DRAFT Protection Profile for General-Purpose Computing Platforms DRAFT

Version: 2.0
2026-03-13
National Information Assurance Partnership

Revision History

Version	Date	Comment
0.1	2020-11-16	Started
1.0	2022-02-10	Initial publication.
1.1	2024-12-13	Updates for CC:2022
2.0	2025-01-13	Draft: Incorporate TRRTs
2.0	2026-03-13	Draft: New draft with revisions from contributors

Contents

1 Introduction
1.1 Overview
1.2 Terms
1.2.1 Common Criteria Terms
1.2.2 Technical Terms
1.3 TOE Overview
1.3.1 TOE Boundary
1.3.2 TOE Operational Environment
1.4 Use Cases
1.5 Package Usage
1.6 Roles
2 Conformance Claims
3 Security Problem Definition
3.1 Threats
3.2 Assumptions
3.3 Organizational Security Policies
4 Security Objectives
4.1 Security Objectives for the Operational Environment
4.2 Security Objectives Rationale
5 Security Requirements
5.1 Security Functional Requirements
5.1.1 Auditable Events for Mandatory SFRs
5.1.2 Class: Security Management (FMT)
5.1.3 Class: Protection of the TSF (FPT)
5.1.4 TOE Security Functional Requirements Rationale
5.2 Security Assurance Requirements
5.2.1 Class ASE: Security Target
5.2.2 Class ADV: Development
5.2.3 Class AGD: Guidance Documentation
5.2.4 Class ALC: Life-cycle Support
5.2.5 Class ATE: Tests
5.2.6 Class AVA: Vulnerability Assessment
Appendix A - Optional Requirements
A.1 Strictly Optional Requirements
A.1.1 Auditable Events for Strictly Optional Requirements
A.1.2 Class ALC: Life-cycle Support
A.1.3 Class: Cryptographic Support (FCS)
A.1.4 Class: User Data Protection (FDP)
A.1.5 Class: Identification and Authentication (FIA)
A.2 Objective Requirements
A.2.1 Auditable Events for Objective Requirements
A.2.2 Class: Protection of the TSF (FPT)
A.3 Implementation-dependent Requirements
Appendix B - Selection-based Requirements
B.1 Auditable Events for Selection-based Requirements
B.2 Class: Security Audit (FAU)
B.3 Class: Cryptographic Support (FCS)

B.4 Class: User Data Protection (FDP)
B.5 Class: Identification and Authentication (FIA)
B.6 Class: Protection of the TSF (FPT)
B.7 Class: Trusted Path/Channels (FTP)
Appendix C - Extended Component Definitions
C.1 Extended Components Table
C.2 Extended Component Definitions
C.2.1 Class: Cryptographic Support (FCS)
C.2.1.1 FCS_CKM_EXT Cryptographic Key Management
C.2.1.2 FCS_HTTPS_EXT HTTPS Protocol
C.2.1.3 FCS_IPSEC_EXT IPsec Protocol
C.2.1.4 FCS_STG_EXT Cryptographic Key Storage
C.2.2 Class: Identification and Authentication (FIA)
C.2.2.1 FIA_AFL_EXT Authentication Failure Handling
C.2.2.2 FIA_PMG_EXT Password Management
C.2.2.3 FIA_TRT_EXT Authentication Throttling
C.2.2.4 FIA_UIA_EXT Administrator Identification and Authentication
C.2.3 Class: Protection of the TSF (FPT)
C.2.3.1 FPT_JTA_EXT Debug Port Access
C.2.3.2 FPT_ROT_EXT Platform Integrity
C.2.3.3 FPT_PPF_EXT Protection of Platform Firmware
C.2.3.4 FPT_RVR_EXT Platform Firmware Recovery
C.2.3.5 FPT_TUD_EXT Platform Firmware Update
C.2.4 Class: Security Management (FMT)
C.2.4.1 FMT_CFG_EXT Secure by Default
C.2.5 Class: Trusted Path/Channels (FTP)
C.2.5.1 FTP_ITC_EXT Trusted Channel Communications
C.2.5.2 FTP_ITE_EXT Encrypted Data Communications
C.2.5.3 FTP_ITP_EXT Physically Protected Channel
C.2.6 Class: User Data Protection (FDP)
C.2.6.1 FDP_ITC_EXT Key Import
C.2.6.2 FDP_TEE_EXT Trusted Execution Environment
Appendix D - Implicitly Satisfied Requirements
Appendix E - Entropy Documentation and Assessment
E.1 Design Description
E.2 Entropy Justification
E.3 Operating Conditions
E.4 Health Testing
Appendix F - Equivalency Guidelines
F.1 Introduction
F.2 Approach to Equivalency Analysis
F.3 Specific Guidance for Determining Product Equivalence
F.4 Technical Equivalence
F.5 Level of Specificity for Tested and Claimed Equivalent Configurations
Appendix G - Use Case Templates
G.1 Server-Class Platform, Physically Secure Environment
G.2 Server-Class Platform, Enhanced Security Requirements
G.3 Portable Clients (laptops, tablets), Enhanced Security Requirements
G.4 CSfC EUD
G.5 Tactical EUD
G.6 Enterprise Desktop Clients
Appendix H - Acronyms
Appendix I - Bibliography

1 Introduction

1.1 Overview

The scope of this Protection Profile (PP) is to describe the security functionality of General-Purpose Computing Platforms in terms of the Common Criteria and to define functional and assurance requirements for such products.

A platform is a collection of hardware devices and firmware that provide the functional capabilities and services needed by tenant software. Such components typically include embedded controllers, trusted platform modules, management controllers, host processors, network interface controllers, graphics processing units, flash memory, storage controllers, storage devices, boot firmware, runtime firmware, human interface devices, and a power supply.

This Protection Profile for General-Purpose Computing Platforms derives requirements from the following documents:

- NIST SP 800-147 *BIOS Protection Guidelines*, April 2011
- NIST SP 800-147B *BIOS Protection Guidelines for Servers*, August 2014
- NIST SP 800-193 *Platform Firmware Resiliency Guidelines*, May 2018

Additionally, the following specifications and standards may be relevant to requirements in this PP:

- NIST SP 800-155 (Draft) *BIOS Integrity Measurement Guidelines (Draft)*, December 2011
- NIST SP 1800-34 *Validating the Integrity of Computing Devices*, December 2022
- Trusted Computing Group, *TCG PC Client Platform Firmware Integrity Measurement* Version 1.0 Revision Specification 43 Family 2.0, May 7, 2021
- IEEE Std 802.1AR-2018, *Secure Device Identity*

1.2 Terms

The following sections list Common Criteria and technology terms used in this document.

1.2.1 Common Criteria Terms

Assurance	Grounds for confidence that a TOE meets the SFRs [CC].
Base Protection Profile (Base-PP)	Protection Profile used as a basis to build a PP-Configuration.
Collaborative Protection Profile (cPP)	A Protection Profile developed by international technical communities and approved by multiple schemes.
Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation (International Standard ISO/IEC 15408).
Common Criteria Testing Laboratory	Within the context of the Common Criteria Evaluation and Validation Scheme (CCEVS), an IT security evaluation facility accredited by the National Voluntary Laboratory Accreditation Program (NVLAP) and approved by the NIAP Validation Body to conduct Common Criteria-based evaluations.
Common Evaluation Methodology (CEM)	Common Evaluation Methodology for Information Technology Security Evaluation.
Direct Rationale	A type of Protection Profile, PP-Module, or Security Target in which the security problem definition (SPD) elements are mapped directly to the SFRs and possibly to the security objectives for the operational environment. There are no security objectives for the TOE.
Extended Package (EP)	A deprecated document form for collecting SFRs that implement a particular protocol, technology, or functionality. See Functional Packages.
Functional Package (FP)	A document that collects SFRs for a particular protocol, technology, or functionality.
Operational Environment (OE)	Hardware and software that are outside the TOE boundary that support the TOE functionality and security policy.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.
Protection Profile Configuration (PP-	A comprehensive set of security requirements for a product type that consists of at least one Base-PP and at least one PP-Module.

Configuration)	
Protection Profile Module (PP-Module)	An implementation-independent statement of security needs for a TOE type complementary to one or more Base-PPs.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in an ST.

1.2.2 Technical Terms

Administrator	An Administrator is responsible for management activities, including setting policies that are applied by the enterprise on the platform. An Administrator can act remotely through a management server, from which the platform receives configuration policies and updates. An Administrator can enforce settings on the system that cannot be overridden by non-Administrator users.
American National Standards Institute (ANSI)	A private organization that oversees development of standards in the United States.
Application	Software that runs on a platform and performs tasks on behalf of the user or owner of the platform.
Application Programming Interface (API)	A specification of routines, data structures, object classes, and variables that allows an application to make use of services provided by another software component, such as a library. APIs are often provided for a set of libraries included with the platform.
Baseboard Management Controller (BMC)	Or Management Controller. A small computer generally found on Server motherboards that performs management tasks on behalf of an Administrator.
Bill of Materials (BOM)	A manifest document containing identifying information about a piece of hardware, firmware, or software along with measurements (usually hashes) describing what the vendor intended for the client to receive. The BOM should be signed. BOMs may accompany updates or product revisions to describe intended changes.
Cipher-based Message Authentication Code (CMAC)	A mode of AES that provides authentication, but not confidentiality.
Commercial Solutions for Classified (CSfC)	An US Department of Defense program for delivering cybersecurity solutions that leverage commercial technologies and products.
Credential	Data that establishes the identity of a user, e.g. a cryptographic key or password.
Critical Security Parameters (CSP)	Information that is either user or system defined and is used to operate a cryptographic module in processing encryption functions including cryptographic keys and authentication data, such as passwords, the disclosure or modification of which can compromise the security of a cryptographic module or the security of the information protected by the module.
Data-at-Rest (DAR) Protection	Countermeasures that prevent attackers, even those with physical access, from extracting data from non-volatile storage. Common techniques include data encryption and wiping.
Developer	An entity that manufactures platform hardware or writes platform software/firmware. For the purposes of this document, vendors and developers are the same.
Diffie-Hellman Key Exchange (DH)	A cryptographic key exchange protocol using public/private key pairs.

Distinguished Name (DN)	Information used in certificate-based operations to uniquely identify a person, organization, or business.
End-User Device (EUD)	A class of computing platform characterized by having a user interface for a single user. Often, EUDs are portable (e.g., laptop, tablet, mobile device), but this is not necessarily the case (e.g., desktop PC).
General Purpose Operating System	A class of OS designed to support a wide-variety of workloads consisting of many concurrent applications or services. Typical characteristics for OSes in this class include support for third-party applications, support for multiple users, and security separation between users and their respective resources.
General-Purpose Computing Platform (GPCP)	A physical computing platform designed to support general-purpose operating systems, virtualization systems, and applications.
Internet of Things (IoT)	Physical computing devices that are embedded with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over communications networks.
Joint Test Action Group (JTAG)	A standard for verifying and testing circuit boards after manufacture.
KECCAK Message Authentication Code (KMAC)	A variable-length keyed hash function described in NIST SP 800-185.
Management Controller (MC)	Or Baseboard Management Controller (BMC). A small computer generally found on server motherboards that performs management tasks on behalf of an Administrator.
Open Mobile Terminal Platform (OMTP)	A forum created by mobile network operators to discuss standards with manufacturers of mobile devices.
Operating System (OS)	Software that manages physical and logical resources and provides services for applications. Operating systems are the generally the primary tenant of a GPCP.
Physical Presence	A user or administrator having physical access to the TOE. An assertion of physical presence can take the form, for example, of requiring entry of a password at a boot screen, unlocking of a physical lock (e.g., a motherboard jumper), or inserting a USB cable before permitting platform firmware to be updated.
Root of Trust (RoT)	Roots of trust are highly reliable hardware, firmware, and software components that perform specific, critical security functions. Roots of trust are the foundation for integrity of computing devices.
Sensitive Data	Sensitive data may include all user or enterprise data or may be specific application data such as PII, emails, messaging, documents, calendar items, and contacts. Sensitive data must minimally include credentials and keys.
Subject Alternative Name (SAN)	An extended X.509 certificate field.
Tenant Software	Software that runs on and is supported by a platform. In the case of a GPCP, tenant software generally consists of an operating system, virtualization system, or "bare-metal" application.
Trusted Execution Environment (TEE)	An isolated and secure area that ensures the confidentiality and integrity of code and data loaded inside.
Trusted Platform Module (TPM)	A security processor that provides services for cryptographic key management, encryption and decryption, platform integrity management, and device identity. TPMs that are discrete/dedicated chips (dTPM) or firmware/integrated with another component such as the CPU (fTPM) are within the scope of a GPCP TOE. Virtual/software TPMs (vTPM) are out of scope.
User	In the context of a GPCP, a User is a human who interacts with the platform through a user interface. Users do not need to be authenticated by the platform to use the platform, but generally authenticate to tenant software such as on Operating System.
Virtualization System (VS)	A software product that enables multiple independent computing systems to execute on the same physical hardware platform without interference from one other.

1.3 TOE Overview

This Protection Profile for General-Purpose Computing Platforms (GPCP) specifies security requirements for general-purpose computing platforms. A GPCP is is a hardware device that is capable of hosting one or more general-purpose operating systems as defined by the Protection Profile for General Purpose Operating Systems, one or more virtualization systems as defined by the Protection Profile for Virtualization, or more than one application. Typical platform implementations include servers, PC clients, laptops, and tablets.

This Protection Profile applies only to platforms that support firmware update.

Mobile Device platforms as defined in the Protection Profile for Mobile Device Fundamentals and Network Device platforms as defined in the collaborative Protection Profile for Network Devices are out of scope of this PP. Mobile Device and Network Device platforms must be evaluated against the more specific requirements in their respective specialized PPs.

Likewise, hardcopy devices such as printers, scanners, copiers, and fax machines are out of scope of this Protection Profile and should be evaluated instead against the Protection Profile for Hardcopy Devices.

Finally, platforms for sharing and isolation of peripheral devices across domains, such as KVM Switches and Isolators, should be evaluated against the Protection Profile for Peripheral Sharing Devices.

The core security features of GPCPs include protected firmware and a boot integrity processes. Platform firmware must be protected such that it is not permitted to execute if it has been modified outside of authorized and authenticated update processes. Other use-case-specific features include audit capabilities, Administrator authentication, and protections against physical tampering.

1.3.1 TOE Boundary

The TOE comprises the hardware and firmware necessary for the hosting of tenant software. Generally, tenant software is an operating system or virtualization system, but may also be "bare-metal" applications. Tenant software is outside the TOE boundary.

For example, for a PC Client platform, the hardware and firmware responsible for booting the platform and operation of platform devices (such as BIOS, device controller firmware, and platform management firmware) would all be included in the TOE. Operating systems and application software is outside the TOE.

For server-class hardware, any management controller responsible for updating platform firmware (such as a baseboard management controller) is expressly included within the TOE.

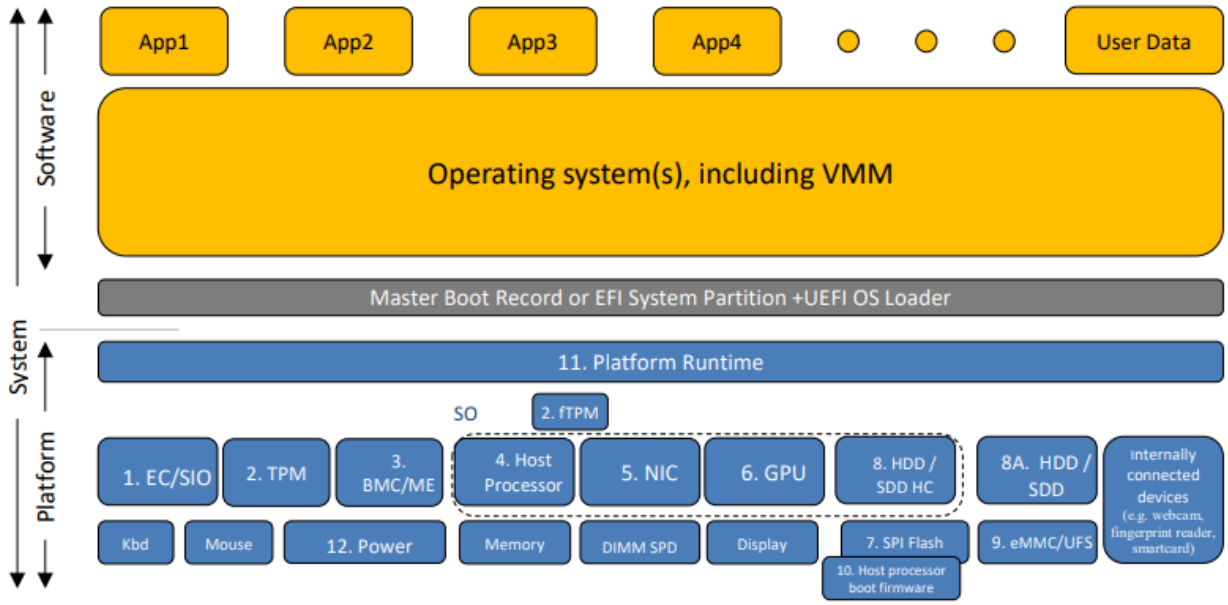


Figure 1: High-Level Architecture of a Generic Platform

Figure 1 (taken from NIST SP 800-193) shows a high-level system architecture for a typical generic computing platform. Tenant software (operating system/virtualization system and applications) is shown in orange. The tenant-specific software responsible for booting the tenant (Master Boot Record, etc.) is shown in grey. Platform components are in blue.

In general, the TOE consists of the platform components represented by the blue boxes, along with their associated firmware. Any particular platform may have additional hardware components, or fewer than those illustrated.

If the GPCP includes Full Drive Encryption (FDE), and the FDE component has been previously evaluated against the FDE cPPs, or will be evaluated against the FDE cPPs concurrently with the GPCP evaluation, then the FDE component may be excluded from the TOE for purposes of the GPCP evaluation.

If the GPCP includes a Dedicated Security Component (DSC) that has been previously evaluated against the DSC cPP, or will be evaluated against the DSC cPP concurrently with the GPCP evaluation, then the DSC component may be excluded from the TOE for purposes of the GPCP evaluation.

1.3.2 TOE Operational Environment

The TOE has no platform since it is itself a platform, but the TOE does have an operational environment. The OE consists of the physical environment in which the TOE operates (e.g., data center, enterprise office, vehicle, outdoors) and any networks to which the TOE may be connected. Different use cases may invoke different requirements depending on the operational environment.

1.4 Use Cases

This Protection Profile supports several use cases. The cases enumerated below add requirements to the baseline for GPCP in response to the needs of the use cases and in differences in the expected operational environments. Use cases not listed below (e.g. consumer-grade desktop, laptop, and tablet computers) need be evaluated only against the baseline requirements subject to the appropriate selections.

The requirements associated with one use case encompasses those of another use case. In these situations, a TOE that meets the larger set of requirements meets both use cases. Specifically

- a TOE that meets [USE CASE 2] Server-Class Platform, Enhanced Security Requirements also meets [USE CASE 1] Server-Class Platform, Physically Secure Environment

[USE CASE 1] Server-Class Platform, Physically Secure Environment

This use case encompasses server-class hardware in a data center. There are no additional physical protections required because the platform is assumed to be protected by the operational environment as indicated by A.PHYSICAL_PROTECTION. The platform is administered using a management controller that is accessed remotely or through a console.

