o The TOE meets described behavior; no deviation

This aspect of the Cryptographic Support security function partially satisfies FCS_HTTPS_EXT.1.

5.2.5.5 NTP Protocol

The TOE can synchronize its system clock with up to 3 NTP servers. The administrator is able to configure the NTP time source(s) by specifying the hostname or IP address. The TOE supports NTP v4.2 as defined in RFC 5905 and uses SHA256 as its means for authenticating the NTP timestamps it receives from configured NTP servers. The TOE will not update NTP timestamps from broadcast or multicast addresses.

This aspect of the Cryptographic Support security function satisfies FCS_NTP_EXT.1.

5.3 Identification and Authentication

5.3.1 User Identification and Authentication

The TOE offers the following mechanisms for administrative access:

- Local access to the TOE's console (command line interface) via USB keyboard and VGA monitor
- Remote access to web management interface via HTTPS
- Remote access to RESTful API admin interface via HTTPS
- Remote access to command line interface via SSH

The TOE provides a local password-based authentication mechanism and authentication via SSH public key for administrators and will enforce an authentication result returned from an external LDAP server. LDAP authentication is supported for administrators accessing the Web UI.

During initial configuration, the TOE's default admin passwords (CLI users: root, service, vmxfer; Web UI users: root) must be changed to conform to the password policy requirements. The root admin and users with the role Account management can define additional administrators. Username/password credentials (or key for SSH) are necessary to log in and access the management functions. The local console connection and the remote Web UI/REST API access only allow password based connection. Passwords or keys can be used for the SSH connection. In addition to these local authentication methods, the TOE allows users to be defined in and authenticated by remote LDAP authentication servers. Users defined in LDAP authenticate using passwords.

When a password is used, the TOE first checks the validity of the authentication against the accounts defined locally within the TOE. If an account with the asserted identity is not found, the identity is checked against those defined in LDAP. If the identity is not found in either, user login is rejected. For a successful login, the identity and password must match those defined in either the local account or in LDAP. A key presented for SSH login must match that for the defined account in the TOE's database. If the asserted identity and password or key cannot be verified then the login fails and an audit record is generated.

The TOE does not allow any security-relevant actions prior to requiring the identification and authentication process except the display of the warning banner. Prior to identification and authentication, the TOE does allow the following non-security relevant activities: HTTPS ICMP ping requests, accept license terms. The availability of these services is not configurable. The TOE provides no feedback to the administrator during authentication other than echoing only '*' characters at the local console or remote interfaces when the administrator enters a password during the login process. A failed authentication attempt doesn't reveal anything about the nature of the failure.

For local password-based credentials, the TOE accepts passwords that consist of any combination of uppercase letters, lowercase letters, numbers, and special characters. The TOE supports characters: !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~ . For the web Ui the minimum password length can be configured from between 8 and 15 characters (default is 8 characters) and for CLI users, the value can be configured from 6 to 15 (8 is default). Password requirements are the same for all CLI users, regardless of whether they connect locally via console or remotely via SSH.

This aspect of the Identification and Authentication security function satisfies FIA_UIA_EXT.1, FIA_UAU_EXT.2, FIA_UAU.7, and FIA_PMG_EXT.1.

5.3.2 Authentication Failure Management

For password-based authentication, the administrator can configure the number of incorrect remote authentication attempts enforced on user accounts to a value between 1 and 50 (with the default being 20) for the web UI and between 1 and 65,535 (default 5) for remote CLI access. If an authentication attempt exceeds the threshold, the TOE will lockout the user until either a configured time-period has passed or the account is unlocked by an administrator. The admin can unlock Web UI/API users from the local console. CLI users can be unlocked from local or remote CLI.

The thresholds for the lockout of web UI users can be configured from the web UI by navigating to `Admin Users/Security->Account Lockout Failure Threshold`; and the thresholds for lockout of remote CLI users and be configured by editing the file: /etc/authselect/custom/with-restrictions/password-auth (using nano). Find the line starting with “auth required pam_faillock.so”dit deny=5 unlock_time=900 and changing the value of the “deny” parameter. The “unlock_time” parameter on this same line can be modified to configure the lockout time of a locked CLI user. The value is in seconds and can configured from 1 and 604800 (default is 15 minutes or 900 seconds). `Admin Users/Security->Account Lockout Duration (Minutes)` can be used to configure the lockout time of a locked web UI user. The setting can be configured from between 1 and 10,080 minutes (default is 30).