This use case adds audit requirements and Administrator authentication requirements to the base mandatory requirements.

For changes to included SFRs, selections, and assignments required for this use case, see G.1 Server-Class Platform, Physically Secure Environment.

[USE CASE 2] Server-Class Platform, Enhanced Security Requirements

This use case adds physical protections to the base requirements for server-class hardware. Additional physical protections are required because the platform is assumed to not be protected by the operational environment, or because of enhanced security requirements imposed by data center or operational policies.

This use case adds requirements for audit, physical protections, and Administrator authentication to the base mandatory SFRs.

For changes to included SFRs, selections, and assignments required for this use case, see G.2 Server-Class Platform, Enhanced Security Requirements.

[USE CASE 3] Portable Clients (laptops, tablets), Enhanced Security Requirements

This use case adds physical protections to the base requirements for portable clients or end-user devices. It is intended for devices are used in high-assurance scenarios.

For changes to included SFRs, selections, and assignments required for this use case, see G.3 Portable Clients (laptops, tablets), Enhanced Security Requirements.

[USE CASE 4] CSfC EUD

EUDs used in accordance with the CSfC Mobile Access Capability Package can include smart phones, tablets, desktops, and laptops. This use case covers the basic CSfC requirements for EUDs other than mobile devices (mobile devices are out of scope for this PP).

CSfC requires that users maintain physical control of EUDs at all times. This use case adds requirements for audit and for protection of debug ports .

The CSfC Use Case requires that End User Devices prohibit the use of removable media either through configuration, policy, or physical modification.

For changes to included SFRs, selections, and assignments required for this use case, see G.4 CSfC EUD.

[USE CASE 5] Tactical EUD

This use case adds requirements for portable end user computing devices in a tactical environment.

For changes to included SFRs, selections, and assignments required for this use case, see G.5 Tactical EUD.

[USE CASE 6] Enterprise Desktop Clients

This use case covers the requirements for non-portable desktop computing devices in a low-threat enterprise physical environment.

This use case adds only audit to the base mandatory SFRs.

For changes to included SFRs, selections, and assignments required for this use case, see G.6 Enterprise Desktop Clients.

1.5 Package Usage

This section contains selections and assignments that are required when the listed Functional Packages are claimed by this PP.

Functional Package for X.509, Version 1.0

Limitations on Signature Algorithms in FIA_X509_EXT.1.1

The TOE must utilize appropriate cryptographic algorithms that conform to CNSA standards. Thus, the TOE shall utilize no other algorithms outside of those specified in RFC 8603 for certificate validation and CRL signature validation (if claimed). If OCSP is

claimed, the ST author shall only claim support for algorithms that use SHA-384 and either 3072-bit RSA or greater, or NIST P-384. In all cases, the assignment that allows the ST author to specify additional algorithms shall not be selected.

CRL or OCSP-based Revocation is Optional With Justification

The TOE must identify where/when CRL or OCSP-based revocation is infeasible and not supported. Examples of such situations include UEFI Secure Boot enforcement at device boot time or administrator interaction with a Baseboard Management Controller (BMC) where resources are severely constrained or internet access is unavailable. Use of CRL or OCSP-based revocation is not required when justified as infeasible.

Restrictions on Acceptable Key Usage Values for FIA_X509_EXT.1.5

The TOE will always support the use of extendedKeyUsage values to verify that X.509 certificates are used in accordance with their intended purpose. Accordingly, the ST author shall claim that the TOE supports the processing of extendedKeyUsage fields in the leaf certificate (as opposed to application of trust store context rules or passing the certification path or other supported context information to an external function) and shall select all values that are relevant to the claimed uses of X.509 in the ST. In particular, since the PP does not define any functions that require the use of S/MIME, the ST author shall not select this as an extendedKeyUsage value to be validated.

Restrictions on Acceptable Functions for FIA_X509_EXT.2.1

The PP does not define SMIME functionality, therefore the ST shall not select this as a function for which X.509 validation functionality is used.

Restrictions on Acceptable Purposes for Certificate Acquisition for FIA_XCU_EXT.2.1

The PP does not define SMIME functionality, therefore the ST shall not select this as a function for which X.509 validation functionality is used.

Functional Package for Transport Layer Security (TLS), Version 2.1

TLS Client Functionality Required in FCS_HTTPS_EXT.1

The ST author shall claim support for TLS as a client.

Functional Package for Secure Shell (SSH), Version 2.0

SSH Client Functionality Required in FIA_UAU.5

Selection of "public key-based authentication" via SSH requires the ST author to claim support for SSH and related functional package requirements.

SSH Functionality Required in FTP_ITC_EXT.1

The ST author shall claim support for SSH when the option to set up a trusted communication channel with SSH as the protocol is selected. The TOE may act as a client, server, or both as necessary and documented by the ST author.

1.6 Roles

For purposes of these requirements there are two entities that interact with a general-purpose computing platform:

1. Users (unprivileged users)
2. Administrators (privileged users)

Users are humans who interact with the platform through user interfaces. They usually have to authenticate themselves to tenant software (e.g. an operating system), but generally not to the platform itself. Throughout this document the term "user" refers generally to a person interacting with the platform.

Administrators are users who manage the platform through a management interface. The interface may be local or remote to the platform.

Administrators manage the physical platform, not the OS (OS Administrators would be classified as platform Users). Administrators must be authenticated to the platform before the platform can allow them to perform administrative functions. For an EUD, this could be accomplished through an interface implemented in firmware. For server-class hardware, the management interface could be implemented in a management controller that is part of the platform. Administrators are assumed to be acting in the best interests of the platform owner.

Tenant Software generally consists of an operating system, virtualization system, or application that uses platform resources to run workloads on behalf of Users. Tenant software generally has the privilege of the User or Administrator in whose context it runs.

2 Conformance Claims

Conformance Statement

An ST must claim exact conformance to this PP.

The evaluation methods used for evaluating the TOE are a combination of the workunits defined in [CEM] as well as the Evaluation Activities for ensuring that individual SFRs and SARs have a sufficient level of supporting evidence in the Security Target and guidance documentation and have been sufficiently tested by the laboratory as part of completing ATE_IND.1. Any functional packages this PP claims similarly contain their own Evaluation Activities that are used in this same manner.

CC Conformance Claims

This PP is conformant to Part 2 (extended) and Part 3 (extended) of Common Criteria CC:2022, Revision 1 as corrected and interpreted in [ERR], Version 1.1.

PP Claim

This PP does not claim conformance to any Protection Profile.

There are no PPs or PP-Modules that are allowed in a PP-Configuration with this PP.

Package Claim

- This PP is Functional Package for Secure Shell (SSH) Version 2.0 conformant.
- This PP is Functional Package for Transport Layer Security Version 2.1 conformant.
- This PP is Functional Package for X.509 Version 1.0 conformant.
- This PP does not conform to any assurance packages.

The functional packages to which the PP conforms may include SFRs that are not mandatory to claim for the sake of conformance. An ST that claims one or more of these functional packages may include any non-mandatory SFRs that are appropriate to claim based on the capabilities of the TSF and on any triggers for their inclusion based inherently on the SFR selections made.

3 Security Problem Definition

The security problem is described in terms of the threats that the GPCP is expected to address, assumptions about the operational environment, and any organizational security policies that the GPCP is expected to enforce.

The platform has three major security responsibilities:

- ensuring the integrity of its own firmware and hardware
- ensuring that it is resilient
- providing security services to tenant workloads

These responsibilities manifest as protecting:

- Platform firmware and hardware
- Platform firmware updates
- Tenant initialization (boot)

3.1 Threats

T.PHYSICAL
An attacker with physical access might be able to compromise TOE integrity, subvert TOE protections, or access tenant data through hardware attacks such as probing, physical manipulation, fault-injection, side-channel analysis, environmental stress, or activating disabled features or pre-delivery services.

T.SIDE_CHANNEL_LEAKAGE
An attacker running in a tenant context might be able to leverage physical effects caused by the operation of the TOE to derive sensitive information about other tenants or the TOE.

T.PERSISTENCE
An attacker might be able to establish a permanent presence on the TOE in firmware. This could result in permanent compromise of tenant information, as well as TOE updates. This threat does not encompass attacker presence in tenant software, as tenant software is not part of the TOE.

T.UPDATE_COMPROMISE
An attacker may attempt to provide a compromised update of TOE firmware. Such updates can undermine the security functionality of the device if they are unauthorized, unauthenticated, or are improperly validated using non-secure or weak cryptography.

T.SECURITY_FUNCTIONALITY_FAILURE
An attacker could leverage failed or compromised security functionality to access, change, or modify tenant data, TOE data, or other security functionality of the device.

T.TENANT_BASED_ATTACK
An attacker running software as a tenant can attempt to access or modify TOE firmware or functionality. Note that direct tenant attacks against other tenants are not encompassed by this threat as they are out of scope.

T.NETWORK_BASED_ATTACK
An attacker from off the TOE can attempt to compromise the TOE through a network interface connected to an active TOE component, such as a management subsystem.

T.UNAUTHORIZED_RECONFIGURATION
An attacker might be able to modify the configuration of the TOE and alter its functionality. This might include, activating dormant subsystems, disabling hardware assists, or altering boot-time behaviors.

T.UNAUTHORIZED_PLATFORM_ADMINISTRATOR
An attacker might be able to attain platform administrator status by defeating or bypassing authentication measures.

3.2 Assumptions

A.PHYSICAL_PROTECTION
The TOE is assumed to benefit from varying levels of protections depending on use case and operational environment.

A.ROT_INTEGRITY
The TOE includes one or more Roots of Trust composed of TOE firmware, hardware, and pre-installed credentials. Roots of Trust are assumed to be free of malicious capabilities as their integrity cannot be verified.

A.TRUSTED_ADMIN

TOE Security Administrator are assumed to be trusted and to act in the best interest of security for the organization. The TOE is not expected to be capable of defending against a malicious Administrator that actively works to bypass or compromise the security of the platform.

A.MFR_ROT

The root signing credential of the manufacturer is assumed to be secure and has not been compromised.

A.TRUSTED_DEVELOPMENT_AND_BUILD_PROCESSES

The TOE cannot protect itself during its own development and build processes. Therefore it is assumed that the developers and participants in the build process are not hostile.

A.SUPPLY_CHAIN_SECURITY

The hardware components that comprise the TOE are assumed to be non-hostile and not compromised at the time of TOE construction. Likewise, the TOE is assumed to retain its integrity throughout transportation until delivery to its operational site.

A.CORRECT_INITIAL_CONFIGURATION

It is assumed that the initial setup and configuration of the TOE at its operational site is correct and in accordance with organizational security policy and operational use case.

A.TRUSTED_USERS

Physically present non-administrative users of the TOE are assumed to be trusted as far as they are assumed to not be actively trying to subvert the system. (Not for all use cases).

A.REGULAR_UPDATES

It is assumed that the manufacturer provides updates to TOE firmware in a timely manner in response to known vulnerabilities, and that Administrators apply these updates when they are received.

3.3 Organizational Security Policies

This PP defines no Organizational Security Policies.

4 Security Objectives

4.1 Security Objectives for the Operational Environment

The following security objectives for the operational environment assist the GPCP in correctly providing its security functionality. These track with the assumptions about the environment.

OE.PHYSICAL_PROTECTION

A TOE may benefit from different levels of protection provided by its operational environment. Platforms that operate within data centers or in other access-controlled environments are expected to receive a considerable degree of protection from these environments. In addition to physical protection, these environments often provide malware-detection and behavior-monitoring services for networked computing assets. On the other hand, a TOE would receive very little protection from a tactical environment.

OE.SUPPLY_CHAIN

The manufacturer is expected to implement processes to ensure that TOE hardware and firmware is not compromised between time of TOE manufacture and delivery to its operational site.

OE.TRUSTED_ADMIN

The administrator of the GPCP is not careless, willfully negligent or hostile, and administers the platform within compliance of enterprise security policy.

4.2 Security Objectives Rationale

This section describes how the assumptions and organizational security policies map to operational environment security objectives.

Table 1: Security Objectives Rationale

Assumption or OSP	Security Objectives	Rationale
A.PHYSICAL_PROTECTION	OE.PHYSICAL_PROTECTION	The operational environment objective OE.PHYSICAL_PROTECTION is realized through A.PHYSICAL_PROTECTION.
A.ROT_INTEGRITY	OE.SUPPLY_CHAIN	The operational environment objective OE.SUPPLY_CHAIN is realized through A.ROT_INTEGRITY.
A.TRUSTED_ADMIN	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.
A.MFR_ROT	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.
A.TRUSTED_DEVELOPMENT_AND_BUILD_PROCESSES	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.
A.SUPPLY_CHAIN_SECURITY	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.
A.CORRECT_INITIAL_CONFIGURATION	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.
A.TRUSTED_USERS	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.
A.REGULAR_UPDATES	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.

5 Security Requirements

This chapter describes the security requirements which have to be fulfilled by the product under evaluation. Those requirements comprise functional components from Part 2 and assurance components from Part 3 of [CC]. The following conventions are used for the completion of operations:

- **Refinement** operation (denoted by **bold text** or ~~strikethrough text~~): Is used to add details to a requirement or to remove part of the requirement that is made irrelevant through the completion of another operation, and thus further restricts a requirement.
- **Selection** (denoted by *italicized text*): Is used to select one or more options provided by the [CC] in stating a requirement.
- **Assignment** operation (denoted by *italicized text*): Is used to assign a specific value to an unspecified parameter, such as the length of a password. Showing the value in square brackets indicates assignment.
- **Iteration** operation: Is indicated by appending the SFR name with a slash and unique identifier suggesting the purpose of the operation, e.g. "/EXAMPLE1."

5.1 Security Functional Requirements

5.1.1 Auditable Events for Mandatory SFRs

Table 2: Auditable Events for Mandatory Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FMT_CFG_EXT.1	No events specified	N/A
FMT_LIM.1	No events specified	N/A
FMT_LIM.2	No events specified	N/A
FMT_MOF.1	No events specified	N/A
FMT_SMF.1	No events specified	N/A
FMT_SMR.1	No events specified	N/A
FPT_PPF_EXT.1	No events specified	N/A
FPT_ROT_EXT.1	No events specified	N/A
FPT_ROT_EXT.2	**[selection:** *Failure of integrity verification, None*]	None.
FPT_STM.1	No events specified	N/A
FPT_TUD_EXT.1	No events specified	N/A

5.1.2 Class: Security Management (FMT)

FMT_CFG_EXT.1 Secure by Default Configuration

FMT_CFG_EXT.1.1

The TSF shall enforce that Administrator credentials be changed immediately after first use when configured with default Administrator credentials or with no Administrator credentials.

Application Note: Default credentials are credentials (e.g., passwords, keys) that are pre-installed (without user interaction) onto the platform, generally by the manufacturer, whether they are default values or randomly generated. This requirement applies only to credentials used by an Administrator for logging in to the TOE, and not to other platform credentials that might come pre-installed.

Evaluation Activities

FMT_CFG_EXT.1
TSS

The evaluator shall check the TSS to determine whether the platform comes pre-installed with default Administrator credentials, or does not require credentials for initial Administrator access.

Guidance
The evaluator shall examine the AGD to ensure that it describes the process for replacing or specifying Administrator credentials on first use.

Tests
If the platform uses default Administrator credentials or no Administrator credentials on first use the evaluator shall run the following tests:

- *Test FMT_CFG_EXT.1:1: The evaluator shall reset the platform to factory state and restart the platform to verify that only the functionality required to set new Administrator credentials is available immediately after Administrator login.*
- *Test FMT_CFG_EXT.1:2: The evaluator shall log in to the platform as Administrator using the default credentials, establish new credentials, and verify that the original default credentials no longer provide Administrative access to the platform.*

FMT_LIM.1 Limited Capabilities

FMT_LIM.1.1

The TSF shall limit its capabilities so that in conjunction with "Limited Availability (FMT_LIM.2)" the following policy is enforced: [Deploying test and debug features after TOE delivery does not allow user data of the TOE to be disclosed or manipulated, TSF data to be disclosed or manipulated, and firmware to be reconstructed such that information about construction of the TSF may enable other attacks.]

Application Note: The GPCP may implement test and debug features such as (but not limited to) JTAG, SWD, UART, and USB. The TOE shall prevent abuse of such functionality after the production test phase. The protection can be achieved by limiting the capability of the implemented features and/or limiting their availability.

Limited capability of test and debug features focuses on the suite of implemented functions that are usable. Functions equivalent to get, read, and query must not be able to reveal TOE user data, data from within the TSF, or firmware executable instructions. Functions equivalent to set, write, and allocate must not alter TOE user data, alter data within the TSF security boundary, or compromise TSF firmware integrity.

FMT_LIM.1 and FMT_LIM.2 have the same evaluation activities.

Evaluation Activities

FMT_LIM.1
TSS
The evaluator shall examine the TSS to determine that all accessible test and debug functions/ports are properly identified. Vendors must disclose test and debug functions/ports to evaluators, but are not required to make the same disclosures to customers, end users, or other parties. Proper identification may take the form of labels printed on the TOE, documentation accompanying the TOE, or information provided to evaluators from the TOE vendor. The evaluator shall check publicly available documents to identify potential mismatches between documented test and debug functions/ports, test and debug mechanisms implemented by component vendors, and test and debug mechanisms identified by reverse engineers. The TOE shall be configured to enforce TSF boundaries per vendor configuration instructions (this may require disabling test and debugging features in firmware configuration).

Guidance
TOEs that do not provide test and debug functionality/ports are compliant with this requirement by default. TOEs that limit the capabilities of test and debug functionality/ports to exclude getting and setting of arbitrary data are also compliant by default.

Tests
The evaluator shall attempt to access test and debug functions/ports. If the test and debug functions/ports allow for authorization credentials, the evaluator shall create an arbitrary authorization credential that is not provided by the vendor. The evaluator shall connect probes or adapters to ports, pads, or interfaces as necessary. Utilize package analyzer or other tools as appropriate.

The evaluator shall conduct the following tests:

- *Test FMT_LIM.1:1: Execute all documented get, read, and query commands. Examine the captured output. Output must not contain user data, TSF data, or firmware instructions. No output is also acceptable.*

- *Test FMT_LIM.1:2: The evaluator shall execute all documented set, write, and allocate functions with the intent to inject arbitrary examination data through the TSF boundary. Attempts to alter user data, TSF data, and TOE firmware must fail.*

FMT_LIM.2 Limited Availability

FMT_LIM.2.1

The TSF shall be designed in a manner that limits its availability so that in conjunction with "Limited capabilities (FMT_LIM.1)" the following policy is enforced: [Deploying test and debug features after TOE delivery does not allow user data of the TOE to be disclosed or manipulated, TSF data to be disclosed or manipulated, and firmware to be reconstructed such that information about construction of the TSF may enable other attacks.]

Application Note: The GPCP may implement test and debug features such as (but not limited to) JTAG, SWD, UART, and USB. The TOE shall prevent abuse of such functionality after the production test phase. The protection can be achieved by limiting the capability of the implemented features and/or limiting their availability.

Limited availability means that test and debug functions/ports may not be accessible. Test and debug functions/ports may be restricted by an authorization credential, physically disconnected, or otherwise rendered unavailable after TOE delivery. This requirement should be included in the ST for use cases that include the threat T.PHYSICAL.

FMT_LIM.1 and FMT_LIM.2 have the same evaluation activities.

Evaluation Activities

FMT_LIM.2

TSS

The evaluator shall examine the TSS to determine that all accessible test and debug functions/ports are properly identified. Vendors must disclose test and debug functions/ports to evaluators, but are not required to make the same disclosures to customers, end users, or other parties. Proper identification may take the form of labels printed on the TOE, documentation accompanying the TOE, or information provided to evaluators from the TOE vendor. The evaluator shall check publicly available documents to identify potential mismatches between documented test and debug functions/ports, test and debug mechanisms implemented by component vendors, and test and debug mechanisms identified by reverse engineers. The TOE shall be configured to enforce TSF boundaries per vendor configuration instructions (this may require disabling test and debugging features in firmware configuration).

Guidance

TOEs that do not provide test and debug functionality/ports are compliant with this requirement by default. TOEs that limit the capabilities of test and debug functionality/ports to exclude getting and setting of arbitrary data are also compliant by default.

Tests

The evaluator shall attempt to access test and debug functions/ports. If the test and debug functions/ports allow for authorization credentials, the evaluator shall create an arbitrary authorization credential that is not provided by the vendor. The evaluator shall connect probes or adapters to ports, pads, or interfaces as necessary. Utilize package analyzer or other tools as appropriate.

The evaluator shall conduct the following tests:

- *Test FMT_LIM.2:1: Execute all documented get, read, and query commands. Examine the captured output. Output must not contain user data, TSF data, or firmware instructions. No output is also acceptable.*
- *Test FMT_LIM.2:2: The evaluator shall execute all documented set, write, and allocate functions with the intent to inject arbitrary examination data through the TSF boundary. Attempts to alter user data, TSF data, and TOE firmware must fail.*

FMT_MOF.1 Management of Security Functions Behavior

FMT_MOF.1.1

The TSF shall restrict the ability to [*determine the behaviour of*] the functions [*listed in Table 3*] to [*the roles indicated in Table 3*].

Application Note: There are two roles defined in this PP: Administrator and User (see FIA_SMR.1). Administrators can perform most management functions on the platform, and only Administrators are

required to authenticate themselves to the platform.

Users have a limited ability to select responses to certain events as specified in the Management Functions table in FMT_SMF.1.

Evaluation Activities

FMT_MOF.1
TSS
The evaluator shall verify that the TSS describes those management functions that may be performed by the Administrator, and those that can be performed by ordinary users. The TSS also describes any functionality that is affected by administrator-configured policy and how. This activity will be performed in conjunction with FMT_SMF.1.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
Testing of this SFR is covered in the tests for FMT_SMF.1.

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1

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

Table 3: Management Functions

Status Markers:
M - Mandatory
O - Optional/Selectable/Conditional
X - Not permitted

#	Management Function	Admin	User	Application Notes
1	Ability to administer the platform [**selection**: *locally*, *remotely*].	O	X	Administration is considered "local" if the Administrator is physically present at the GPCP. Administration is considered "remote" if communications between the Administrator and GPCP is over a network. If "*locally*" is selected, then function 6 is mandatory. If "*remotely*" is selected, then FTP_TRP.1 must be claimed in the ST and functions 5, 6, and 13 are mandatory.
2	Ability to configure and manage the audit functionality and audit data.	O	X	Management of audit data includes the ability to delete it. This Function must be claimed if FAU_GEN.1 is claimed in the ST.
3	Ability to configure name/address of audit/logging server to which to send audit/logging records.	O	X	This function must be claimed if FAU_STG.1 is claimed in the ST.
4	Ability to review audit records.	O	X	This Function must be claimed if FAU_SAR.1 is claimed in the ST.

5	Ability to initiate a trusted channel or accept an incoming channel for remote administration.	O	X	This Function must be claimed if FTP_TRP.1 is claimed in the ST. This function is mandatory when function 1 remote administrator is selected.
6	Ability to manage authentication credentials for Administrators.	O	X	This Function must be claimed if FIA_UIA_EXT.1 is claimed in the ST. This function is mandatory when function 1 is selected (local or remote).
7	Ability to set parameters for allowable number of authentication failures.	O	X	This Function must be claimed if FIA_AFL_EXT.1 is claimed in the ST.
8	Ability to configure password length and complexity.	O	X	This Function must be claimed if FIA_PMG_EXT.1 is claimed in the ST. If password length and complexity are not configurable, then the Administrator Option should be denied.
9	Ability to configure authentication throttling policy.	O	X	This Function must be claimed if FIA_TRT_EXT.1 is claimed in the ST. If authentication throttling policy is not configurable, then the Administrator Option should be denied.
10	Ability to manage authentication methods and change default authorization factors.	O	X	This Function must be claimed if FIA_UAU.5 is claimed in the ST. If authentication methods are not configurable, then the Administrator Option should be denied.
11	Ability to configure certificate revocation checking methods.	O	X	This function must be claimed if FIA_X509_EXT.1 is claimed in the ST (i.e., the TOE claims conformance to Functional Package for X.509, version 1.0. If TOE does not support configuration of certificate revocation checking methods, then the Administrator option should be denied.
12	Ability to configure TSF behavior when certificate revocation status cannot be determined.	O	X	This function must be claimed if FIA_X509_EXT.2 is claimed in the ST (i.e., the TOE claims conformance to Functional Package for X.509, version 1.0 and the claims made in the SFR indicate that the administrator is allowed to configure how the TSF treats a certificate with undetermined revocation status.
13	Ability to manage the IPsec reference identifier.	O	X	This function must be claimed if FCS_IPSEC_EXT.1 is claimed in the ST. This function is mandatory when function 1 remote administrator is selected.
14	Ability to configure default action to take on boot integrity failure.	O	X	This Function must be claimed if "*in accordance with Administrator-configurable policy*" is selected in FPT_ROT_EXT.2.2 or FPT_ROT_EXT.3.2.
15	Ability to configure default action to take on	O	X	This Function must be claimed if FPT_TUD_EXT.2 or FPT_TUD_EXT.3 is claimed

	update failure.			in the ST and "*in accordance with Administrator-configurable policy*" is selected in FPT_TUD_EXT.2.5 or FPT_TUD_EXT.3.4.
16	Ability to initiate the update process.	O	O	This Function must be claimed if something other than "*no mechanism for platform firmware update*" is selected in FPT_TUD_EXT.1.1.
17	Ability to determine the action to take on update failure.	O	O	This Function must be claimed if FPT_TUD_EXT.2 or FPT_TUD_EXT.3 are claimed in the ST.
18	Ability to determine the action to take on integrity check failure.	O	O	This Function must be claimed if FPT_ROT_EXT.2 or FPT_ROT_EXT.3 is claimed in the ST. The Administrator Option must be selected if "*by express determination of an [Administrator]*" is selected in FPT_ROT_EXT.2.2 or FPT_ROT_EXT.3.2. The User Option must be selected if "*by express determination of an [User]*" is selected in FPT_ROT_EXT.2.2 or FPT_ROT_EXT.3.2.
19	Ability to manage import and export of keys/secrets to and from protected storage.	O	X	This Function must be claimed if FCS_STG_EXT.1 is claimed in the ST.

Application Note: These functions become Mandatory or Selectable as indicated in the Notes.

If function 1 is not selected, then no other administrative functions may be selected.

Evaluation Activities

FMT_SMF.1
TSS
The evaluator shall examine the TSS to ensure that it describes each management function and its associated actions.

Guidance
The evaluator shall examine the AGD to ensure that it describes how the Administrator performs each management function that the ST claims the TOE supports.

The evaluator shall verify for each claimed management function that the guidance is sufficiently detailed to allow the function to be performed.

Tests
The evaluator shall test each management function included in the ST to demonstrate that the function can be performed only by the roles indicated in Table 3 and the result of the function is demonstrated.

FMT_SMR.1 Security Roles

FMT_SMR.1.1

The TSF shall maintain the roles [*User* and [**selection**: *Administrator*, *no other roles*]].

FMT_SMR.1.2

The TSF shall be able to associate users with roles.

Application Note: If "*Administrator*" is selected, then the user authentication SFRs in FIA must be claimed.

A User is a human who interacts with the GPCP through a user interface. Users do not authenticate themselves to the GPCP, though they may be authenticated by tenant software. The User role is considered to exist even if no humans normally interact with a GPCP.

An Administrator is a privileged user that must be authenticated by the GPCP in order to administer the GPCP. This role is distinct from OS or VS administrators, who are may are authenticated to tenant software and are considered to be Users in the context of the GPCP.

Evaluation Activities

FMT_SMR.1
Documentation and testing for roles is covered in the Evaluation Activities for FMT_SMF.1

5.1.3 Class: Protection of the TSF (FPT)

FPT_PPF_EXT.1 Protection of Platform Firmware and Critical Data

FPT_PPF_EXT.1.1

The TSF shall allow modification of platform firmware and critical data only through the update mechanisms described in FPT_TUD_EXT.1.

Application Note: Platform firmware must be modifiable only through one of the secure update mechanisms specified in FPT_TUD_EXT.1. If the update mechanism itself is implemented in platform firmware, then naturally, it must itself also be modifiable only through the secure update mechanism. Configuration data used by platform firmware that is stored in nonvolatile memory is not included in these protections. Executable portions of the TSF and data critical for ensuring the integrity of the TSF are included in these protections. Specifically, this includes the key store and the signature verification algorithm used by the update mechanisms.

Evaluation Activities

FPT_PPF_EXT.1
TSS
The evaluator shall examine the TSS to ensure that it explains how the various areas of platform firmware and critical data are protected from modification outside of the platform firmware update mechanism described in FPT_TUD_EXT.1. If the TOE implements an authenticated update mechanism as specified in FPT_TUD_EXT.2, then the evaluator shall ensure that the TSS describes specifically how the signature verification code and key store is protected from update outside of the secure platform firmware update mechanism.

Guidance
The evaluator shall check the AGD to ensure that there are instructions for how to securely modify the platform firmware and critical data using a mechanism specified in FPT_TUD_EXT.1.

Tests
The evaluator shall perform the following test:

The evaluator shall attempt to overwrite or modify the platform firmware without invoking one of the update mechanisms specified in FPT_TUD_EXT.1. The test succeeds if the attempts to overwrite platform firmware fail. The evaluator shall attempt at least three such tests--one that attempts to overwrite the first platform firmware that executes after boot, one that targets the secure update mechanism (if implemented), and one that targets firmware that has been integrity-checked since the last boot.

FPT_ROT_EXT.1 Platform Integrity Root

FPT_ROT_EXT.1.1

The integrity of platform firmware shall be rooted in [**selection**:

- *code and/or data written to immutable memory or storage*
- *credentials held in immutable storage on-platform or protected storage off-platform*
- *a separate management controller that is itself rooted in a mechanism that meets this requirement*
- *integrity measurements held securely in an on-platform dedicated security component*
- *integrity measurements held securely by an off-platform entity*

].

Application Note: Roots of Trust are components that constitute a set of unconditionally trusted functions. The above are acceptable roots of trust for platform firmware integrity.

The ST Author must select the root of trust used to ensure the integrity of the first platform firmware that executes. The integrity of subsequently executed platform firmware must be traceable back to this root or to some other root as specified in FPT_ROT_EXT.2. This SFR should be iterated for additional TOE roots (for example, a management controller or firmware executed from an add-in card).

An "on-platform dedicated security component" could be, for example, a TPM or other secure element that provides security services to the platform such as measurement or secure storage.

Selection of "*a separate management controller...* " implies the existence of an Administrator role.

Evaluation Activities

FPT_ROT_EXT.1
TSS
The evaluator shall verify that the TSS describes the Root of Trust on which initial integrity of platform firmware is anchored, consistent with the selection above. The description shall include means by which the Root of Trust is protected from modification.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
There are no test activities for this component.

FPT_ROT_EXT.2 Platform Integrity Extension

FPT_ROT_EXT.2.1

The integrity of all mutable platform firmware outside of the platform integrity root specified in FPT_ROT_EXT.1 shall be assessed prior to [**selection**: *execution, use, or access to TSF data*] through [**selection**:

- *computation of a hash by [**selection**: trusted code, an on-platform Dedicated Security Component (DSC)] and verification by [**selection**: trusted code and optionally trusted data when appropriate, an on-platform Dedicated Security Component (DSC), an off-platform verifier]* .
- *verification of a digital signature by [**selection**: trusted code and optionally trusted data when appropriate, an on-platform Dedicated Security Component (DSC), an off-platform verifier]* .
- *[**assignment**: some other well-documented evaluation mechanism involving integrity evidence and a verifier entity (e.g. an A/B Test, Hardware Security Module (HSM)).]*

].

Application Note: This requirement specifies the means to extend the initial integrity of platform firmware established by FPT_ROT_EXT.1.1 to subsequently executed platform firmware and data located in mutable storage. (Integrity of code and data written to immutable storage is assured).

Integrity must be extended through cryptographic means: either through hashes or digital signatures computed and verified by firmware that is trusted because it has previously had its integrity verified or is itself a Root of Trust. Verification can be performed by TOE components such as management controllers or non-TOE trusted entities such as remote verifiers.

If "*computation and verification of a hash by trusted code*" is selected, then FCS_COP.1/Hash must be claimed.

If "*verification of a digital signature by trusted code and optionally trusted data when appropriate*" is selected, then FCS_COP.1/SigVer must be claimed.

FPT_ROT_EXT.2.2

The TOE shall take the following actions if an integrity check specified in FPT_ROT_EXT.2.1 fails: [**selection**:

- *Stop all execution, or*
- *Notify an [**selection**: Administrator, User] by [**selection**: generating an audit event, [**assignment**: other notification method(s)]], and [**selection**:*
  - *Stop all execution*
  - *Shut down, or*
  - *Initiate a recovery process as specified in FPT_RVR_EXT.1*
  - *Skip all instructions that failed the integrity check and continue execution*

  *][**selection**:*
  - *automatically*
  - *in accordance with Administrator-configurable policy*

- *by express determination of an [**selection**: Administrator, User]*

]

].

Application Note: Notification of an administrator can take many forms. For server-class platforms, such notification could take the form of administrator alerts or audit events. For platforms without management controllers, notification could be achieved, for example, by blinking lights, beep codes, screen indications, or local logging.

If "Administrator" is selected anywhere in FPT_ROT_EXT.2.2, or if "in accordance with Administrator-configurable policy" is selected, then all Administrator authentication requirements must be included in the ST (FIA_UIA_EXT.1, FIA_UAU.5, FIA_PMG_EXT.1, FIA_AFL_EXT.1, FIA_UAU.7).

If "*generating an audit event*" is selected, then FAU_GEN.1, FAU_SAR.1, FAU_STG.1, FAU_STG.2, and FAU_STG.5 must be claimed in the ST. This selection should be made only if the TOE is capable of generating an audit event, e.g. if the TOE is a server that includes a management controller.

If "*Initiate a recovery process as specified in FPT_RVR_EXT.1*" is selected, then FPT_RVR_EXT.1 must be included in the ST.

If "*in accordance with administrator-configurable policy*" is selected, then management function 14 must be claimed in FMT_SMF.1.

Evaluation Activities

FPT_ROT_EXT.2
TSS
The evaluator shall verify that the TSS describes the means by which initial integrity of platform firmware is extended to other platform components, and that the means are consistent with the selection(s) made in FPT_ROT_EXT.2. The TSS shall also describe how the TOE responds to failure of verification consistent with the selections in FPT_ROT_EXT.2.2.

Guidance
The evaluator shall examine the AGD to ensure that it describes the actions taken and notification methods used in case of failure to establish the integrity of the platform firmware root. If the actions are configurable, the AGD shall explain how they are configured.

Tests
The evaluator shall modify the platform firmware in a way that should cause a failure of the integrity check. The test passes if the mechanism specified in FPT_ROT_EXT.2.2 is triggered on the first subsequent boot of the platform.

Depending on the protections implemented, the evaluator may need a specially crafted update module from the vendor to perform this test. But note that this is not necessarily the same as a test of the update mechanism. The update mechanism can be tested either at boot time or at the time of the update. This verification check must be done during boot.

If modification of platform firmware in situ or using the update mechanism is deemed to be not feasible within the time and cost constraints of the evaluation, then the evaluator shall make such an argument in the AAR, and with concurrence of the CC scheme, this test can be replaced by evidence of vendor testing.

FPT_STM.1 Reliable Time Stamps

FPT_STM.1.1

The TSF shall be able to provide reliable time stamps.

Application Note: It is acceptable for the TSF to provide time stamp data either through an internal clock or a counter. It is also permissible for the TSF to obtain time data from a clock contained within the same physical enclosure as the TOE.

Devices may be configured with an internal clock value or counter value during the production phase. This SFR does not require the internal clock match a specific external clock such as a regional or world clock (e.g. GMT, UTC). This SFR also does not establish any requirements regarding clock drift because the ability to synchronize with an external clock authority cannot be guaranteed.

In the event of a disruption to reliable time (e.g. total loss of line and battery power, a service event) the internal clock or counter should continue incrementing once the disruption is resolved. If a default clock or counter value is set, then the device must continue to increment time predictably and may optionally generate an audit event indicating that a default has been set. The TOE must claim FAU_GEN.1 if audit events are generated due to changes to reliable time.

Evaluation Activities

FPT_STM.1
TSS
The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time. The TSS provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

Guidance
The evaluator shall examine the AGD to ensure it instructs the Administrator on any mechanisms for configuring the time source.

Tests
The evaluator shall perform the following tests:

If the TSF provides a mechanism to manually set the time, the evaluator shall use the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time is reported correctly and that the clock or time-related counter increments in an expected way (i.e. the clock or counter is not frozen to the value set during this test).

Otherwise, the evaluator shall use an available interface to observe that the clock or time-related counter is not null and is incrementing in an expected way (i.e. the clock or counter is not frozen to a constant/static value).

FPT_TUD_EXT.1 TOE Firmware Update

FPT_TUD_EXT.1.1

The TSF shall implement [**selection**:

- *an authenticated platform firmware update mechanism as described in FPT_TUD_EXT.2*
- *a delayed-authentication platform firmware update mechanism as described in FPT_TUD_EXT.3*
- *a secure local platform firmware update mechanism described in FPT_TUD_EXT.4*
- *no mechanism for platform firmware update*

].

Application Note: The purpose of the platform firmware update mechanism is to ensure the authenticity and integrity of platform firmware updates.

If platform firmware is immutable (not updateable by any non-destructive means), then the ST Author selects "*no mechanism for platform firmware update*."

If the platform implements an update mechanism that does not require physical presence at the platform, and that authenticates firmware updates prior to installing them, then the ST Author selects "*an authenticated platform firmware update mechanism...* " and includes FPT_TUD_EXT.2 and FCS_COP.1/SigVer in the ST.

If the platform implements an update mechanism that does not require physical presence at the platform, and that does not authenticate firmware updates prior to installing them, then the ST Author selects "*a delayed-authentication platform firmware update mechanism...* " and includes FPT_TUD_EXT.3 and FCS_COP.1/SigVer in the ST.

If platform firmware is modifiable only through a local update requiring physical presence at the platform, then the ST Author must select "*a secure local platform firmware update mechanism...* " and include FPT_TUD_EXT.4 in the ST.

Evaluation Activities

FPT_TUD_EXT.1
TSS
If the ST Author selects "no mechanism for platform firmware update," then the evaluator shall examine the TSS to ensure that it explains all ways of modifying platform firmware in the absence of any provided mechanism. For example, breaking open the case and prying a chip off the motherboard and then reprogramming the chip. The purpose of this activity is to ensure that the TOE does not implement a local update mechanism that does not meet the requirements of FPT_TUD_EXT.4.