There can never be a case where all administrators are locked out because the TOE does not subject the root administrator accessing the CLI locally (who must directly connect to the TOE using USB) to lockouts. The root user can always login from the local CLI console. Authentication failure handling is only enforced on password credentials defined in the TOE. User authentication performed by LDAP or based on SSH private key are not subject to the lockout mechanism.

This aspect of the Identification and Authentication security function satisfies FIA_AFL.1.

5.3.3 X.509 Certificate Validation

The TOE performs revocation checking of certificates during TLS connection using the CRL as specified in RFC 5280 Section 6.3.

The TOE performs certificate validity and revocation checking on server certificates presented to it when connecting to an external syslog, API, LDAP or SMTP server; and when operating as the TLS client on a database channel. Additionally, the TOE supports TLS mutual authentication, performing certificate validation for KMIP communications; and Database Transaction and Replication channels, which use a master-servant arrangement where the servant operates as the TLS server.

The TOE performs the revocation checking after having checked the validity of the server certificate and its chain, conformant to RFC 5280. This includes supporting a minimum path length of three certificates

and the certification path terminating with a trusted CA certificate designated as a trust anchor. The TOE requires that TLS server certificates have the Server Authentication purpose and that TLS client certificates have the Client Authentication purpose in order to be considered valid. The TOE will not treat a CA certificate as such unless the basicConstraints extension is present with the CA flag set to TRUE.

Revocation checking is done by checking each CRL Distribution Point (CDP) extension in all incoming certificates in the chain. For each CDP extension found, the indicated path is used to check a local cache of CRLs. The CRLs in this cache are updated regularly, at the earlier of (a) every 24 hours, or (b) one hour before expiry of the next expiring CRL. If the CRL is not in the cache, it is downloaded from the HTTP endpoint in the CDP and added to the cache. The CRL update is automatic, there is no way to manually load a CRL. CRLs are downloaded based on any CRL Endpoint fields encoded in the TLS certificate presented by the remote server. Revocation checking is performed on all the cached CRLs available. If any CRL referenced by a certificate is not in the cache, or indicates that the certificate has been revoked, the connection is not established.

The TOE does not use certificates for trusted updates or executable code integrity verification and therefore the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) is not used in the extendedKeyUsage field. The TOE does not support OCSP and therefore those extensions are not checked.

This aspect of the Identification and Authentication security function satisfies FIA_X509_EXT.1/Rev.

5.3.4 X.509 Certificate Authentication

The TOE uses X.509 certificates for authentication of TLS and HTTPS trusted channels. The TOE relies upon the administrator to load the certificates for the root CA. The TOE validates X.509 certificates presented to it for all interfaces where the TOE acts as a TLS client, specifically:

- Remote syslog communications
- Remote LDAP communications
- Remote SMTP communications
- Outbound database replication communications (if the TOE's interaction with the operational environment is such that it is configured as a master node)
- Outbound registration for database replication (if the TOE's interaction with the operational environment is such that it is configured as a subordinate node)

The TOE also validates TLS client certificates where it acts as a TLS server that requires mutual authentication. Specifically, this applies to inbound KMIP communications and inbound database replication communications (if the TOE's interaction with the operational environment is such that it is configured as a subordinate node). In these cases, and for server interfaces that are not mutually authenticated (Apache server for remote administration and inbound registration for database replication), the TOE presents a server certificate to remote TLS clients.

Certificates presented for registration and database replication are automatically generated during the registration process, using signing CAs that are configured as being trusted. Certificates presented for all other interfaces use certificates signed by an external CA in response to a CSR generated by the TOE (see section 5.3.5 below). Trusted CA certificates are configured by a Security Administrator.

During revocation checking, if the TOEs TLS client/server cannot establish a connection to determine revocation status, the connection will not be established.

This aspect of the Identification and Authentication security function satisfies FIA_X509_EXT.2.

5.3.5 X.509 Certificate Requests

The TOE allows an administrator to request the TOE perform on-board key generation of 2048-bit RSA key pair sizes and then outputs a Certificate Signing Request (CSR), which the administrator can have signed by a suitable CA. The TOE generates Certificate Requests as specified by RFC 2986 and is able to provide the following information in the request: public key, Common Name, Organization, Organizational Unit and, Country.

This aspect of the Identification and Authentication security function satisfies FIA_X509_EXT.3.