This requirement is met if the platform implements no means for updating platform firmware and the TSS describes a method for updating or replacing platform firmware that involves destroying or damaging the TOE or some of its components.

If the ST Author selects "an authenticated platform firmware update mechanism... ," then this requirement is satisfied if FPT_TUD_EXT.2 is satisfied.

If the ST Author selects "a delayed-authentication platform firmware update mechanism... ," then this requirement is satisfied if FPT_TUD_EXT.3 is satisfied.

If the ST Author selects "a secure local platform firmware update mechanism... ," then this requirement is satisfied if FPT_TUD_EXT.4 is satisfied.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
There are no test activities for this component.

5.1.4 TOE Security Functional Requirements Rationale

The following rationale provides justification for each SFR for the TOE, showing that the SFRs are suitable to address the specified threats:

Table 4: SFR Rationale

Threat	Addressed by	Rationale
T.PHYSICAL	FPT_JTA_EXT.1	Mitigates threat by restricting access to debug ports to authorized Administrators or physical presence.
	FPT_ROT_EXT.3 (objective)	Mitigates threat by ensuring integrity of physical components and responding to integrity failures.
	FPT_JTA_EXT.2 (sel-based)	Mitigates threat by enforcing access control to debug ports.
	FPT_PHP.1 (sel-based/objective)	Mitigates threat by passively detecting physical tampering.
	FPT_PHP.2 (sel-based/objective)	Mitigates threat by providing methods to detect and report physical tampering.
	FPT_PHP.3 (sel-based)	Mitigates threat by resisting physical tampering.
T.SIDE_CHANNEL_LEAKAGE	FPT_TUD_EXT.1	Mitigates threat by providing a means to eliminate side-channel flaws through updates.
T.PERSISTENCE	FPT_ROT_EXT.1	Mitigates threat by providing platform integrity to prevent intrusion of a persistent presence on the platform.
	FPT_RVR_EXT.1 (sel-based)	Mitigates threat with firmware recovery mechanism in case of failure.
	FCS_STG_EXT.2 (sel-based)	Mitigates threat by enforcing access control on key data to prevent its unauthorized disclosure.
T.UPDATE_COMPROMISE	FPT_PPF_EXT.1	Mitigates threat by using the official update process to be the only method to modify platform firmware.
	FPT_ROT_EXT.2	Mitigates threat by providing a means to attest the validity of updates.
	FCS_COP.1/Hash (sel-based)	Mitigates threat by providing a means to validate the integrity of an update using a hash.
	FCS_COP.1/SigVer (sel-based)	Mitigates threat by providing a means to validate the integrity of an update using a hash.
	FPT_TUD_EXT.2 (sel-based)	Mitigates threat by using a digital signature mechanism to verify the integrity of updates and a rollback protection mechanism to prevent application of an unauthorized update.
	FPT_TUD_EXT.3 (sel-based)	Mitigates threat by using the TOE's root of trust to validate the authenticity and integrity of an update when it is applied.

	FPT_TUD_EXT.4 (sel-based)	Mitigates threat through an update mechanism that requires physical access to the TOE to use.
T.SECURITY_ FUNCTIONALITY_FAILURE	FCS_STG_EXT.1 (optional)	Mitigates threat by generating keys/secrets and storing them in a secure manner, as well as destroying them on request.
	FCS_CKM.6 (sel-based)	Mitigates threat by using appropriate key destruction methods to protect the confidentiality of credential data.
	FCS_COP.1/SigGen (sel-based)	Mitigates threat by generating digital signatures with strong encryption.
	FCS_COP.1/SKC (sel-based)	Mitigates threat by establishing strong symmetric-key cryptography.
	FPT_FLS.1 (sel-based)	Mitigates threat by ensuring a DRBG self-test failure causes the TOE to enter an error state where it cannot perform secure functions using that DRBG.
	FDP_ITC_EXT.1 (sel-based)	Mitigates threat by importing keys and credentials in a secure fashion.
	FCS_RBG.1 (sel-based)	Mitigates threat by performing random-bit generation with sufficient complexity.
	FCS_RBG.2 (sel-based)	Mitigates threat by using an external seed source to ensure sufficiently strong random-bit generation.
	FCS_RBG.3 (sel-based)	Mitigates threat by using an internal seed source to ensure sufficiently strong random-bit generation.
	FCS_RBG.4 (sel-based)	Mitigates threat by using multiple internal seed sources to ensure sufficiently strong random-bit generation.
	FCS_RBG.5 (sel-based)	Mitigates threat by ensuring that each noise source's random data is combined to ensure strong entropy when multiple sources are used.
	FPT_TST.1 (sel-based)	Mitigates threat by using self-tests to ensure correct operation of the DRBG.
T.TENANT_BASED_ATTACK	FPT_STM.1	Mitigates threat by ensuring that audit data indicating a potential attack is accurately timestamped.
	FCS_RBG.6 (optional)	Mitigates threat by providing a well-defined interface by which tenant software can access the TSF to obtain random data.
	FDP_TEE_EXT.1 (optional)	Mitigates threat by establishing a trusted execution environment for tenant software to use.
	FCS_CKM.1/AKG (sel-based)	Mitigates threat by generating strong cryptographic asymmetric keys to protect stored data.
	FCS_CKM.1/SKG (sel-based)	Mitigates threat by generating strong cryptographic symmetric keys to protect stored data.
	FCS_CKM.5 (sel-based)	Mitigates threat by utilizing strong algorithms to derive keys that protect stored data.
	FCS_CKM.6 (sel-based)	Mitigates threat by implementing key destruction to prevent the disclosure of keys used to protect stored data.
	FCS_COP.1/Hash (sel-based)	Mitigates threat by implementing hash functions used for trusted communications.
	FCS_COP.1/KeyedHash (sel-based)	Mitigates threat by implementing MAC functions used for trusted communications.
	FCS_COP.1/SigGen (sel-based)	Mitigates threat by implementing signature generation functions used for protected storage.

	FCS_COP.1/SigVer (sel-based)	Mitigates threat by implementing signature verification functions used for protected storage.
	FCS_COP.1/SKC (sel-based)	Mitigates threat by implementing symmetric encryption functions used for protected storage.
	FAU_GEN.1 (sel-based)	Mitigates threat by generating audit records that could provide evidence of attack or misuse.
	FAU_SAR.1 (sel-based)	Mitigates threat by recording audit data in a manner that could be interpreted to discover evidence of attack.
	FAU_STG.1 (sel-based)	Mitigates threat by using an external server to preserve audit data that may provide evidence of an attack.
	FAU_STG.2 (sel-based)	Mitigates threat by preventing audit records indicating a potential attack from being destroyed.
	FAU_STG.5 (sel-based)	Mitigates threat by ensuring that exhaustion of audit storage does not prevent audit data indicating a potential attack from being generated.
	FCS_STG_EXT.2 (sel-based)	Mitigates threat by using cryptography to protect the confidentiality of key data from outside access.
	FCS_STG_EXT.3 (sel-based)	Mitigates threat by using cryptography to protect the integrity of key data from outside modification.
	FPT_JTA_EXT.1	Mitigates threat by restricting access to debug ports to authorized Administrators or physical presence.
	FPT_JTA_EXT.2 (sel-based)	Mitigates threat by enforcing access control to debug ports.
T.NETWORK_BASED_ATTACK	FPT_STM.1	Mitigates threat by ensuring that audit data indicating a potential attack is accurately timestamped.
	FCS_CKM.1/AKG (sel-based)	Mitigates threat by generating strong cryptographic asymmetric keys to protect data in transit.
	FCS_CKM.1/SKG (sel-based)	Mitigates threat by generating strong cryptographic symmetric keys to protect data in transit.
	FCS_CKM.2 (sel-based)	Mitigates threat by implementing key establishment to negotiate trusted channels to protect data in transit.
	FCS_CKM.6 (sel-based)	Mitigates threat by implementing key destruction to prevent the compromise of trusted channels.
	FCS_CKM_EXT.7 (sel-based)	Mitigates threat by implementing MAC functions used for trusted communications.
	FCS_COP.1/Hash (sel-based)	Mitigates threat by implementing hash functions used for trusted communications.
	FCS_COP.1/KAT (sel-based)	Mitigates threat by implementing key agreement and transport functions used for trusted communications.
	FCS_COP.1/SigGen (sel-based)	Mitigates threat by implementing signature generation functions used for trusted communications.
	FCS_COP.1/SigVer (sel-based)	Mitigates threat by implementing signature verification functions used for trusted communications.
	FCS_COP.1/SKC (sel-based)	Mitigates threat by implementing symmetric encryption functions used for trusted communications.
	FAU_GEN.1 (sel-based)	Mitigates threat by generating audit records that could provide evidence of attack or misuse.

	FCS_HTTPS_EXT.1 (sel-based)	Mitigates threat by implementing HTTPS as a means to protect data in transit.
	FCS_IPSEC_EXT.1 (sel-based)	Mitigates threat by implementing IPsec as a means to protect data in transit.
	FTP_ITC_EXT.1 (sel-based)	Mitigates threat by ensuring that sensitive data in transit uses trusted protocols.
	FTP_ITE_EXT.1 (sel-based)	Mitigates threat by ensuring that sensitive data transmitted over untrusted channels is encrypted prior to transit.
	FTP_ITP_EXT.1 (sel-based)	Mitigates threat by using a physically protected channel to protect data in transit.
	FAU_SAR.1 (sel-based)	Mitigates threat by recording audit data in a manner that could be interpreted to discover evidence of attack.
	FAU_STG.1 (sel-based)	Mitigates threat by using an external server to preserve audit data that may provide evidence of an attack.
	FAU_STG.2 (sel-based)	Mitigates threat by preventing audit records indicating a potential attack from being destroyed.
	FAU_STG.5 (sel-based)	Mitigates threat by ensuring that exhaustion of audit storage does not prevent audit data indicating a potential attack from being generated.
	FTP_TRP.1 (sel-based)	Mitigates threat by ensuring that remote administration only uses trusted channels.
T.UNAUTHORIZED_ RECONFIGURATION	FMT_CFG_EXT.1	Mitigates threat by preventing knowledge of a default credential from being used to access the TSF without authorization.
	FMT_MOF.1	Mitigates threat by permitting management functions to be used only by authorized users.
	FMT_SMF.1	Mitigates threat by specifying the management functions implemented by the TSF.
	FMT_SMR.1	Mitigates threat by defining the management roles which can be used to grant access to management functions.
	FIA_UIA_EXT.1 (sel-based)	Mitigates threat by preventing the TSF from being modified by an unauthenticated subject.
T.UNAUTHORIZED_ PLATFORM_ADMINISTRATOR	FPT_STM.1	Mitigates threat by ensuring that time-based authentication throttling or lockout is accurately enforced.
	FIA_TRT_EXT.1 (optional)	Mitigates threat by throttling authentication to prevent access via brute force.
	FIA_AFL_EXT.1 (sel-based)	Mitigates threat by limiting further authentication attempts once a failure threshold of a critical authentication mechanism has been reached.
	FIA_PMG_EXT.1 (sel-based)	Mitigates threat by enforcing password complexity requirements to prevent credentials from being easily guessed.
	FIA_UAU.5 (sel-based)	Mitigates threat by implementing multiple authentication mechanisms for accessing the TSF.
	FIA_UAU.7 (sel-based)	Mitigates threat by preventing disclosure of authentication data during authentication attempts.

5.2 Security Assurance Requirements

The Security Objectives in were constructed to address threats identified in Section 3.1 Threats. The Security Functional Requirements (SFRs) in Section 5.1 Security Functional Requirements are a formal instantiation of the Security Objectives. The PP identifies the Security

Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of SARs from CC part 3 that are required in evaluations against this PP. Individual Evaluation Activities to be performed are specified both in Section 5.1 Security Functional Requirements as well as in this section.

The general model for evaluation of GPCPs against STs written to conform to this PP is as follows:

After the ST has been approved for evaluation, the ITSEF will obtain the TOE, supporting environmental IT, and the administrative/user guides for the TOE. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The ITSEF also performs the Evaluation Activities contained within Section 5.1 Security Functional Requirements, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The Evaluation Activities that are captured in Section 5.1 Security Functional Requirements also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the PP.

The TOE Security Assurance Requirements are identified in Table 5.

Table 5: Security Assurance Requirements

Assurance Class	Assurance Components
Security Target (ASE)	Conformance Claims (ASE_CCL.1)
	Extended Components Definition (ASE_ECD.1)
	ST Introduction (ASE_INT.1)
	Security Objectives for the Operational Environment (ASE_OBJ.1)
	Direct Rationale Security Requirements (ASE_REQ.1)
	Security Problem Definition (ASE_SPD.1)
	TOE Summary Specification (ASE_TSS.1)
Development (ADV)	Basic Functional Specification (ADV_FSP.1)
Guidance Documents (AGD)	Operational User Guidance (AGD_OPE.1)
	Preparative Procedures (AGD_PRE.1)
Life Cycle Support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM Coverage (ALC_CMS.1)
	Timely Security Updates (ALC_TSU_EXT.1)
Tests (ATE)	Independent Testing – Conformance (ATE_IND.1)
Vulnerability Assessment (AVA)	Vulnerability Survey (AVA_VAN.1)

5.2.1 Class ASE: Security Target

As per ASE activities defined in [CEM].

5.2.2 Class ADV: Development

The information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST. The TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The Evaluation Activities contained in Section 5.1 Security Functional Requirements should provide the ST Authors with sufficient information to determine the appropriate content for the TSS section.

ADV_FSP.1 Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TSFIs. It is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this PP will not necessarily have interfaces to the Operational Environment that are not directly invokable by TOE users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this PP, the activities for this family should focus on understanding the interfaces presented in the TSS, KMD, and any other supplemental evidence that may be required to satisfy the TSS Evaluation Activities, such as a non-public interface specification, in response to the functional requirements and the interfaces presented in the AGD documentation. No additional "functional specification" documentation is necessary to satisfy the Evaluation Activities specified. The interfaces that need to be evaluated are characterized through the information needed to perform the Evaluation Activities listed, rather than as an independent, abstract list.

Developer action elements:

ADV_FSP.1.1D

The developer shall provide a functional specification.

Content and presentation elements:

ADV_FSP.1.1C

The developer shall provide a tracing from the functional specification to the SFRs.

Application Note: As indicated in the introduction to this section, the functional specification comprises the information contained in the TSS, KMD, and any additional supplemental documentation. The developer may reference a website accessible to application developers and the evaluator. The Evaluation Activities in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

ADV_FSP.1.2C

The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.3C

The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.4C

The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.5C

The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Evaluator action elements:

ADV_FSP.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2E

The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

Evaluation Activities

ADV_FSP.1
There are no specific Evaluation Activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1 Security Functional Requirements, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other Evaluation Activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

5.2.3 Class AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel. Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes instructions to successfully install the TSF in that environment; and Instructions to manage the security of the TSF as a product and as a component of the larger operational environment. Guidance pertaining to particular security functionality is also provided; requirements on such guidance are contained in the Evaluation Activities specified with each requirement.

AGD_OPE.1 Operational User Guidance (AGD_OPE.1)

Developer action elements:

AGD_OPE.1.1D

The developer shall provide operational user guidance.

Application Note: The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages. Rather than repeat information here, the developer should review the Evaluation Activities for this component to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.1C

The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

Application Note: User and administrator are to be considered in the definition of user role.

AGD_OPE.1.2C

The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3C

The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

Application Note: This portion of the operational user guidance should be presented in the form of a checklist that can be quickly executed by IT personnel (or end-users, when necessary) and suitable for use in compliance activities. When possible, this guidance is to be expressed in the eXtensible Configuration Checklist Description Format (XCCDF) to support security automation. Minimally, it should be presented in a structured format which includes a title for each configuration item, instructions for achieving the secure configuration, and any relevant rationale.

AGD_OPE.1.4C

The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5C

The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AGD_OPE.1.6C

The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7C

The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

AGD_OPE.1
Some of the contents of the operational guidance are verified by the Evaluation Activities in Section 5.1 Security Functional Requirements and evaluation of the TOE according to the [CEM]. The following additional information is also required:

- *If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.*
- *If the TOE supports firmware updates, the documentation must describe the process for verifying updates to the TOE by verifying a digital signature – this may be done by the TOE or the underlying platform. The evaluator will verify that this process includes the following steps: Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory). Instructions for*

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

The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

AGD_PRE.1 Preparative Procedures (AGD_PRE.1)

Developer action elements:

AGD_PRE.1.1D

The developer shall provide the TOE, including its preparative procedures.

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

Content and presentation elements:

AGD_PRE.1.1C

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2C

The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_PRE.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2E

The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

Evaluation Activities

AGD_PRE.1
As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements.

5.2.4 Class ALC: Life-cycle Support

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

ALC_CMC.1 Labeling of the TOE (ALC_CMC.1)

This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user.

Developer action elements:

ALC_CMC.1.1D

The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.1.1C

The TOE shall be labeled with a unique reference.

Application Note: Unique reference information includes:

- TOE Model Name

- TOE Version
- TOE Description
- Software Identification (SWID) tags, if available
- Bill of Materials (BOM), manifests, and hashes, if available

Evaluator action elements:

ALC_CMC.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_CMC.1
The evaluator will check the ST to ensure that it contains sufficient information to specifically identify the TOE and the version that meets the requirements of the ST. Further, the evaluator will check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator will examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

ALC_CMS.1 TOE CM Coverage (ALC_CMS.1)

Given the scope of the TOE and its associated evaluation evidence requirements, this component's Evaluation Activities are covered by the Evaluation Activities listed for ALC_CMC.1.

Developer action elements:

ALC_CMS.1.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.1.1C

The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.2C

The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_CMS.1
The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the OS is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the Evaluation Activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

The evaluator will ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler and linker flags). The evaluator will ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator will ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

ALC_TSU_EXT.1 Timely Security Updates

This component requires the TOE developer, in conjunction with any other necessary parties, to provide information as to how the TOE is updated to address security issues in a timely manner. The documentation describes the process of providing updates to the public from the time a security flaw is reported/discovered, to the time an update is released. This description includes the parties involved (e.g., developer/OEM, component manufacturers) and the steps that are performed (e.g., developer testing), including worst case time periods, before an update is made available to the public.

For TOE implementations with immutable firmware, update might not be possible other than through replacement of the entire device. In this case, delivery of a new device with the necessary security fixes would constitute deployment of the security update.

Developer action elements:

ALC_TSU_EXT.1.1D

The developer shall provide a description in the TSS of how timely security updates are made to the TOE.

ALC_TSU_EXT.1.2D

The developer shall provide a description in the TSS of how users are notified when updates change security properties or the configuration of the product.

Content and presentation elements:

ALC_TSU_EXT.1.1C

The description shall include the process for creating and deploying security updates for TOE firmware.

ALC_TSU_EXT.1.2C

The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

Note: The reporting mechanism could include web sites, email addresses, as well as a means to protect the sensitive nature of the report (e.g., public keys that could be used to encrypt the details of a proof-of-concept exploit).

Evaluator action elements:

ALC_TSU_EXT.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_TSU_EXT.1
The evaluator will verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates for TOE firmware. The evaluator will also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator will also verify that each mechanism for deployment of security updates is described.

The evaluator will verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability. The evaluator will verify that this time is expressed in a number or range of days.

The evaluator will verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

5.2.5 Class ATE: Tests

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

ATE_IND.1 Independent Testing – Conformance (ATE_IND.1)

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operational) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 5.1

Security Functional Requirements being met, although some additional testing is specified for SARs in Section 5.2 Security Assurance Requirements. The Evaluation Activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the hardware configurations that are claiming conformance to this PP. Given the scope of the TOE and its associated evaluation evidence requirements, this component's Evaluation Activities are covered by the Evaluation Activities listed for ALC_CMC.1.

Developer action elements:

ATE_IND.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements:

ATE_IND.1.1C

The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.2E

The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

Evaluation Activities

ATE_IND.1
The evaluator will prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's Assurance Activities.

While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the OS and its platform.

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

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

5.2.6 Class AVA: Vulnerability Assessment

For the first generation of this protection profile, the evaluation lab is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, the evaluator will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

AVA_VAN.1 Vulnerability Survey (AVA_VAN.1)

Developer action elements:

AVA_VAN.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN.1.1C

The TOE shall be suitable for testing.

Evaluator action elements:

AVA_VAN.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.2E

The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

Application Note: Public domain sources include the Common Vulnerabilities and Exposures (CVE) dictionary for publicly known vulnerabilities.

AVA_VAN.1.3E

The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

Evaluation Activities

AVA_VAN.1
The evaluator will generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Appendix A - Optional Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this PP. This appendix contains three other types of optional requirements:

The first type, defined in Appendix A.1 Strictly Optional Requirements, are strictly optional requirements. If the TOE meets any of these requirements the vendor is encouraged to claim the associated SFRs in the ST, but doing so is not required in order to conform to this PP.

The second type, defined in Appendix A.2 Objective Requirements, are objective requirements. These describe security functionality that is not yet widely available in commercial technology. Objective requirements are not currently mandated by this PP, but will be mandated in the future. Adoption by vendors is encouraged, but claiming these SFRs is not required in order to conform to this PP.

The third type, defined in Appendix A.3 Implementation-dependent Requirements, are Implementation-dependent requirements. If the TOE implements the product features associated with the listed SFRs, either the SFRs must be claimed or the product features must be disabled in the evaluated configuration.

A.1 Strictly Optional Requirements

A.1.1 Auditable Events for Strictly Optional Requirements

Table 6: Auditable Events for Strictly Optional Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_CKM.5	No events specified	N/A
FCS_CKM_EXT.8	No events specified	N/A
FCS_STG_EXT.1	No events specified	N/A
FDP_TEE_EXT.1	No events specified	N/A
FIA_TRT_EXT.1	Authentication throttling triggered.	None.

A.1.2 Class ALC: Life-cycle Support

ALC_FLR.1 Basic Flaw Remediation (ALC_FLR.1)

This SAR is optional and may be claimed at the ST-Author's discretion.

Developer action elements:

ALC_FLR.1.1D

The developer shall document and provide flaw remediation procedures addressed to TOE developers.

Content and presentation elements:

ALC_FLR.1.1C

The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.1.2C

The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.1.3C

The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.1.4C

The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

Evaluator action elements:

ALC_FLR.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_FLR.1
The evaluator shall inspect the TSS and verify it identifies how to access the flaw remediation procedures.

ALC_FLR.2 Flaw Reporting Procedures (ALC_FLR.2)

This SAR is optional and may be claimed at the ST-Author's discretion.

Developer action elements:

ALC_FLR.2.1D

The developer shall document and provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.2.2D

The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.

ALC_FLR.2.3D

The developer shall provide flaw remediation guidance addressed to TOE users.

Content and presentation elements:

ALC_FLR.2.1C

The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.2.2C

The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.2.3C

The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.2.4C

The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.2.5C

The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.2.6C

The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users.

ALC_FLR.2.7C

The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.2.8C

The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

Evaluator action elements:

ALC_FLR.2.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_FLR.2
The evaluator shall inspect the TSS and verify it identifies how to access the flaw remediation procedures.
The evaluator shall inspect the guidance document and verify it describes how to access the flaw remediation guidance.

ALC_FLR.3 Systematic Flaw Remediation (ALC_FLR.3)

This SAR is optional and may be claimed at the ST-Author's discretion.

Developer action elements:

ALC_FLR.3.1D
The developer shall document and provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.3.2D
The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.

ALC_FLR.3.3D
The developer shall provide flaw remediation guidance addressed to TOE users.

Content and presentation elements:

ALC_FLR.3.1C
The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.3.2C
The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.3.3C
The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.3.4C
The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.3.5C
The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.3.6C
The flaw remediation procedures shall include a procedure requiring timely response and the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw.

ALC_FLR.3.7C
The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users.

ALC_FLR.3.8C
The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.3.9C
The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

ALC_FLR.3.10C
The flaw remediation guidance shall describe a means by which TOE users may register with the developer, to be eligible to receive security flaw reports and corrections.

ALC_FLR.3.11C
The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE.

Evaluator action elements:

ALC_FLR.3.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_FLR.3
The evaluator shall inspect the TSS and verify it identifies how to access the flaw remediation procedures.
The evaluator shall inspect the guidance document and verify it describes how to access the flaw remediation guidance.

A.1.3 Class: Cryptographic Support (FCS)

FCS_CKM.5 Cryptographic Key Derivation

FCS_CKM.5.1

The TSF shall derive cryptographic keys [**selection**: *Key type*] from [**selection**: *Input parameters*] in accordance with a specified key derivation algorithm [**selection**: *Key derivation algorithm*] and specified cryptographic key sizes [**selection**: *Key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 7 provides the allowable choices for completion of the selection operations of FCS_CKM.5.

Table 7: Allowable choices for FCS_CKM.5

Key type	Input parameters	Key derivation algorithm	Key sizes	List of standards
KDF-CTR	[**selection**: *Direct Generation from a Random Bit Generator as specified in FCS_RBG.1, Concatenated keys*]	KPF2 - KDF in Counter Mode using [**selection**: *AES-256-CMAC, HMAC-SHA-384, HMAC-SHA-512*] as the PRF	[**selection**: *256, 384, 512*] bits	[**selection**: *ISO/IEC 11770-6:2016 (Subclause 7.3.2) [KPF2], NIST SP 800-108 Revision 1 Update 1 (Section 4.1) [KDF in Counter Mode]*]
KDF-FB	[**selection**: *Direct Generation from a Random Bit Generator as specified in FCS_RBG.1, Concatenated keys*]	KPF3 - KDF in Feedback Mode using [**selection**: *AES-256-CMAC, HMAC-SHA-384, HMAC-SHA-512*] as the PRF	[**selection**: *256, 384, 512*] bits	[**selection**: *ISO/IEC 11770-6:2016 (Subclause 7.3.3) [KPF3], NIST SP 800-108 Revision 1 Update 1 (Section 4.2) [KDF in Feedback Mode]*]
KDF-DPI	[**selection**: *Direct Generation from a Random Bit Generator as specified in FCS_RBG.1, Concatenated keys*]	KDF in Double Pipeline Iteration Mode using [**selection**: *AES-256-CMAC, HMAC-SHA-384, HMAC-SHA-512*] as the PRF	[**selection**: *256, 384, 512*]bits	[**selection**: *ISO/IEC 11770-6:2016 (Subclause 7.3.4) [KPF4], NIST SP 800-108 Revision 1 Update 1 (Section 4.3) [KDF in Double-Pipeline Iteration Mode]*]
KDF-XOR	More than one intermediary key	exclusive OR (XOR)	[**selection**: *256, 384, 512*] bits	N/A
KDF-ENC	Two keys	Encrypting using an algorithm specified in [**selection**: *FCS_COP.1/SKC, FCS_COP.1/AEAD*] with a 256-bit key.	[**selection**: *256, 384, 512*] bits	N/A

KDF-HASH	Shared secret	Hash function [**selection**: *SHA-384, SHA-512*]	[**selection**: *256, 384, 512*] bits	NIST SP 800-56C Revision 2 (Section 4.1, Option 1) [One-Step Key Derivation]
KDF-MAC-1S	Shared secret, salt, IV, output length, fixed information	Keyed hash [**selection**: *HMAC-SHA-384, HMAC-SHA-512*]	[**selection**: *256, 384, 512*] bits	NIST SP 800-56C Revision 2 (Section 4.1, Options 2, 3) [One-Step Key Derivation]
KDF-MAC-2S	Shared secret, salt, IV, output length, fixed information, and [**selection**: *auxiliary shared secret, no other parameters*]	MAC Step [**selection**: *HMAC-SHA-384, HMAC-SHA-512*] as randomness extraction and; KDF Step [**selection**: *KDF-CTR, KDF-FB, KDF-DPI*].	[**selection**: *256, 384, 512*] bits	NIST SP 800-56C Revision 2 (Section 5) [Two-Step Key Derivation]

Application Note: If KDF-CTR, KDF-FB, or KDF-DPI is claimed, then either FCS_COP.1/CMAC or FCS_COP.1/KeyedHash must also be claimed, depending on the selection made for PRF.

If KDF-ENC is claimed, then either FCS_COP.1/SKC or FCS_COP.1/AEAD must be claimed, depending on the encryption algorithm claimed.

If KDF-Hash is claimed, then FCS_COP.1/Hash must also be claimed.

If KDF-MAC-1S is claimed, then FCS_COP.1/KeyedHash must also be claimed.

If KDF-MAC-2S is claimed, then both FCS_COP.1/KeyedHash and FCS_COP.1/CMAC must also be claimed.

In KDF-MAC-2S, CMAC has been removed as a selection for the MAC step because it requires selection of 128 bits for the output key size, which is not supported in CNSA. If HMAC is selected in the MAC step, then the same HMAC is used as the KDF.

The security strengths of the Pseudo-Random functions for the key derivation methods must be sufficient for the security strength of the keys derived through those methods. Since CNSA permits keys no smaller than 256 bits, no 128- or 192-bit PRFs are permitted.

Evaluation Activities

FCS_CKM.5

TSS

The evaluator shall verify that the TSS describes and documents:

- *that the security strengths of the Pseudo-Random functions for the key derivation methods are sufficient for the security strength of the keys derived through those methods.*
- *that the security strengths of the input parameters are sufficient for the security strength of the keys derived through these methods.*
- *that, if concatenated keys or intermediary keys are input parameters, the TSS describes the sources of the keys, and the order in which they are concatenated, along with any other values that are concatenated with them. This may occur in instances when input keying material for the KDF comes from two independent sources, for example, a client and a server.*
- *that, if KDF-XOR is selected the TSS describes this method as there is no standard that specifies how to derive a key from two keys using only XOR.*
- *that, if KDF-ENC is selected, the TSS documents the encryption algorithm used from FCS_COP.1/SKC or FCS_COP.1/AEAD with 256-bit keys, and describes which of the inputs is the plaintext and which is the key. There are no standards that specify how to derive a key from two keys using encryption (KDF-ENC).*
- *that, for KDF-MAC-1S, KDF-MAC-2S, and KDF-KMAC, the TSS documents that for each invocation of a KDF that reuses the same input shared secret or key, each invocation must use a distinct context string, IV, or salt. The TSS must also describe the composition and sizes of these input parameters. The evaluator must ensure that the context string, IV, and salt are generated in conformance with the relevant standards.*
- *that, if the TOE uses the derived key in a key chain/hierarchy, that TSS describes how the key is used as part of the key chain/hierarchy.*

Guidance

The evaluator shall verify that the Guidance instructs the administrator how to set any configurable parameters, such as context strings, salts, and IVs.

The evaluator shall verify that the Guidance instructs the administrator how to configure the TOE to choose specific PRFs, modes, and parameters.

Tests
The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

KDF in Counter Mode

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-CTR	[**selection:** Direct Generation from a Random Bit Generator as specified in FCS_RBG.1, Concatenated keys]	KPF2 - KDF in Counter Mode using [**selection:** AES-256-CMAC HMAC-SHA-384, HMAC-SHA-512] as the PRF	[**selection:** 256, 384, 512] bits	[**selection:** ISO/IEC 11770-6:2016 (Subclause 7.3.2) [KPF2], NIST SP 800-108 Revision 1 Update 1 (Section 4.1) [KDF in Counter Mode]]

To test the TOE's ability to derive cryptographic keys using KDF in Counter Mode/KDF2, the evaluator shall perform the Counter KDF Algorithm Functional Test using the following input parameters:

- *Pseudo Random Function (PRF) [AES-256-CMAC, HMAC-SHA-384, HMAC-SHA-512]*
- *Derived key length [256, 384, 512] bits*
- *Location of the counter [after fixed data, before fixed data, middle fixed data]*
- *Counter length [8, 16, 24, 32] bits*

Counter KDF Algorithm Functional Test
For each supported combination of the above input parameters the evaluator shall require the implementation under test to derive two keys using random data. The evaluator shall compare the resulting keys with keys generated using a known-good implementation using the same input parameters.

KDF in Feedback Mode

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-FB	[**selection:** Direct Generation from a Random Bit Generator as specified in FCS_RBG.1, Concatenated keys]	KPF3 - KDF in Feedback Mode using [**selection:** AES-256-CMAC, HMAC-SHA-384, HMAC-SHA-512] as the PRF	[**selection:** 256, 384, 512] bits	[**selection:** ISO/IEC 11770-6:2016 (Subclause 7.3.3) [KPF3], NIST SP 800-108 Revision 1 Update 1 (Section 4.2) [KDF in Feedback Mode]]

To test the TOE's ability to derive cryptographic keys using KDF in Feedback Mode/KDF3, the evaluator shall perform the Feedback KDF Algorithm Functional Test using the following input parameters:

- *Pseudo Random Function (PRF) [AES-256-CMAC, HMAC-SHA-384, HMAC-SHA-512]*
- *Derived key length [256, 384, 512] bits*
- *Location of the counter [none, after fixed data, before fixed data, before iterator]*
- *Counter length [0, 8, 16, 24, 32] bits*

Feedback KDF Algorithm Functional Test
For each supported combination of the above input parameters the evaluator shall require the implementation under test to derive two keys using random data. The evaluator shall compare the resulting keys with keys generated using a known-good implementation using the same input parameters.

KDF in Double-Pipeline Iteration Mode

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-DPI	[**selection:** Direct Generation from a Random Bit Generator as specified	KPF4 - KDF in Double-Pipeline Iteration Mode using [**selection:** AES-256-CMAC, HMAC-SHA-384,	[**selection:** 256, 384, 512] bits	[**selection:** ISO/IEC 11770-6:2016 (Subclause 7.3.4) [KPF4], NIST SP 800-108 Revision 1 Update 1 (Section

	in FCS_RBG.1, Concatenated keys]	HMAC-SHA-512] as the PRF		4.3) [KDF in Double-Pipeline Iteration Mode]]

To test the TOE's ability to derive cryptographic keys using KDF in Double Pipeline Iteration Mode/KDF4, the evaluator shall perform the Double Pipeline Iteration KDF Algorithm Functional Test using the following input parameters:

- *Pseudo Random Function (PRF) [AES-256-CMAC, HMAC-SHA-384, HMAC-SHA-512]*
- *Derived key length [256, 384, 512] bits*
- *Location of the counter [none, after fixed data, before fixed data, before iterator]*
- *Counter length [0, 8, 16, 24, 32] bits*

Double Pipeline Iteration KDF Algorithm Functional Test
For each supported combination of the above input parameters the evaluator shall require the implementation under test to derive two keys using random data. The evaluator shall compare the resulting keys with keys generated using a known-good implementation using the same input parameters.

KDF XORing Keys

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-XOR	More than one intermediary keys	exclusive OR (XOR)	[**selection:** 256, 384, 512] bits	N/A

There are no tests for this key derivation method.

KDF by Encrypting Keys

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-ENC	Two keys	Encrypting using an algorithm specified in [**selection:** FCS_COP.1/SKC, FCS_COP.1/AEAD] with a 256-bit key	[**selection:** 256, 384, 512] bits	N/A

Specific testing for this key derivation method is covered by testing for the supported symmetric encryption algorithms in FCS_COP.1/SKC or FCS_COP.1/AEAD.

KDF by Hashing a Shared Secret

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-HASH	Shared secret	Hash function [**selection:** SHA-384, SHA-512]	[**selection:** 256, 384, 512] bits	NIST SP 800-56C Revision 2 (Section 4.1, Option 1) [One-Step Key Derivation]

To test the TOE's ability to derive cryptographic keys by hashing a shared secret (a.k.a. One-Step HASH-based Key Derivation), the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Auxiliary Function [SHA-384, SHA-512]*
- *Derived key length [256, 384, 512] bits*

Algorithm Functional Test
For each supported fixed information pattern and combination of the above input parameters the evaluator shall require the implementation under test to derive 15 keys using random data for a shared secret that is the same size as the derived key. The evaluator shall compare the resulting keys with keys derived using a known-good implementation using the same fixed information patterns and input parameters.

One-Step MAC-based KDF

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards

KDF-MAC-1S	Shared secret, salt, output length, fixed information	Keyed Hash function [**selection:** HMAC-SHA-384, HMAC-SHA-512]	[**selection:** 256, 384, 512] bits	NIST SP 800-56C Revision 2 (Section 4.1, Options 2, 3) [One-Step Key Derivation]

To test the TOE's ability to derive cryptographic keys using One-Step MAC-based Key Derivation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Auxiliary Function [HMAC-SHA-384, HMAC-SHA-512]*
- *Salt [0s, random]*
- *Derived key length [256, 384, 512] bits*
- *Fixed information pattern*

Algorithm Functional Test

For each supported fixed information pattern and combination of the above input parameters the evaluator shall require the implementation under test to derive 15 keys using random data for a shared secret. The evaluator shall compare the resulting keys with keys derived using a known-good implementation using the same fixed information patterns and input parameters.

Two-Step MAC-based KDF

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-MAC-2S	Shared secret, salt, IV, output length, fixed information, and [**selection:** auxiliary shared secret, no other parameters]	MAC Step [**selection:** HMAC-SHA-384, HMAC-SHA-512] as randomness extraction and; KDF Step [**selection:** KDF-CTR, KDF-FB, KDF-DPI]	[**selection:** 256, 384, 512] bits	NIST SP 800-56C Revision 2 (Section 5) [Two-Step Key Derivation]

To test the TOE's ability to derive cryptographic keys using Two-Step MAC-based Key Derivation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *MAC mode [HMAC-SHA-384, HMAC-SHA-512]*
- *KDF Mode [Counter, feedback, Double Pipeline Iteration]*
- *Salt [0s, random]*
- *Length of shared secret [224-65535]*
- *Length of Auxiliary Shared Secret [0, 112-65535]*
- *Derived key length [256, 384, 512] bits*
- *Fixed information pattern*
- *Counter location [none, before fixed data, after fixed data, before iterator]*
- *Counter length [0, 8, 16, 24, 32]*

Algorithm Functional Test

The evaluator shall define a test group for each supported combination of KDF mode, MAC mode, fixed information pattern, derived key length, counter location, counter length, salt method, and five random pairs of shared secrets & auxiliary secrets (if supported) such that collectively the minimum length, maximum length and three random lengths of each are included in each test group. For each test group, the evaluator shall require the implementation under test to derive 25 keys using random data for a shared secret, either a random salt or a salt of all 0s, and, if supported, an auxiliary shared secret consisting of random data. The evaluator shall compare the resulting keys with keys derived using a known-good implementation using the same input parameters.