5.4 Security Management

5.4.1 Security Roles and Specification of Management Functions

The TOE offers the following mechanisms for administrative access:

- Local access to TOE console (command line interface) via USB keyboard and VGA monitor
- Remote access to web management interface (web UI) via HTTPS
- Remote access to RESTful API admin interface via HTTPS
- Remote access to command line interface via SSH

Local management is available by plugging a keyboard into a USB port and a monitor into a VGA port, giving access to the TOE console, which provides a command line interface. The REST API and Web UI functions are only available remotely (over a network to a browser-equipped endpoint). The TOE has no graphical capability.

The TOE supports the following security relevant roles for Web UI or RESTful administrative users:

- Root—able to perform all management functions (including configuring the TOE's "Cluster" Trust Store)
- Account management—able to manage admin user accounts
- System management—able to manage various system aspects, such as logs, time, network configuration, web server configuration, replication, rebooting, and configuring the TOE's "Cluster" Trust Store
- Service management—able to manage services such as SSH and NTP

The TOE supports the following roles for CLI administrative users (whether via console or SSH):

- Root – able to perform all management functions, including configuring the Red Hat System-wide Trust Store but cannot log in remotely[7]
- Service
    - This user can:

[7] Since root access via ssh is disabled, there is no root level trusted public keys database (authorized_keys) and therefore no associated management function.

        - Log in remotely (or via console)
        - Manage their own trusted public keys database (for their SSH public key)
        - View status of running services, system time, network status
        - View system configuration settings
        - View audit information about user actions in the web UI
    - This user cannot:
        - View system audit logs in /var/log/messages
        - View SSH key material
- vmxfer user - has same privileges as service user.

The TOE provides the following management functions listed below. The interfaces at which these functions can be performed are also listed.

- Configure the access banner – Web UI, RESTful API
- Configure the session inactivity time before session termination – CLI (for CLI users), Web UI, RESTful API (for Web UI/RESTful API users)
- Update the TOE and verify TOE updates prior to installation using hash comparison – CLI
- Configure the authentication failure parameters – CLI (for CLI users), Web UI, RESTful API (for Web UI/RESTful API users)
- Ability to manage the cryptographic keys: (e.g. CSRs and TLS private key for web server) – CLI, Web UI, RESTful API, depending on key;
    - CLI used to clear TPM/HSM keys and to manage SSH trusted public keys and Red Hat System-wide Trust Store (see below)
    - Web UI used to enable CC Mode to enter the evaluated configuration (which also regenerates static keys) and to configure signed credentials for authentication to/from environmental instances of KMS (replication)
    - Web UI/RESTful API used to configure X.509 certificates and trust store to configure signed credentials for TLS communications to/from environmental instances of KMS (replication), X.509 server certificates for KMIP, and SSH backup server connections
- Ability to re-enable an Administrator account – CLI;
- Set the system time and timezone (used for time stamps) – Web UI, RESTful API
- Configure NTP – Web UI, RESTful API
- Configure Red Hat System-wide Trust Store[8] – CLI (requires root user role)
- Configure TOE's "Cluster" Trust Store [9] – Web UI, RESTful API (requires root or System management permissions)
- Manage the trusted public keys database – CLI (for remote admin public key authentication), Web UI, RESTful API (for outbound backup server authentication and host key configuration)

[8] defines Trusted CAs that may be used to authenticate SNMP or remote logging end points.

[9] defines which CAs are trusted by the cluster for the purposes of database transaction, replication and KMIP key distribution.

This aspect of the Security Management security function satisfies FMT_SMR.2 and FMT_SMF.1.

5.4.2 Management of Security Functions Behavior

The ability to perform TOE updates is restricted to the CLI Root user using role-based access control and performed via CLI only.

This aspect of the Security Management security function satisfies FMT_MOF.1/ManualUpdate.

5.4.3 Management of TSF Data

The TOE does not provide any access to management functions prior to authentication and restricts the ability to manage TOE data and cryptographic keys to Security Administrators using role-based access control methods. The TOE implements multiple administration roles that together provide the capabilities of the Security Administrator role defined in [cPPND]. The ability to manage the TOE data is restricted to these roles. The TOE includes two Trust Stores that can be managed by the root user: Red Hat System-wide Trust Store and the TOE's "Cluster" Trust Store. The TOE's "Cluster" Trust Store can also be managed by a user with System management permissions.

Specifically, for management of cryptographic keys, the Security Administrator may use the TOE to generate CSRs (which contain key pairs), configure key pairs for SSH key authentication, zeroize keys, and load certificates (whether it is certificate for the TOE generated by an external CA or a certificate used to validate a presented TLS client or server certificate) into the TOE's Trust Store. The Security Administrator can also manage the SSH host private key by entering Common Criteria mode.

The following table describes the key and credential management operations available to the administrators.

Table 10: Cryptographic Key Management

Function	Purpose	Administrator
Generate signed certificate authority and CSRs	Used to configure the intermediate CAs used to sign certificates used for database replication communications.	Root, System management
Management of CAs for replication connections	Used to configure the root CAs used to sign certificates used for database replication communications.	Root, System management
Import CA certificate	Used to configure the root CAs used to sign certificates used for database replication communications.	Root, System management
Configure internal credentials (qkm_server)/preparing for replication with remote or local CA's	Used for application layer authentication with an environmental instance of the KMS for replication.	Root, System management
Updating, deleting, importing credentials (Web Server,	Used for application layer authentication with an environmental instance of the KMS for replication or for remote administrators.	Root, System management

Replication, Client credentials)		
Generate CSR for server certificate	Used to obtain a TLS server certificate for the TOE's inbound TLS interfaces.	Root, System management
Generate locally signed credentials	Used for application layer authentication with an environmental instance of the KMS for replication.	Root, System management
Generate remotely signed credentials	Used for application layer authentication with an environmental instance of the KMS for replication.	Root, System management
Import a signed certificate	Used to obtain a TLS server certificate for the TOE's inbound TLS interfaces.	Root, System management
Delete a credential	Used for application layer authentication with an environmental instance of the KMS for replication.	Root, System management
Clearing the TPM key (destroys all private keys except for the factory root CA)	Used to wipe the TOE's stored data.	Root
Overwrite HSM key (destroys system root CA key and KMIP database encryption key)	Used to wipe the TOE's stored data.	Root
Enter Common Criteria mode – also rotates SSH host keys	Used to erase persistently stored keys loaded onto the TOE.	Root, System management
Manage trusted public keys database (remote SSH host public key)	Used to identify a known remote SSH host.	Root, System management, Service management
Manage trusted public keys database (outbound SSH client key pair)	Used to authenticate to a remote SSH host.	Root, System management, Service management
Manage trusted public keys database (remote SSH user public key, and SSH admin public/private key)	Used to authenticate a CLI administrator using SSH.	Root (can manage any keys), service (can manage their own key), vmxfer (can only manage their own key)

This aspect of the Security Management security function satisfies FMT_MTD.1/CoreData and FMT_MTD.1/CryptoKeys.

5.5 Protection of the TSF

5.5.1 Protection of Administrator Passwords

The TOE stores web administrative password data in a database as the output of the PBKDF2 derivation function with SHA-256 hash, a random 32-byte salt and 10,000 iterations. The TOE stores CLI passwords as the output of 400,000 rounds of SHA512 hashes (with 12-byte salt) in /etc/shadow.

The TOE does not provide interfaces for an administrator to view, extract, or read the password. The TOE only accepts passwords during authentication attempts (or during administrative changing of the password), salts and hashes the provided password (and then either compares it to the stored value or stores the new value depending upon whether the administrator is authenticating or changing the administrative password).

This aspect of the TSF Protection security function satisfies FPT_APW_EXT.1.

5.5.2 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys)

The TOE stores its private keys and Critical Security parameters (CSPs) as follows and does not provide any interfaces to view them:

- SSH private host keys, SSH client private keys, and TLS private keys for all communication channels except KMIP are stored encrypted with AES-256-XTS.
- TLS private keys for KMIP are stored on disk wrapped by the HSM using AES-256-CBC.
- The master HSM key is stored on the HSM encrypted using AES-256-CBC.
- The master TPM key and TPM seed value are stored in plaintext inside the server's TPM chip. They exist only within the TPM boundary and not are accessible to any user (no interface is provided to view).
- The TPM random secret is stored in plaintext on the hard drive. It is protected by file system permissions and is not accessible to any user.

This aspect of the TSF Protection security function satisfies FPT_SKP_EXT.1.

5.5.3 Reliable Time Stamps

The TOE uses time when creating audit records and when checking certificates presented to it (for expiration and revocation), for session inactivity timeout, and for administrator lock out periods. The TOE obtains and maintains time through configuration of up to 3 NTP v4 (RFC 5905) servers via the Web UI (`System > Time`) with which the TOE synchronizes its clock. The TOE also has a battery-backed real-time clock that is used to guarantee the availability of time data.

This aspect of the TSF Protection security function satisfies FPT_STM_EXT.1.