FCS_CKM_EXT.8 Password-Based Key Derivation

FCS_CKM_EXT.8.1

The TSF shall perform password-based key derivation functions in accordance with a specified cryptographic algorithm [HMAC-[**selection**: *SHA-384, SHA-512*], with iteration count of [**assignment**: *number of iterations*] using a randomly generated salt of length [**assignment**: *equal to or greater than 128*] and output cryptographic key sizes [**selection**: *256, 384, 512*] bits that meet the following standard: [NIST SP 800-132 (Section 5.3) [PBKDF2]].

Application Note: NIST recommends a minimum "number of iterations" of 1000 but prefers the largest number feasible given performance constraints.

NIST recommends that the randomly generated portion of the salt have length of at least 128 bits and must be derived from Random Bit Generation.

If this SFR is claimed, then FCS_COP.1/KeyedHash and FCS_RBG.1 must also be claimed.

For CNSA compliance, only SHA-384 or SHA-512 may be used.

Evaluation Activities

FCS_CKM_EXT.8
TSS
The evaluator must verify that the TSS documents that the selection of the keyed hash algorithm, iteration count, length of salt, and other mitigations are sufficient for the security strength of the key derived.

The evaluator shall examine the TSS to verify that the salt is generated in accordance with the relevant specification.

The evaluator shall examine the TSS to determine whether the TOE implements other mitigations against password-exhaustion attacks. Examples include the use of interface-based mitigations and secret salts.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

To test the TOE's ability to derive cryptographic keys from a password using PBKDF2 the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *HMAC algorithms [SHA-384, SHA-512]*
- *Iteration count [1-10000000]*
- *Derived Key size [256, 384, 512] bits*
- *Password length [8-128] bytes*
- *Salt length [128-4096] bits in multiples of 8.*

Algorithm Functional Test

For each supported HMAC algorithm, the evaluator shall generate 50 test cases using supported values for the above parameters such that

- *All supported derived key sizes are tested at least 10 times,*
- *Iteration counts are random values between the supported minimum and maximum values, with the supported minimum and maximum tested at least once each,*
- *Passwords are random byte strings representing upper- and lower-case letters of random supported lengths such that the minimum and maximum lengths are tested at least once, and*
- *Salts are random values between the supported minimum and maximum lengths such that the supported minimum and maximum lengths are both tested at least once.*

The evaluator shall compare the resulting keys from each test case with keys derived using a known-good implementation with the same input parameters.

FCS_STG_EXT.1 Protected Storage

FCS_STG_EXT.1.1

The TSF shall provide [**selection**: *mutable hardware-based*, *immutable hardware-based*, *software-based*] protected storage for asymmetric private keys and [**selection**: *symmetric keys*, *persistent secrets*, *no other keys*].

Application Note: This SFR should be included in the ST if the TOE provides protected storage as a service for tenant software, or if it stores keys or other persistent secrets for its own use.

This SFR must be claimed if the TOE includes a Dedicated Security Component that provides storage services, such as a TPM.

If the protected storage is implemented in software that is protected as required by FCS_STG_EXT.2, the ST Author is expected to select "*software-based*." If "*software-based*" is selected, the ST Author is expected to select "*software-based key storage*" in FCS_STG_EXT.2 and also claim FCS_STG_EXT.3.

If this SFR is included in the ST, then FCS_CKM.6 must also be claimed.

FCS_STG_EXT.1.2

The TSF shall support the capability of [**selection**:

- *importing keys/secrets into the TOE*
- *causing the TOE to generate [**selection**: asymmetric, symmetric]keys/secrets*

] upon request of [**selection**: *a client application, an administrator*].

Application Note: If "causing the TOE to generate keys/secrets" is selected in FCS_STG_EXT.1.2, then the ST must include at least one of FCS_CKM.1/AKG or FCS_CKM.1/SKG depending on the value of the internal selection.

FCS_STG_EXT.1.3

The TSF shall be capable of destroying keys/secrets in the protected storage upon request of [**selection**: *a client application, an administrator*].

Evaluation Activities

FCS_STG_EXT.1
TSS
The evaluator shall review the TSS to determine that the TOE implements the required protected storage. The evaluator shall ensure that the TSS contains a description of the protected storage mechanism that justifies the selection of mutable hardware-based or software-based.

Guidance
The evaluator shall examine the AGD to ensure that it describes the process for generating keys, importing keys, or both, based on what is claimed by the ST. The evaluator shall also examine the AGD to ensure that it describes the process for destroying keys that have been imported or generated.

Tests
The evaluator shall test the functionality of each security function as described below. If the TOE supports both import and generation of keys, the evaluator shall repeat the testing as needed to demonstrate that the keys resulting from both operations are treated in the same manner. The devices used with the tooling may need to be non-production devices in order to enable the execution of testing and gathering of evidence.

- *Test FCS_STG_EXT.1:1: The evaluator shall import or generate keys/secrets of each supported type according to the operational guidance. The evaluator shall write, or the developer shall provide access to, an application that generates a key/secret of each supported type and calls the import functions. The evaluator shall verify that no errors occur during import.*
- *Test FCS_STG_EXT.1:2: The evaluator shall write, or the developer shall provide access to, tenant software that uses a generated or imported key/secret:*
  - *For RSA, the secret shall be used to sign data.*
  - *For ECDSA, the secret shall be used to sign data.*

  The evaluator shall verify that the tenant software is able to access and use the key/secret as described. Additionally, the evaluator shall create arbitrary authorization data and key/secret storage identifiers, and check that the created data and identifiers do not permit access to other keys/secrets that have not been imported or generated by the evaluator or which would not be available to a TOE user. In other words, the evaluator shall test that imported keys/secrets do not reveal, destroy, or modify keys/secrets belonging to other users or otherwise restricted by the TOE itself.
- *Test FCS_STG_EXT.1:3: The evaluator shall destroy keys/secrets of each supported type according to the operational guidance. The evaluator shall write, or the developer shall provide access to, tenant software that destroys an imported or generated key/secret. The evaluator shall verify that the tenant software is able to cause the deletion of only keys that were created or imported on its behalf. The keys must not be recoverable following deletion.*

A.1.4 Class: User Data Protection (FDP)

FDP_TEE_EXT.1 Trusted Execution Environment for Tenant Software

FDP_TEE_EXT.1.1

The TSF shall implement a trusted execution environment that conforms to the following standard: [*Advanced Trusted Environment: OMTP TR1 v1.1*] and make this TEE available to tenant software.

Application Note: This SFR should be claimed in the ST if the TOE includes a trusted execution environment for the use of tenant software.

Evaluation Activities

FDP_TEE_EXT.1
TSS
The evaluator shall examine the TSS to ensure that it describes the protections provided by the TOE's TEE implementation.

Guidance
The evaluator shall examine the AGD to ensure that it describes the steps required for tenant software to invoke the TEE.

Tests
There are no test activities for this component.

A.1.5 Class: Identification and Authentication (FIA)

FIA_TRT_EXT.1 Authentication Throttling

FIA_TRT_EXT.1.1

The TSF shall limit user authentication attempts by [**selection**: *preventing authentication via an external port, enforcing a delay between incorrect authentication attempts*] for all authentication mechanisms selected in FIA_UAU.5.1.

FIA_TRT_EXT.1.2

The minimum delay between incorrect authentication attempts shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

Application Note: This SFR should be included in the ST if the TOE implements a mechanism for limiting the number or frequency of Administrator authentication attempts.

The authentication throttling applies to all authentication mechanisms selected in FIA_UAU.5.1. The user authentication attempts in this requirement are attempts to guess the Authentication Factor. The developer can implement the timing of the delays in the requirements using unequal or equal timing of delays. The minimum delay specified in this requirement provides defense against brute forcing.

Evaluation Activities

FIA_TRT_EXT.1
TSS
The evaluator shall verify that the TSS describes the method by which authentication attempts are not able to be automated. The evaluator shall ensure that the TSS describes either how the TSF disables authentication via external interfaces (other than the ordinary user interface) or how authentication attempts are delayed in order to slow automated entry and shall ensure that no more than 10 attempts can be attempted per 500 milliseconds for all authentication mechanisms selected in FIA_UAU.5.1.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
There are no test activities for this component.

A.2 Objective Requirements

A.2.1 Auditable Events for Objective Requirements

Table 8: Auditable Events for Objective Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FPT_ROT_EXT.3	Detection of attempted intrusion.	None.

A.2.2 Class: Protection of the TSF (FPT)

FPT_ROT_EXT.3 Hardware component integrity

FPT_ROT_EXT.3.1

Outside of the integrity root specified in FPT_ROT_EXT.1, the integrity of [**assignment**: *critical platform hardware components*] shall be verified prior to execution or use through: [**assignment**: *method for ensuring integrity of platform hardware components*].

Application Note: The purpose of this objective requirement is to encourage platform and component vendors to adopt mechanisms similar to those defined in NIST SP 1800-34 for ensuring the integrity of the hardware supply chain. The scope of SP 1800-34 is to cover "manufacturing and OEM processes that protect against counterfeits, tampering, and insertion of unexpected software and hardware, and the corresponding customer processes that verify that client and server computing devices and components have not been tampered with or otherwise modified. Manufacturing processes that cannot be verified by the customer are explicitly out of scope."

As a basic step, SP 1800-34 specifies that critical platform components should include immutable hardware IDs that can be listed in a hardware component manifest that is provided to the purchaser and signed by the manufacturer. It should then be possible for the TOE to verify the signature on the manifest and check that each hardware ID in the manifest matches the IDs in the actual hardware. The component manifest and hardware IDs provide proof of provenance for the TOE and its hardware components.

For purposes of this requirement, hardware identities can be verified once on first boot, on every boot, when new hardware is detected, or during normal operation of the platform - as long as the hardware integrity is verified before the component or device is used.

The ST Author lists the hardware components for which the integrity is checked, and the methods used for conducting the checks. "Critical components" generally would include chassis, motherboards, CPUs, network cards, memory chips, hard drives, controllers, graphics processors, and service controllers.

FPT_ROT_EXT.3.2

The TOE shall take the following actions if an integrity check specified in FPT_ROT_EXT.3.1 fails:

1. Halt,
2. Notify an [**selection**: *Administrator, User*] by [**selection**: *generating an audit event*, *[**assignment**: other notification method(s)]*], and
3. [**selection, choose one of**:
    - *Stop all execution and shut down*
    - *Continue execution without the integrity-compromised component*
    - *Continue execution*

    ]
    [**selection, choose one of**:
    - *in accordance with administrator-configurable policy*
    - *by express determination of an [**selection**: Administrator, User]*

    ]

.

Application Note: Notification of an administrator can take many forms. For server-class platforms, such notification could take the form of administrator alerts or audit events. For platforms without management controllers, notification could be achieved, for example, by blinking lights, beep codes, screen indications, or local logging.

If "*administrator*" is selected anywhere in FPT_ROT_EXT.3.2, or if "*in accordance with administrator-configurable policy*" is selected, then all administrator authentication requirements must be included in the ST (FIA_UIA_EXT.1, FIA_UAU.5, FIA_PMG_EXT.1, FIA_AFL_EXT.1, FIA_UAU.7).

If "*generating an audit event*" is selected, then FAU_GEN.1, FAU_SAR.1, FAU_STG.1, FAU_STG.2, and FAU_STG.5, must be included in the ST.

If "*in accordance with administrator-configurable policy*" is selected, then FMT_MOF.1 and FMT_SMF.1 must be claimed in the ST.

Evaluation Activities

FPT_ROT_EXT.3
TSS
The evaluator shall verify that the TSS describes the means by which integrity of platform hardware and firmware is maintained from TOE manufacture to delivery of the TOE to its operational site. The TSS shall also describe how the TOE

responds to failure of an integrity check consistent with the selections in FPT_ROT_EXT.3.2.

Guidance
The evaluator shall examine the AGD to ensure that it describes the actions taken and notification methods used in case of detection of a platform integrity violation. If the actions are configurable, the AGD shall explain how they are configured.

Tests
There are no test activities for this component.

A.3 Implementation-dependent Requirements

This PP does not define any Implementation-dependent requirements.

Appendix B - Selection-based Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements based on selections in the body of the PP: if certain selections are made, then additional requirements below must be included.

B.1 Auditable Events for Selection-based Requirements

Table 9: Auditable Events for Selection-based Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	No events specified	N/A
FAU_STG.1	On failure of logging function, capture record of failure and record upon restart of logging function.	None.
FAU_STG.2	No events specified	N/A
FAU_STG.5	Audit trail full. Overwrite of audit records is commencing.	None.
FCS_CKM.1/AKG	No events specified	N/A
FCS_CKM.1/SKG	No events specified	N/A
FCS_CKM.2	No events specified	N/A
FCS_CKM.6	No events specified	N/A
FCS_CKM_EXT.7	No events specified	N/A
FCS_COP.1/AEAD	No events specified	N/A
FCS_COP.1/CMAC	No events specified	N/A
FCS_COP.1/Hash	No events specified	N/A
FCS_COP.1/KeyEncap	No events specified	N/A
FCS_COP.1/KeyWrap	No events specified	N/A
FCS_COP.1/KeyedHash	No events specified	N/A
FCS_COP.1/SKC	No events specified	N/A
FCS_COP.1/SigGen	No events specified	N/A
FCS_COP.1/SigVer	No events specified	N/A
FCS_COP.1/XOF	No events specified	N/A
FCS_HTTPS_EXT.1	Failure to establish a HTTPS Session.	• Reason for failure. • Non-TOE endpoint of connection (IP address) for failures.
	Establishment/Termination of a HTTPS session.	Non-TOE endpoint of connection (IP address).
FCS_IPSEC_EXT.1	Failure to establish an IPsec SA.	• Reason for failure. • Non-TOE endpoint of connection (IP address).

	Establishment/Termination of an IPsec SA.	Non-TOE endpoint of connection (IP address).
FCS_RBG.1	Failure of the randomization process	None.
FCS_RBG.2	No events specified	N/A
FCS_RBG.3	No events specified	N/A
FCS_RBG.4	No events specified	N/A
FCS_RBG.5	No events specified	N/A
FCS_RBG.6	No events specified	N/A
FDP_ITC_EXT.1	No events specified	N/A
FIA_AFL_EXT.1	Failed attempt at Administrator authentication.	None.
FIA_PMG_EXT.1	No events specified	N/A
FIA_UAU.5	No events specified	N/A
FIA_UAU.7	No events specified	N/A
FIA_UIA_EXT.1	All use of the authentication mechanism.	Provided user identity, origin of the attempt (e.g. console, remote IP address).
FPT_FLS.1	Failure of the TSF.	None.
FPT_PHP.1	**[selection:** *Detection of intrusion., None***]**	None.
FPT_PHP.2	**[selection:** *Detection of intrusion., None***]**	None.
FPT_PHP.3	Detection of attempted intrusion.	None.
FPT_RVR_EXT.1	No events specified	N/A
FPT_TST.1	Execution of self-tests.	None.
FPT_TUD_EXT.2	**[selection:** *Failure of update authentication/integrity check/rollback, None***]**	Version numbers of the current firmware and of the attempted update.
	[selection: *Failure of update operation, None***]**	Version numbers of the current firmware and of the attempted update.
	[selection: *Success of update operation, None***]**	Version numbers of the new and old firmware images.
FPT_TUD_EXT.3	**[selection:** *Failure of update authentication/integrity/rollback check, None***]**	Version numbers of the current firmware and of the attempted update.
	[selection: *Failure of update operation, None***]**	Version numbers of the current firmware and of the attempted update.
	[selection: *Success of update operation, None***]**	Version numbers of the new and old firmware images.
FPT_TUD_EXT.4	No events specified	N/A
FTP_ITC_EXT.1	Initiation of the trusted channel.	User ID and remote source (IP Address) if feasible.
	Termination of the trusted channel.	User ID and remote source (IP Address) if feasible.
	Failures of the trusted path functions.	User ID and remote source (IP Address) if feasible.

FTP_ITE_EXT.1	No events specified	N/A
FTP_ITP_EXT.1	No events specified	N/A
FTP_TRP.1	Initiation of the trusted channel.	Administrator ID and remote source (IP Address), if feasible.
	Termination of the trusted channel.	Administrator ID and remote source (IP Address), if feasible.
	Failures of the trusted path functions.	User ID and remote source (IP Address), if feasible.

B.2 Class: Security Audit (FAU)

FAU_GEN.1 Audit Data Generation

The inclusion of this selection-based component depends upon selection in:

- ***FPT_ROT_EXT.2.2,***
- ***FPT_ROT_EXT.3.2***

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Physically Secure Environment***
- ***Server-Class Platform, Enhanced Security Requirements***
- ***CSfC EUD***
- ***Enterprise Desktop Clients***

FAU_GEN.1.1

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

1. Start-up and shutdown of the audit functions
2. All administrative actions
3. Start-up, shutdown, and reboot of the platform
4. Specifically defined auditable events in Table 2
5. [**selection**:
   - *Specifically defined auditable event in Table 6 for Strictly Optional requirements*
   - *Specifically defined auditable event in Table 8 for Objective requirements*
   - *Specifically defined auditable event in Table 9 for Selection-based requirements*
   - *Additional information defined in the audit table for the Functional Package for Transport Layer Security (TLS), version 2.1*
   - *Additional information defined in the audit table for the Functional Package for Secure Shell (SSH), version 2.0*
   - *Additional information defined in the audit table for the Functional Package for X.509, version 1.0*
   - *no additional auditable events*

   ].

FAU_GEN.1.2

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

a. Date and time of the event
b. Type of event
c. Subject and object identity (if applicable)
d. The outcome (success or failure) of the event
e. [**Additional information defined in Table 2**]
f. **[selection:**
   - ***Additional information defined in Table 6 for Strictly Optional SFRs***
   - ***Additional information defined in Table 8 for Objective SFRs***
   - ***Additional information defined in Table 9 for Selection-Based SFRs***
   - ***Additional information defined in the audit table for the Functional Package for Transport Layer Security (TLS), version 2.1***
   - ***Additional information defined in the audit table for the Functional Package for Secure Shell (SSH), version 2.0***

- ***Additional information defined in the audit table for the Functional Package for X.509, version 1.0***
- ***no other information***

].

Application Note: The ST Author should include this SFR in the ST if the TOE generates audit events for integrity verification or boot failures as indicated by the appropriate selections in FPT_ROT_EXT.2, FPT_ROT_EXT.3, FPT_TUD_EXT.2, or FPT_TUD_EXT.3.4; or if the TOE supports the Server (basic or enhanced), CSfC EUD, or Enterprise Desktop use cases.

If this SFR is included in the ST, then all the other FAU SFRs must also be claimed.

Appropriate entries from Table 6, Table 8, and Table 9 should be included in the ST if the associated SFRs and selections are included.

Specific auditable events required for SFRs from the functional packages are defined in the respective packages.

Evaluation Activities

FAU_GEN.1
TSS
The evaluator shall check the TSS and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type shall be covered, along with a brief description of each field.

Guidance
The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP. The evaluator shall examine the AGD and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements claimed in the ST. The evaluator shall document the methodology or approach taken while determining which actions in the AGD are security-relevant with respect to this PP.

Tests
The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed and administrative actions. For administrative actions, the evaluator shall test that each action determined by the evaluator above to be security relevant in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

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

FAU_SAR.1 Audit Review

This component must be included in the ST if any of the following SFRs are included:
- ***FAU_GEN.1***

FAU_SAR.1.1

The TSF shall provide **the Administrator** with the capability to read **all audited events and record contents** from the audit data.

FAU_SAR.1.2

The TSF shall provide the audit data in a manner suitable for the **Administrator** to interpret the information.

Application Note: This SFR must be included in the ST if FAU_GEN.1 is claimed.

Evaluation Activities

FAU_SAR.1
TSS
There are no additional TSS evaluation activities for this component.

Guidance

The evaluator shall review the AGD for the procedure on how to review the audit records.

Tests
The evaluator shall verify that the audit records provide all of the information specified in FAU_GEN.1 and that this information is suitable for human interpretation. The evaluation activity for this requirement is performed in conjunction with the evaluation activity for FAU_GEN.1.

FAU_STG.1 Audit Data Storage Location

This component must be included in the ST if any of the following SFRs are included:

- ***FAU_GEN.1***

FAU_STG.1.1

The TSF shall be able to[**selection**: *store audit data on the TOE itself, transmit audit data to an external IT entity using a trusted channel in accordance with FTP_ITC_EXT.1, write audit data to removable media under administrative control in accordance with FIA_UIA_EXT.*]

Application Note: The ST Author selects "*trusted channel*" and includes FTP_ITC_EXT.1 in the ST if the TOE offloads audit data to external IT entity over a network connection. Protocols used for implementing the trusted channel must be selected in FTP_ITC_EXT.1.

The ST Author selects "*removable media*" if the TOE supports offload of audit data using removable media such as thumb drives or disks. Note that the CSfC Use Case prohibits the use of removable media.

Evaluation Activities

FAU_STG.1.1
TSS
The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server.

Guidance
If "trusted channel" is selected above, the evaluator shall examine the AGD to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server. Furthermore, it must describe whether the transfer mechanism is periodic or continuous, and what happens in the event of a loss of connectivity.

If "removable media" is selected, the evaluator shall ensure that the AGD describes the process for accessing audit data and copying it to media. The AGD must also include high-level guidance on how frequently this operation may need to be done to minimize risk of data loss.

Tests
If "trusted channel" is selected above, testing of the trusted channel mechanism itself is to be performed as specified in the evaluation activities for FTP_ITC_EXT.1. In addition, the evaluator must perform the following test:

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

If "removable media" is selected above, the evaluator must run the system for a time long enough to generate some audit data and then collect audit data onto removable media for transfer to another machine. On another machine, the evaluator shall examine the audit data to ensure that it appears to be complete and correct. This test may be performed in conjunction with any other requirement that generates audit events.

FAU_STG.2 Protected Audit Trail Storage

This component must be included in the ST if any of the following SFRs are included:

- ***FAU_GEN.1***

FAU_STG.2.1

The TSF shall protect the stored audit data in the audit trail from unauthorized deletion.

FAU_STG.2.2

The TSF shall be able to [*prevent*] unauthorized modifications to the stored audit data in the audit trail.

Application Note: Deletion of audit data within the TOE is "authorized" if the deletion is initiated or performed by an Administrator.

Notwithstanding this requirement, audit records may be overwritten if local audit record storage is full in accordance with FAU_STG.5.

This SFR must be included in the ST if FAU_GEN.1 is claimed.

Evaluation Activities

FAU_STG.2
TSS
The evaluator shall ensure that the TSS lists the locations of all logs and the access controls of those files such that unauthorized modification and deletion are prevented.

Guidance
The evaluator shall ensure that the Guidance describes the steps necessary for an authorized administrator to delete audit records, if such a capability is implemented.

Tests
The evaluator shall perform the following tests:

- *Test FAU_STG.2:1: [conditional] If the TOE implements an audit record deletion capability, then the evaluator shall attempt to delete the audit trail in a manner that the access controls should prevent (as an unauthorized user) and shall verify that the attempt fails.*
- *Test FAU_STG.2:2: The evaluator shall attempt to modify the audit trail in a manner that the access controls should prevent (as an unauthorized application) and shall verify that the attempt fails.*

FAU_STG.5 Prevention of Audit Data Loss

This component must be included in the ST if any of the following SFRs are included:

- ***FAU_GEN.1***

FAU_STG.5.1

The TSF shall optionally notify the administrator or user that storage is full and [*overwrite the oldest stored audit records*] if the audit data storage is full.

Application Note: This SFR must be included in the ST if FAU_GEN.1 is claimed.

Evaluation Activities

FAU_STG.5
TSS
The evaluator shall examine the TSS to ensure that it describes the size limits on the audit records, the detection of a full audit trail, and the action(s) taken by the TSF when the audit trail is full. The evaluator shall ensure that the action(s) results in the deletion or overwrite of the oldest stored record.

Guidance
The evaluator shall examine the AGD to ensure that it describes the means used by the TOE to indicate that the audit trail is full and overwrite is about to commence.

Tests
The evaluator shall cause audit records to be written until the size limits are met and exceeded. The evaluator shall verify that the overwrite function works as described in the TSS and that the indication of full audit trail is evident as described in

the AGD.

B.3 Class: Cryptographic Support (FCS)

FCS_CKM.1/AKG Cryptographic Key Generation - Asymmetric Key

The inclusion of this selection-based component depends upon selection in:

- ***FCS_STG_EXT.1.2,***
- ***FTP_ITC_EXT.1.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_CKM_EXT.7***
- ***FCS_COP.1/KeyEncap***
- ***FCS_COP.1/SigGen***
- ***FCS_IPSEC_EXT.1***

This component may also be included in the ST as if optional.

FCS_CKM.1.1/AKG

The TSF shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [**selection**: *Cryptographic key generation algorithm*] and specified cryptographic **algorithm parameters** ~~key sizes~~ [**selection**: *Cryptographic algorithm parameters*] that meet the following: [**selection**: *List of standards*] .

Table 10 provides the allowable choices for completion of the selection operations of FCS_CKM.1/AKG.

Table 10: Allowable choices for FCS_CKM.1/AKG

Identifier	Cryptographic key generation algorithm	Cryptographic algorithm parameters	List of standards
RSA	RSA	Modulus of size [**selection**: *3072, 4096, 6144, 8192*] bits	NIST FIPS PUB 186-5 (Section A.1.1)
ECC-ERB	ECC-ERB - Extra Random Bits	Elliptic Curve [**selection**: *P-384, P-521*]	FIPS PUB 186-5 (Section A.2.1) NIST SP 800-186 (Section 3) [NIST Curves]
ECC-RS	ECC-RS - Rejection Sampling	Elliptic Curve [**selection**: *P-384, P-521*]	FIPS PUB 186-5 (Section A.2.2) NIST SP 800-186 (Section 3) [NIST Curves]
FFC-ERB	FFC-ERB - Extra Random Bits	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]* • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.3) [key pair generation] [**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
FFC-RS	FFC-RS - Rejection Sampling	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]*	NIST SP 800-56A Revision 3 (Section 5.6.1.1.3) [key pair generation]

		• *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	[**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
LMS	LMS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Winternitz parameter = [**selection**: *1, 2, 4, 8*] Tree height = [**selection**: *5, 10, 15, 20, 25*]	RFC 8554 [LMS] NIST SP 800-208 [parameters]
ML-KEM	ML-KEM KeyGen	Parameter set = ML-KEM-1024	NIST FIPS 203 (Section 7.1)
ML-DSA	ML-DSA KeyGen	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.1)
XMSS	XMSS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Tree height = [**selection**: *10, 16, 20*]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]

Application Note: This SFR must be included in the ST if asymmetric key generation is a service provided by the TOE to tenant software, or if it is used by the TOE itself to support or implement PP-specified security functionality.

This SFR must also be claimed in the ST if FCS_IPSEC_EXT.1 is claimed, or if "*causing the TOE to generate [asymmetric] keys/secrets*" is selected in FCS_STG_EXT.1.2.

Furthermore, this SFR must be claimed if TLS or HTTPS is claimed in FTP_ITC_EXT.1.

If this SFR is included in the ST, then FCS_CKM.6 and FCS_RBG.1 must also be claimed.

For RSA the choice of the modulus implies the resulting key sizes of the public and private keys generated using the specified standard methods. RSA key generation with modulus size 2048 bits is no longer permitted by CNSA.

For Finite Field Cryptography (FFC) DSA, ST authors should consult schemes for guidelines on use. FIPS PUB 186-5 does not approve DSA for digital signature generation but allows DSA for digital signature verification for legacy purposes. "FFC-ERB" or "FFC–RS" may be claimed only for generating private and public keys when "DH" is claimed in FCS_CKM_EXT.7.

When generating ECC keys pairs for key agreement and if "ECDH" is claimed in FCS_CKM_EXT.7, then "ECC–ERB" or "ECC–RS" must be claimed. The sizes of the private key, which is a scalar, and the public key, which is a point on the elliptic curve, are determined by the choice of the curve.

When generating ECC key pairs for digital signature generation and if "ECDSA" is claimed in FCS_COP.1/SigGen, then "ECC–ERB" or "ECC–RS" must be claimed. The sizes of the private key, which is a scalar, and the public key, which is a point on the elliptic curve, are determined by the choice of the curve.

Evaluation Activities

FCS_CKM.1/AKG
TSS
The evaluator shall examine the TSS to verify that it describes how the TOE generates a key based on output from a random bit generator as specified in FCS_RBG.1. The evaluator shall review the TSS to verify that it describes how the functionality described by FCS_RBG.1 is invoked.

The evaluator shall examine the TSS to verify that it identifies the usage, and key lifecycle for keys generated using each selected algorithm.

The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed in accordance with the relevant standards.

If the TOE uses the generated key in a key chain/hierarchy then the evaluator shall verify that the TSS describes how the key is used as part of the key chain/hierarchy.

Guidance
The evaluator shall verify that the Guidance instructs the administrator how to configure the TOE to generate keys for the selected key generation algorithms for all key types and uses identified in the TSS.

Tests
The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

RSA Key Generation

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
RSA	RSA	Modulus of size [**selection:** 3072, 4096, 6144, 8192] bits	NIST FIPS PUB 186-5 (Section A.1.1)

FIPS PUB 186-5 Key Pair generation specifies five methods for generating the primes p and q.

These are:

1. *Random Provable primes*
2. *Random Probable primes*
3. *Provable primes with conditions based on auxiliary provable primes*
4. *Probable primes with conditions based on auxiliary provable primes*
5. *Probable primes with conditions based on auxiliary probable primes*

In addition to the key generation method, the input parameters are:

- *Modulus [3072, 4096, 6144, 8192]*
- *Hash algorithm [SHA-384, SHA-512] (methods 1, 3, 4 only)*
- *Rabin-Miller prime test [2^{100}, $2^{Security\ String}$] (methods 2, 4, 5 only)*
- *p mod 8 value [0,1,3,5,7]*
- *q mod 8 value [0,1,3,5,7]*
- *Private key format [standard, Chinese Remainder Theorem]*
- *Public exponent [fixed value, random]*

The evaluator shall verify the ability of the TSF to correctly produce values for the RSA key components, including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Testing for Random Provable Primes and Conditional Methods

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods (methods 1, 3-5), the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair.

For each supported combination of the above input parameters, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Testing for Random Probable Primes Method

If the TOE generates Random Probable Primes (method 2) then, if possible, the Random Probable primes method should also be verified against a known good implementation as described above. If verification against a known good implementation is not possible, the evaluator shall have the TSF generate 25 key pairs for each supported key length nlen and verify that all of the following are true:

- *n = p*q*
- *p and q are probably prime according to Miller-Rabin tests with error probability $<2^{(-125)}$*

- $2^{16} < e < 2^{256}$ *and e is an odd integer*
- *GCD(p-1,e) = 1*
- *GCD(q-1,e) = 1*
- $|p-q| > 2^{(nlen/2 - 100)}$
- $p \geq squareroot(2)*(2^{(nlen/2 -1)})$
- $q \geq squareroot(2)*(2^{(nlen/2 -1)})$
- $2^{(nlen/2)} < d < LCM(p-1,q-1)$
- *e*d = 1 mod LCM(p-1,q-1)*

Elliptic Curve Key Generation

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
ECC-ERB	ECC - Extra Random Bits	Elliptic Curve [**selection:** P-384, P-521]	NIST FIPS PUB 186-5 (Section A.2.1) NIST SP 800-186 (Section 3) [NIST Curves]
ECC-RS	ECC - Rejection Sampling	Elliptic Curve [**selection:** P-384, P-521]	NIST FIPS PUB 186-5 (Section A.2.2) NIST SP 800-186 (Section 3) [NIST Curves]

To test the TOE's ability to generate asymmetric cryptographic keys using elliptic curves, the evaluator shall perform the ECC Key Generation Test and the ECC Key Validation Test using the following input parameters:

- *Elliptic curve [P-384, P-521]*
- *Key pair generation method [extra random bits, rejection sampling]*

ECC Key Generation Test
For each supported combination of the above input parameters the evaluator shall require the implementation under test to generate 10 private/public key pairs (d, Q). The private key, d, shall be generated using a random bit generator as specified in FCS_RBG.1. The private key, d, is used to compute the public key, Q'. The evaluator shall confirm that 0<d<n (where n is the order of the group), and the computed value Q' is then compared to the generated public/private key pairs' public key, Q, to confirm that Q is equal to Q'.

ECC Key Validation Test
For each supported combination of the above parameters the evaluator shall generate 12 private/public key pairs using the key generation function of a known-good implementation. For each set of 12 public keys, the evaluator shall modify four public key values by shifting x or y out of range by adding the order of the field and modify four other public key values by shifting x or y so that they are still in bounds, but not on the curve. The remaining public key values are left unchanged (i.e., correct). To determine correctness, the evaluator shall submit the public keys to the public key validation (PKV) function of the TOE and shall confirm that the results correspond as expected for the modified and unmodified values.

Finite Field Cryptography Key Generation

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
FFC-ERB	FFC – Extra Random Bits	Static domain parameters approved for [**selection:** IKE groups [**selection:** MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [**selection:** ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.3) [key pair generation] [**selection:** RFC 3526 [IKE groups], RFC 7919 [TLS groups]]
FFC-RS	FFC – Rejection Sampling	Static domain parameters approved for [**selection:** IKE groups [**selection:** MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [**selection:** ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.4) [key pair generation]

			[**selection:** RFC 3526 [IKE groups], RFC 7919 [TLS groups]]

To test the TOE's ability to generate asymmetric cryptographic keys using finite fields, the evaluator shall perform the Safe Primes Generation Test and the Safe Primes Validation Test using the following input parameter:

- *Fields/Groups [MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*

Safe Primes Generation Test
For each supported safe primes group, generate 10 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Safe Primes Verification Test
For each supported safe primes group, use a known good implementation to generate 10 key pairs. For each set of 10, the evaluator shall modify three such that they are incorrect. The remaining values are left unmodified (i.e. correct). To determine correctness, the evaluator shall submit the key pairs to the public key validation (PKV) function of the TOE and shall confirm that the results correspond as expected for the modified and unmodified values.

LMS Key Generation

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
LMS	LMS Key Generation	Private key size = [**selection:** 192 bits with [**selection:** SHA-256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]]; Winternitz parameter = [**selection:** 1, 2, 4, 8]; Tree height = [**selection:** 5, 10, 15, 20, 25]	RFC 8554 [LMS] NIST SP 800-208 [parameters]

To test the TOE's ability to generate asymmetric cryptographic keys using LMS, the evaluator shall perform the LMS Key Generation Test using the following input parameters:

- *Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]*
- *Winternitz [1, 2, 4, 8]*
- *Tree height [5, 10, 15, 20, 25]*

LMS Key Generation Test
For each supported combination of the hash algorithm, Winternitz parameter, and tree height the evaluator shall generate one public key for each of the test cases. The number of test cases depends on the tree height:

Table 11: Number of LMS Test Cases

Height	Number of test cases
5	5
10	4
15	3
20	2
25	1

The evaluator shall verify the correctness of the TSF's implementation by comparing the public key generated by the TSF with that generated by a known good implementation using the same input parameters.

ML-KEM Key Generation

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards

ML-KEM	ML-KEM Key Generation	Parameter set = [ML-KEM-1024]	NIST FIPS PUB 203 (Section 7.1)

To test the TOE's ability to generate asymmetric cryptographic keys using ML-KEM, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Parameter set [ML-KEM-1024]*
- *Random seed d [32 bytes]*
- *Random seed z [32 bytes]*

The random seeds d and z are normally generated internally by ML-KEM.KeyGen(). This test will actually be testing the TSF's internal key generation function on ML-KEM.KeyGen_internal(d, z) as described in FIPS 203, unless the test setup allows capturing the seeds.

Algorithm Functional Test
For each supported parameter set the evaluator shall require the implementation under test to generate 25 key pairs using 25 different randomly generated pairs of 32-byte seed values (d, z). To determine correctness, the evaluator shall compare the resulting key pairs (ek, dk) with those generated using a known-good implementation using the same inputs.

ML-DSA Key Generation

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
ML-DSA	ML-DSA Key Generation	Parameter set = ML-DSA-87	NIST FIPS PUB 204 (Section 5.1)

To test the TOE's ability to generate asymmetric cryptographic keys using ML-DSA, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Parameter set [ML-DSA-87]*
- *Random seed [32 bytes]*

The random seed is normally generated internally by ML-DSA.KeyGen(). This test will actually be testing the TSF's internal key generation function ML-DSA.KeyGen_internal(seed) as described in FIPS 204, unless the test setup allows capturing the seed.

Algorithm Functional Test
For each supported parameter set the evaluator shall require the implementation under test to generate 25 key pairs using 25 different randomly generated 32-byte seed values. To determine correctness, the evaluator shall compare the resulting key pairs with those generated using a known-good implementation using the same inputs.

XMSS Key Generation

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
XMSS	XMSS	Private key size = [**selection:** 192 bits with [**selection:** SHA-256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]], tree height = [**selection:** 10, 16, 20]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]

To test the TOE's ability to generate asymmetric cryptographic keys using XMSS, the evaluator shall perform the XMSS Key Generation Test using the following input parameters:

- *Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]*
- *Tree height [10, 16, 20] (XMSS only)*

Table 12: Number of Test Cases for XMSS

Height	Number of test cases
10	5
16	4
20	3
40	2

60	1

XMSS Key Generation Test
For each supported combination of hash algorithm and tree height, the evaluator shall generate one public key for each test case. The number of test cases depends on the tree height as specified in Table 12.

The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Note: The number of test cases is limited due to the extreme amount of time it can take to generate XMSS trees.

FCS_CKM.1/SKG Cryptographic Key Generation - Symmetric Key

The inclusion of this selection-based component depends upon selection in:

- ***FCS_STG_EXT.1.2,***
- ***FTP_ITC_EXT.1.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_COP.1/AEAD***
- ***FCS_COP.1/CMAC***
- ***FCS_COP.1/KeyedHash***
- ***FCS_COP.1/KeyWrap***
- ***FCS_COP.1/SKC***

This component may also be included in the ST as if optional.

FCS_CKM.1.1/SKG

The TSF shall generate **symmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [**selection**: *Cryptographic Key Generation Algorithm*] and specified cryptographic key sizes [**selection**: *Cryptographic Key Sizes*] that meet the following: [**selection**: *List of standards*] .

Table 13 provides the allowable choices for completion of the selection operations of FCS_CKM.1/SKG.

Table 13: Allowable choices for FCS_CKM.1/SKG

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Key Sizes	List of standards
RSK	Direct Generation from a Random Bit Generator as specified in FCS_RBG.1	[**selection**: *256, 384, 512*] bits	NIST SP 800-133 Revision 2 (Section 6.1)[Direct generation of symmetric keys]

Application Note: This SFR must be included in the ST if it is a service provided by the TOE to tenant software, or if it is used by the TOE itself to support or implement PP-specified security functionality.