5.5.4 TSF Testing

During system boot and initialization, the TOE performs a series of tests to confirm correct operation of cryptographic components and the integrity of the software environment. Specific tests performed are:

1. Known-Answer Tests (KATs) for the OpenSSL library, with one KAT for each cryptographic algorithm used by the TOE
2. Startup tests for the built-in QRNG entropy source
3. HSM Operational Mode Check
4. Software integrity tests

The self-tests are sufficient to demonstrate that the TOE is operating correctly since they encompass the cryptographic functionality, the integrity of the entire TOE executable code, and detect hardware degradation that would lead to entropy failure providing assurance that cryptographic related hardware (QRNG and HSM) are functioning appropriately.

This section (and subsections) satisfies the TSF Testing security function: FPT_TST_EXT.1.

5.5.4.1 OpenSSL KATs

KATs are executed for the following cryptographic algorithms: SHA, AES, RSA, ECDSA, DH, ECDH, DRBG, and HMAC. For each self-test, the TOE uses known inputs to calculate an expected cryptographic result, and compares that result to the calculated result. If the calculated result matches the expected result, the test passes; if it does not match, the test fails. The result of each KAT is recorded in the audit log.

5.5.4.2 QRNG start-up tests

Startup tests are performed every time the QRNG Control Daemon initializes the card, before any output is available to the end-user. The results of these tests are written to the system log. The startup tests include:

- KATs for all CMAC, DRBG, and AES-ECB modules
- Continuous Health Check (positive and negative tests)
- Behavioral (Statistical) Tests: Repetition Count Test and Adaptive Proportion Test as specified in SP800-90B

During the startup phase, the Control Daemon puts the card into monitoring mode. This allows the QRNG card to run all raw samples through the Continuous Health Checks for a duration of three seconds without any commands being sent to the QRNG card. At the end of the period, the health check error flags are checked by the QRNG control daemon to ensure that the noise source functions as expected. During three seconds of operation, one can expect $500*10^6$ [samples/second] * 3 [seconds]=$15*10^8$ samples to have run through the Continuous Health Checks. The Continuous Health Checks are implemented in the FPGA firmware.

If any of these tests fail, the QRNG control software will enter an error state and will not provide entropy output. This error state will be recorded in the system log and reported via the web interface. A dashboard tile will go red if the QRNG is not operational - either if it is initializing, retraining or in an error state. This is the first thing the user sees when they log into the Web UI. They can then visit the QRNG status page in the web UI to see if this is an error state or part of the startup / retraining process. This status can also be queried via the REST API by an external monitoring system if required.

5.5.4.3 HSM Operational Mode Check

The CMVP #4765 validated HSM cryptographic module runs its own internal self-tests on power-up. The start-up self-tests include firmware integrity, library partition integrity and entropy source health tests.

Firmware integrity is verified using RSA with 4096 bit key and SHA2-256. Library partition integrity is verified using ECDSA with curve P-521 and SHA2-512. The HSM includes RSA and ECDSA digital signatures performed across its firmware and library partition, respectively. Upon powerup, integrity of the software/library partition is verified by calculating its hash and comparing it to the expected hash within the digital signature. If the hashes match, the signatures are valid, and the test passes.

The entropy source health tests are the SP 800-90B Adaptive Proportion Test and Repetition Count tests that are run on the output bits of the entropy source. The Repetition Count Test detects when the source gets "stuck" on one output for much longer than expected. An outline of test execution is as follows:

- Let H = entropy estimate for noise source
- P[max] = 2^{-H} is maximum probability for any value.
- Probability of seeing T identical values in a row <= P[max](T-1)
- Choose T so that P[max](T-1) is small enough that false positives aren't important (for example, 2-50).
- Memory requirements = previous sample value, counter of how many repetitions were seen

The Adaptive Proportion Test is designed to detect a large loss of entropy that might occur as a result of some physical failure or environmental change affecting the noise source. The test continuously measures the local frequency of occurrence of a sample value in a sequence of noise source samples to determine if the sample occurs too frequently. Thus, the test detects when some value begins to occur much more frequently than expected, given the source's assessed entropy per sample. The test takes a sample from the noise source, and then counts the number of times that the same value occurs within the next W-1 samples. If that count reaches the cutoff value C, the test declares an error.

If any of these tests fail, the HSM will not enter an operational state. The TOE will, as part of its startup procedure, verify that the HSM is in an operational state and therefore that the HSM internal self-tests passed. If the HSM self-tests fail, the TOE will still operate however, storage and retrieval of managed objects will not be possible as the DEK (Database Encryption Key) will be unavailable. The TOE will log the result of this check (either for success or failure cases). The log can be viewed by the root administrator using the command hsmadmin logs get --log system --esn 1875-5D70-55B2. The esn can be found with the command: /opt/nfast/bin/nfkminfo | grep esn.

In addition, the web UI dashboard, console and ssh login screens display a warning message to report when the HSM is in an error state.

5.5.4.4 Software Integrity tests

The system will run integrity checks on boot, checking that the SHA256 hash of each system file matches an expected value. Each TOE file is placed in one of three categories: *WebUI, SSH* and *General.* The files represent the program code and library files that implement TOE security functions. When integrity checks pass, these files have not been modified from their expected states, set at time of install / trusted update.

- An integrity failure of the *Web UI* files will display a warning banner via SSH, disable the web server and prevent access to the web UI.
- An integrity failure of *General* or *Web UI* files will present a warning banner via SSH – user can still log in via SSH or localhost.
- An integrity failure of *SSH* files will disable the SSH server. A warning message will be presented via the login page of the Web UI.

Integrity test failures result in the following error messages, and audit record generation.

- integrity failure of the Web UI files displays the error message, "WARNING: test INTEGRITY_WEBUI failed" at the SSH interface.
- Integrity failure of the SSH files displays the error message, "WARNING: test INTEGRITY_SSH failed" at the Web UI.
- Integrity failure of the General files displays the error message, "WARNING: test INTEGRITY_GENERAL failed" at the SSH interface and at the Web UI.

Guidance documentation is provided to administrators with instructions on how to determine which specific file failed the integrity test as well as actions they should take upon receiving an error message. Audit records are also generated for successful self-tests. The failed self-test banners are independent of the customizable log-in banner.

5.5.5 Trusted Update

The TOE provides an administrator the ability to view the currently executing software version by clicking on the **account** icon at the top-right of any page in the web UI to show the available options and then selecting the **About System** item. The firmware version for the QRNG card can be accessed via Web UI, Entropy -> Qrng Device; and the firmware version for HSM can be accessed via the CLI, using command /opt/nfast/bin/enquiry and looking for the 'version string' element in the output. The TOE does not display other software/firmware versions since it does not support a mechanism for a delayed activation.

The TOE supports manual trusted updates for its software, firmware, and third-party components included in the TOE. The TOE update is a single self-contained package with the update script and all required update packages.

Qlabs obtains the third-party components updates via official package management repositories (RHEL, EPEL (Extra Packages for Enterprise Linux), PyPl (Python), Remi, Rsyslog, PostgreSQL) and verifies their integrity as follows. All updates are obtained over TLS. Updates from RHEL and EPEL repositories are retrieved with the 'sslverify' flag set to enforce that the upstream certificate is signed by a trusted CA. Every package in the repository is signed with a GPG key owned by RHEL. Before the package is installed on a TOE, this signature is validated against the public key published by Red Hat. Packages from PyPl repositories are retrieved with the upstream certificate signed by a trusted CA. Qlabs verifies their SHA256 checksum matches the expected value in their metadata file. Most of the python packages are flagged as 'Trusted' packages by PyPI, which means they have been uploaded to PyPI via a trusted source. Packages from Remi, Rsyslog and Postgre are retrieved with the upstream certificate signed by a trusted CA. QLabs verifies that each package in the repository is signed with a GPG key by the package owner. Before the package is installed on a TOE, QLabs validates this signature against the public key provided by the package maintainer.

QLabs copies the validated third-party package updates into an install bundle, along with any updates to QLabs proprietary software, including the rpm-gpg keys The integrity of the entire TOE update package is protected and verified through the use of a SHA-256 published hash. When QuintessenceLabs support

announces a new update, the announcement will include a SHA-256 checksum that can be used to validate updates obtained from the QuintessenceLabs download server.

Updates can be manually initiated from the CLI. The update files must first be obtained by the administrator from QuintessenceLabs' support portal. The customer contacts their QuintessenceLabs support representative via email or support ticket, and requests access to the upgrade. The QuintessenceLabs support team makes the required version available on a secure download server. The customer logs into the secure download server (https://download.quintessencelabs.com/files/portal/login.jsp) using their email and password, which was set up ahead of time by QuintessenceLabs support. The user can find the required upgrade files, and verify that the checksum displayed next to the file name matches the checksum they received from QuintessenceLabs support via email or their support ticket.

Once downloaded, the administrator must copy (using scp) the archive onto the appliance. They must then log into the appliance using SSH (or direct console access) and elevate to the root user. They must then decompress the update, and execute the update script to install the update.

The admin can determine whether the update attempt was successful or not by examining the output of the command. The update will print (and will write into the audit log) a message indicating that the update succeeded. If the update fails, a message indicating this will be displayed to the user. The history of successfully applied updates can be reviewed via the Web interface: log in to the web interface, and click on the username at the top-right of the interface, next to the *help* menu. This will display a drop-down menu – select *About System*. This will navigate to a page that displays the current software version and a list of applied patches.

The TOE does not provide an automated update mechanism.

This aspect of the TSF Protection security function satisfies FPT_TUD_EXT.1.

5.6 TOE Access

5.6.1 Session Termination

The TOE terminates inactive sessions when the administrator configurable inactive timeout value is reached. The TOE provides two mechanisms, one for the CLI, which can be accessed locally via the console or remotely via SSH, and a second for the Web UI. Session timeout for the CLI is configured using the CLI. The new value will take effect for new sessions. The default is 900 seconds and can be configured to values between 1 and 2^31 seconds. The TOE does not provide the capability to disable the session timeout function. For the web server, the inactive time is configured using the Web UI, and can be configured for between 10 and 86400 seconds (300 seconds is the default). The configuration will take effect during the next administrative session.

The Administrator can disconnect their administrative session from the CLI using the **`EXIT`** command that will logout the user. From the web UI, the administrator can click on their username in the top right and select logout.

This aspect of the TOE Access security function satisfies FTA_SSL_EXT.1, FTA_SSL.3, and FTA_SSL.4.

5.6.2 Access Banner

The TOE provides the administrator the ability to configure a banner that the TOE displays before each local and remote administrator login. Administrators' login to the TOE remotely through the TOE's SSH or

TLS-protected management ports, or locally via direct connection to the TOE's USB ports. The banner can be configured via SSH / CLI or via Web UI. The banner is configured via the CLI by editing the file: /etc/issue.d/qlabs.issue or via the web UI by navigating to `System -> Node Settings`, edit the Message field and click the Save button. Note that regardless of where the configuration is performed, the same banner is displayed for CLI, SSH and Web UI users.

This aspect of the TOE Access security function satisfies FTA_TAB.1.

5.7 Trusted Path/Channels

The TOE protects communications with authorized external IT entities using TLS, SSH and HTTPS for the following purposes:

- providing key management services with external KMIP clients (mutually authenticated TLS, TOE acts as server),
- database transaction for communicating database information to environment, TOE can be client or server (mutually authenticated TLS, TOE acts as client for pushing data to operational environment or as a server for receiving data from the environment),
- API replication for setup of environmental database transaction connectivity, TOE can be client or server (HTTPS, TOE acts as a client for registering a database channel to the operational environment, TOE acts as server for receiving registration connection from environment)[10],
- LDAP authentication for authenticating remote administrators (TLS client, TOE acts as client),
- sending email notifications via SMTP (TLS client, TOE acts as server),
- export of audit records to external syslog server (TLS Client, TOE acts as client),
- export a TOE backup to an external backup server (SSH, TOE acts as client)

The administrator can connect to the TOE using its HTTPS (Web UI, Rest API) or SSH protected administration channel.

The Trusted Path/Channels security function satisfies FTP_ITC.1 and FTP_TRP.1/Admin.

[10] The workflow for database replication is such that each subordinate node registers to the master node, and once registered, the master node transmits database replication information to the subordinate node. Whether the TOE first acts as the client followed by the server, or vice versa, is dependent on whether the TOE is transmitting database information to the environment or receiving it.

6 Protection Profile Claims

This ST conforms to the collaborative Protection Profile for Network Devices, Version 2.2e, 23 March 2020 [cPPND] and including the following optional and selection-based SFRs: FAU_STG.1, FCS_HTTPS_EXT.1; FCS_NTP_EXT.1; FCS_SSHC_EXT.1; FCS_SSHS_EXT.1; FCS_TLSC_EXT.1; FCS_TLSS_EXT.1; FCS_TLSC_EXT.2; FCS_TLSS_EXT.2; FIA_X509_EXT.1/Rev; FIA_X509_EXT.2; FIA_X509_EXT.3; and FMT_MTD.1/CryptoKeys.

As explained in Section 2, Security Problem Definition, the Security Problem Definition of the [cPPND] has been included by reference into this ST. The TOE is not a distributed TOE and does not have a virtual network device configuration, therefore the corresponding assumptions: A.COMPONENTS_RUNNING, A.VS_TRUSTED_ADMINISTRATOR, A.VS_REGULAR_UPDATES, A.VS_ISOLATION, and A.VS_CORRECT_CONFIGURATION do not apply to the TOE,

As explained in Section 3, Security Objectives, the ST reproduces the security objectives for the operational environment from [cPPND]. The TOE is not a distributed TOE and does not have a virtual network device configuration, therefore the corresponding objectives: OE.COMPONENTS_RUNNING and OE.VM_CONFIGURATION are not applicable.

As explained in Section 4, IT Security Requirements, the SFRs have all been drawn from [cPPND]. As such, operations on SFRs already performed in that PP are not identified in this ST. Rather; the SFRs have been copied from [cPPND] and any formatting used in that PP has been removed. Operations performed on SFRs in the writing of this ST are identified in accordance with the conventions described in Section 1.3.

The SARs for the TOE are included by reference from the [cPPND].

7 Rationale

This ST includes by reference the [cPPND] Security Problem Definition and SARs and reproduces the security objectives for the Operational Environment. The ST makes no additions to the [cPPND] assumptions. [cPPND] SFRs have been reproduced with the Protection Profile operations completed. Operations on the SFRs follow [cPPND] application notes and assurance activities. Consequently, [cPPND] rationale applies but is incomplete. The TOE Summary Specification rationale below serves to complete the rationale required for the security target.

7.1 TOE Summary Specification Rationale

Each subsection in Section 5, the TOE Summary Specification (TSS), describes a security function of the TOE. Each description identifies the SFRs that are covered by that description and, as such, provides the rationale that indicates which requirements are satisfied by aspects of the corresponding security function. The security functions work together to satisfy all of the security functional requirements. Furthermore, all of the security functions are necessary in order for the TOE to provide the required security functionality.

This section, in conjunction with the TSS, provides evidence that the security functions are suitable to meet the TOE security requirements. The collection of security functions work together to provide all of the security requirements. The security functions described in the TSS are all necessary for the required security functionality in the TOE. Table 11: Security Functions vs. Requirements Mapping summarizes the relationship between security requirements and security functions.

Table 11: Security Functions vs. Requirements Mapping

	Security audit	Cryptographic support	Identification and authentication	Security management	Protection of the TSF	TOE access	Trusted path/channels
FAU_GEN.1	X						
FAU_GEN.2	X						
FAU_STG.1	X						
FAU_STG_EXT.1	X						
FCS_CKM.1		X					
FCS_CKM.2		X					
FCS_CKM.4		X					
FCS_COP.1/DataEncryption		X					
FCS_COP.1/SigGen		X					
FCS_COP.1/Hash		X					
FCS_HTTPS_EXT.1		X					
FCS_COP.1/KeyedHash		X					

	Security audit	Cryptographic support	Identification and authentication	Security management	Protection of the TSF	TOE access	Trusted path/channels
FCS_NTP_EXT.1		X					
FCS_RBG_EXT.1/OpenSSH		X					
FCS_RBG_EXT.1/HSM		X					
FCS_SSHC_EXT.1		X					
FCS_SSHS_EXT.1		X					
FCS_TLSC_EXT.1		X					
FCS_TLSC_EXT.2		X					
FCS_TLSS_EXT.1/Apache		X					
FCS_TLSS_EXT.1/R		X					
FCS_TLSS_EXT.2		X					
FIA_AFL.1			X				
FIA_PMG_EXT.1			X				
FIA_UIA_EXT.1			X				
FIA_UAU_EXT.2			X				
FIA_UAU.7			X				
FIA_X509_EXT.1/Rev			X				
FIA_X509_EXT.2			X				
FIA_X509_EXT.3			X				
FMT_MOF.1/ManualUpdate				X			
FMT_MTD.1/CoreData				X			
FMT_SMF.1				X			
FMT_SMR.2				X			
FPT_APW_EXT.1					X		
FPT_SKP_EXT.1					X		
FPT_TST_EXT.1					X		
FPT_TUD_EXT.1					X		
FPT_STM_EXT.1					X		
FTA_SSL_EXT.1						X	
FTA_SSL.3						X	
FTA_SSL.4						X	
FTA_TAB.1						X	

	Security audit	Cryptographic support	Identification and authentication	Security management	Protection of the TSF	TOE access	Trusted path/channels
FTP_ITC.1							X
FTP_TRP.1/Admin							X