This SFR must be included in the ST if "*causing the TOE to generate [symmetric] keys/secrets*" is selected in FCS_STG_EXT.1.2.

This SFR must be claimed if any SFRs are claimed that require generation of a symmetric key, such as FCS_COP.1/AEAD, FCS_COP.1/KeyedHash, FCS_COP.1/KeyWrap, FCS_COP.1/CMAC, or FCS_COP.1/SKC.

If this SFR is claimed in the ST, then FCS_CKM.6 and FCS_RBG.1 must also be claimed.

Evaluation Activities

FCS_CKM.1/SKG
TSS
The evaluator shall examine the TSS to verify that it describes how the TOE obtains a symmetric cryptographic key through direct generation from a random bit generator as specified in FCS_RBG.1. The evaluator shall review the TSS to verify that it describes how the functionality described by FCS_RBG.1 is invoked.

The evaluator shall examine the TSS to verify that it identifies the usage, and key lifecycle for keys generated using each selected algorithm.

If the TOE uses the generated key in a key chain/hierarchy then the evaluator shall verify that the TSS describes how the key is used as part of the key chain/hierarchy.

Guidance
The evaluator shall verify that the AGD instructs the administrator how to configure the TOE to use the RBG to generate symmetric keys for all uses identified in the ST.

Tests
The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

To test the TOE's ability to generate symmetric cryptographic keys using a random bit generator, the evaluator shall configure the symmetric cryptographic key generation capability for each claimed key size. The evaluator shall use the description of the RBG interface to verify that the TOE requests and receives an amount of RBG output greater than or equal to the requested key size.

FCS_CKM.2 Cryptographic Key Distribution

The inclusion of this selection-based component depends upon selection in:

- ***FTP_ITC_EXT.1.1,***
- ***FTP_ITE_EXT.1.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_IPSEC_EXT.1***

This component may also be included in the ST as if optional.

FCS_CKM.2.1

The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method [**selection**: *key encapsulation*, *key wrapping*, *encrypted channels*] that meets the following: [none].

Application Note: If "key encapsulation" is selected, FCS_COP.1/KeyEncap must be claimed, which specifies the relevant list of standards.

If "key wrapping" is selected, FCS_COP.1/KeyWrap must be claimed, which specifies the relevant list of standards.

If "encrypted channels" is selected, FTP_ITC_EXT.1 must be claimed.

Evaluation Activities

FCS_CKM.2
TSS
The evaluator shall ensure that the TSS documents that the security strength supported by the selected key distribution methods is sufficient for the security strength of the keys distributed through those methods.

It is not necessary to identify the services that use each key distribution method here. That information should be documented in the requirements for the individual services and protocols that invoke key distribution.

Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key distribution methods.

Tests
Specific testing for this component is covered by testing for the claimed components in FCS_COP.1/KeyEncap, FCS_COP.1/KeyWrap, or FTP_ITC_EXT.1.

FCS_CKM.6 Timing and Event of Cryptographic Key Destruction

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_CKM.1/AKG***
- ***FCS_CKM.1/SKG***
- ***FCS_CKM.2***
- ***FCS_CKM.5***
- ***FCS_CKM_EXT.7***
- ***FCS_CKM_EXT.8***
- ***FCS_STG_EXT.1***
- ***FIA_AFL_EXT.1***

FCS_CKM.6.1

The TSF shall destroy [**assignment**: *list of cryptographic keys (including keying material)*] when [**selection**: *no longer needed, as specified in NIST SP 800-57 Part 1 Rev. 5, [**assignment**: other circumstances for key or key material destruction]*].

Application Note: The purpose of key destruction is "To remove all traces of a cryptographic key so that it cannot be recovered by either physical or electronic means." (NIST SP 800-57 Part 1 Rev. 5).

The TOE implements methods for destroying keys and key material as specified in FCS_CKM.6.2.

The ST Author shall list all such keys and keying material that are subject to destruction in the first assignment. The description of each key must also include the types of memory or storage (e.g.: DRAM, SRAM, flash, etc.) where the key may be found while in use and between uses. Cryptographic material that is infeasible to destroy shall also be identified accompanied with a justification or explanation. Keys related to the operation of or under the protection of an FDE may be subject to requirements enumerated in the FDE cPPs.

FCS_CKM.6.2

The TSF shall destroy cryptographic keys and keying material specified by FCS_CKM.6.1 in accordance with a specified cryptographic key destruction method:

- For volatile memory, the destruction shall be executed by a [**selection**:
  - *single overwrite consisting of [**selection, choose one of**: a pseudorandom pattern using the TSF's RBG, zeroes, ones, a new value of a key, [**assignment**: some value that does not contain any CSP]]*
  - *Removal of power to the memory*
  - *Removal of all references to the key directly followed by a request for garbage collection*

  ]
- For non-volatile memory [**selection, choose one of**:
  - *that employs a wear-leveling algorithm, the destruction shall be executed by a [**selection**:*
    - *single overwrite consisting of [**selection, choose one of**: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [**assignment**: some value that does not contain any CSP]]*
    - *secure block erase*
    - *ONFI Sanitize command*
    - *vendor-specific secure erase command*
    - *cryptographic erase*

    ]
  - *that does not employ a wear-leveling algorithm, the destruction shall be executed by a [**selection**:*
    - *[**selection, choose one of**:*
      - *single pass*
      - *[**assignment**: ST Author-defined multi-pass]*

      ]
      *overwrite consisting of [**selection, choose one of**: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [**assignment**: some value that does not contain any CSP]] followed by a read-verify. If the read-verification of the overwritten data fails, the process shall be repeated again up to [**assignment**: number of times to attempt overwrite] times, whereupon an error is returned*
    - *block erase*

    ]

  ]

The above must be implemented in a way that conforms to NIST SP 800-88.

Application Note: This SFR must be included in the ST if the TOE handles sensitive cryptographic keys or credentials. In particular, if the TOE creates or stores keys, it must be able to destroy them. Specifically, this SFR must be included in the ST if any of the following SFRs are claimed: FCS_CKM.1/AKG, FCS_CKM.1/SKG, FCS_CKM.2, FCS_CKM.5, FCS_COP.1/KAT, FCS_STG_EXT.1, or FIA_AFL_EXT.1 .

The term "*cryptographic keys*" in this SFR includes the authorization data that is the entry point to a key chain and all other cryptographic keys and keying material (whether in plaintext or encrypted form). Examples of keys and key material include intermediate keys, encryption keys, signing keys, verification keys, authentication tokens, and submasks. This SFR does not apply to the public component of asymmetric key pairs, or to keys that are permitted to remain stored such as device identification keys. The TOE will have mechanisms to destroy keys and key material when the keys and key material is no longer needed. Based on their implementations, vendors will explain when certain keys are no longer needed.

In the case of volatile memory, the selection "*removal of all references to the key directly followed by a request for garbage collection*" is used in a situation where the TSF cannot address the specific physical memory locations holding the data to be erased and therefore relies on addressing logical addresses (which frees the relevant physical addresses holding the old data) and then requesting the platform to ensure that the data in the physical addresses is no longer available for reading (i.e. the "garbage collection" referred to in the SFR text).

The selection for destruction of data in non-volatile memory includes block erase as an option, and this option applies only to flash memory. A block erase does not require a read verify, since the mappings of logical addresses to the erased memory locations are erased as well as the data itself.

Some selections allow assignment of "*some value that does not contain any CSP*. " This means that the TOE uses some specified data not drawn from an RBG meeting FCS_RBG requirements, and not being any of the particular values listed as other selection options. The point of the phrase "*does not contain any CSP*" is to ensure that the overwritten data is carefully selected, and not taken from a general pool that might contain current or residual data that itself requires confidentiality protection.

When selecting a key destruction method for non-volatile memory that employs a wear leveling algorithm, cryptographic erase is the preferred option, especially when the key is used solely to decrypt user or keying material. For devices that support ONFI, the Security Target (ST) must include documentation demonstrating that the ONFI Sanitize command erases all physical instances of the target key. If a vendor-specific secure erase command is used instead, the ST must specify its opcodes, sequence of operations, and provide evidence of its effectiveness; the TOE must also be able to verify successful completion of the erase. If using secure block erase, the method must be FTL-aware, and capable of identifying and erasing all physical pages, including any stale or remapped copies of the key. Overwriting should only be selected if the NAND device has no wear leveling, or if wear leveling can be reliably disabled or bypassed through secure firmware.

Evaluation Activities

FCS_CKM.6

TSS

The evaluator shall examine the TSS to verify that it

- *lists all relevant keys and keying material,*
- *describes the source of any key material,*
- *documents all memory or storage media types in which the keys or keying material may be found both during use and between uses,*
- *documents all relevant destruction situations (including the point in time at which the destruction occurs; e.g. factory reset or device wipe function, change of authorization data, change of DEK, completion of use of an intermediate key), and*
- *describes the destruction method used in each case.*

The evaluator shall confirm that the description of the data and storage locations is consistent with the functions carried out by the TOE. Where keys are stored encrypted or wrapped under another key then this should be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions.

The evaluator shall verify that the TSS identifies any configurations or circumstances that may not conform to the key destruction requirement, such as delays in the destruction of keys in some configurations or circumstances.

Guidance

The evaluator shall verify that the Guidance describes any configurable parameters related to the lifecycle of cryptographic keys, including lifetimes or storage parameters.

Tests
The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The evaluator shall perform the following for each of the key destruction circumstance or trigger described in FCS_CKM.6.1:

- *Test FCS_CKM.6:1: If the key destruction method acts upon keys and keying material held in volatile memory that is subject to overwrite by the TOE (whether or not the plaintext key is subsequently encrypted for storage in volatile or non-volatile memory).*

  The evaluator shall:
  1. *Record the value of the key or keying material.*
  2. *Cause the TOE to perform normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to dump the appropriate memory into a binary file.*
  4. *Search the content of the binary file created in Step #2 to locate all instances of the known key value from Step #1.*

     Note that the primary purpose of Step #3 is to demonstrate that appropriate search commands are being used for Steps #8 and #9.
  5. *Cause the TOE to destroy the key by triggering the circumstance from FCS_CKM.6.1 currently being tested.*
  6. *Cause the TOE to stop execution but not exit.*
  7. *Cause the TOE to dump the appropriate memory into a binary file.*
  8. *Search the contents of the binary file created in Step #7 for instances of the known key value from Step #1.*
  9. *Break the key value from Step #1 into an evaluator-chosen set of fragments and perform a search using each fragment. (Note that the evaluator shall first confirm with the developer how the key is normally stored, in order to choose fragment sizes that are the same or smaller than any fragmentation of the data that may be implemented by the TOE. The endianness or byte-order should also be taken into account in the search.)*

  Steps #1-8 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

  Step #9 ensures that partial key fragments do not remain in memory. If the evaluator finds a 32-or-greater-consecutive-bit fragment, then fail immediately. Otherwise, there is a chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is also found in this repeated run, then the test fails unless the developer provides a reasonable explanation for the collision, then the evaluator may give a pass on this test.

- *Test FCS_CKM.6:2: If the key destruction method acts upon keys and keying material held in non-volatile memory that is subject to overwrite or erasure by the TOE.*
  1. *Record the value of the key or keying material.*
  2. *Cause the TOE to perform normal cryptographic processing with the key from Step #1.*
  3. *Search the non-volatile memory the key was stored in for instances of the known key value from Step #1.*

     Note that the primary purpose of Step #3 is to demonstrate that appropriate search commands are being used for Steps #5 and #6.
  4. *Cause the TOE to destroy the key by triggering the circumstance from FCS_CKM.6.1 currently being tested.*
  5. *Search the non-volatile memory in which the key was stored for instances of the known key value from Step #1. If a copy is found, then the test fails.*
  6. *Break the key value from Step #1 into an evaluator-chosen set of fragments and perform a search using each fragment. (Note that the evaluator shall first confirm with the developer how the key is normally stored, in order to choose fragment sizes that are the same or smaller than any fragmentation of the data that may be implemented by the TOE. The endianness or byte-order should also be taken into account in the search).*

  Step #6 ensures that partial key fragments do not remain in non-volatile memory. If the evaluator finds a 32-or-greater-consecutive-bit fragment, then fail immediately. Otherwise, there is a chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is also found in this repeated run, then the test fails unless the developer provides a reasonable explanation for the collision, then the evaluator may give a pass on this test.

FCS_CKM_EXT.7 Cryptographic Key Agreement

This component may also be included in the ST as if optional.

FCS_CKM_EXT.7.1

The TSF shall derive shared cryptographic keys with input from multiple parties in accordance with specified cryptographic key agreement algorithms [**selection**: *Cryptographic algorithm*] and specified cryptographic parameters [**selection**: *Cryptographic parameters*] that meet the following: [**selection**: *List of standards*] .

Table 14 provides the allowable choices for completion of the selection operations of FCS_CKM_EXT.7.

Table 14: Allowable choices for FCS_CKM_EXT.7

Identifier	Cryptographic algorithm	Cryptographic parameters	List of standards
DH	Finite Field Cryptography Diffie-Hellman	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]* • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	NIST SP 800-56A Revision 3 (Section 5.7.1.1) [DH] [**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
ECDH	Elliptic Curve Diffie-Hellman	Elliptic Curve [**selection**: *P-384, P-521*]	NIST SP 800-56A Revision 3 (Section 5.7.1.2) [ECDH] NIST SP 800-186 (Section 3.2.1) [NIST Curves]

Application Note: All of the above algorithms with the selectable parameters are CNSA compliant.

This SFR must be included in the ST if key agreement is a service provided by the TOE to tenant software, or if they are used by the TOE itself to support or implement PP-specified security functionality.

If this SFR is claimed, then FCS_CKM.6 an FCS_CKM.1/AKG must also be claimed.

This SFR has dependencies on FCS_COP.1/Hash and FCS_COP.1/XOF only if ML-KEM is selected.

Evaluation Activities

FCS_CKM_EXT.7
TSS
The evaluator shall ensure that the TSS documents that the security strength of the material contributed by the TOE is sufficient for the security strength of the key and the agreement method.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

FFC Diffie-Hellman Key Agreement

Identifier	Cryptographic Algorithm	Cryptographic Parameters	List of Standards
DH	Finite Field Cryptography Diffie-Hellman	Static domain parameters approved for [**selection:** IKE groups [**selection:** MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [**selection:** ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]]	NIST SP 800-56A Revision 3 (Section 5.7.1.1) [DH] [**selection:** RFC 3526 [IKE Groups], RFC 7919 [TLS Groups]]

To test the TOE's implementation of FFC Diffie-Hellman Key Agreement, the evaluator shall perform the Algorithm Functional Test and Validation Test using the following input parameters:

- *Domain Parameter Group [MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*

Algorithm Functional Test
For each supported domain parameter group, the evaluator shall generate 10 test cases by generating the initiator and responder secret keys using random data, calculating the responder public key, and creating the shared secret. The resulting shared secrets shall be compared with those generated by a known-good implementation using the same inputs.

Validation Test
For each supported combination of the above parameters the evaluator shall generate 15 Diffie Hellman initiator/responder key pairs using the key generation function of a known-good implementation. For each set of key pairs, the evaluator shall modify five initiator private key values. The remaining key values are left unchanged (i.e., correct). To determine correctness, the evaluator shall confirm that the 15 shared secrets correspond as expected for both the modified and unmodified inputs.

Elliptic Curve Diffie-Hellman Key Agreement

Identifier	Cryptographic Algorithm	Cryptographic Parameters	List of Standards
ECDH	Elliptic Curve Diffie-Hellman	Elliptic Curve [**selection:** P-384, P-521	NIST SP 800-56A Revision 3 (Section 5.7.1.2) [ECDH] NIST SP 800-186 (Section 3.2.1) [NIST Curves]

To test the TOE's implementation of Elliptic Curve Diffie-Hellman Key Agreement, the evaluator shall perform the Algorithm Functional Test and Validation Test using the following input parameters:

- *Elliptic Curve [P-384, P-521]*

Algorithm Functional Test
For each supported Elliptic Curve the evaluator shall generate 10 test cases by generating the initiator and responder secret keys using random data, calculating the responder public key, and creating the shared secret. The resulting shared secrets shall be compared with those generated by a known-good implementation using the same inputs.

Validation Test
For each supported Elliptic Curve the evaluator shall generate 15 Diffie Hellman initiator/responder key pairs using the key generation function of a known-good implementation. For each set of key pairs, the evaluator shall modify five initiator private key values. The remaining key values are left unchanged (i.e., correct). To determine correctness, the evaluator shall confirm that the 15 shared secrets correspond as expected for the modified and unmodified values.

FCS_COP.1/AEAD Cryptographic Operation – Authenticated Encryption with Associated Data

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.5.1*,**
- ***FTP_ITC_EXT.1.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_IPSEC_EXT.1***
- ***FCS_STG_EXT.2***

This component may also be included in the ST as if optional.

FCS_COP.1.1/AEAD

The TSF shall perform [*authenticated encryption with associated data*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic algorithm*] and cryptographic key sizes [**selection**: *Cryptographic key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 15 provides the allowable choices for completion of the selection operations of FCS_COP.1/AEAD.

Table 15: Allowable choices for FCS_COP.1/AEAD

Identifier	Cryptographic algorithm	Cryptographic key sizes	List of standards
AES-CCM	AES in CCM mode with unpredictable, non-repeating nonce, minimum size of 64 bits	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C*] [CCM]
AES-GCM	AES in GCM mode with non-repeating IVs using [**selection**: *deterministic, RBG-based*], IV construction; the tag must be of length [**selection**: *96, 104, 112, 120, 128*] bits.	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D*] [GCM]

Application Note: The use of 256-bit keys for AES encryption is required by CNSA.

Evaluation Activities

FCS_COP.1/AEAD

TSS

The evaluator shall examine the TSS to ensure that it describes the construction of any IVs, nonces, and tags in conformance with the relevant specifications.

If a CCM mode algorithm is selected, then the evaluator shall examine the TOE summary specification to confirm that it describes how the nonce is generated and that the same nonce is never reused to encrypt different plaintext pairs under the same key.

If a GCM mode algorithm is selected, then the evaluator shall examine the TOE summary specification to confirm that it describes how the IV is generated and that the same IV is never reused to encrypt different plaintext pairs under the same key. The evaluator shall also confirm that for each invocation of GCM, the length of the plaintext is at most (2^{32})-2 blocks.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

AES-CCM

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CCM	AES in CCM mode with nonrepeating nonce, minimum size of 64 bits	256 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [**selection:** ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C] [CCM]

To test the TOE's implementation of AES-CCM authenticated encryption functionality the evaluator shall perform the Algorithm Functional Tests described below using the following input parameters:

- *Key Size [256] bits*
- *Associated data size [0-65536] bits in increments of 8*

- *Payload size [0-256] bits in increments of 8*
- *IV/Nonce size [64-104] bits in increments of 8*
- *Tag size [32-128] bits in increments of 16*

Algorithm Functional Tests

Unless otherwise specified, the following tests should use random data, a tag size of 128 bits, IV/Nonce size of 104 bits, payload size of 256 bits, and associated data size of 256 bits. If any of these values are not supported, any supported value may be used. The evaluator shall compare the output from each test case against results generated by a known-good implementation with the same input parameters.

Variable Associated Data Test

For each claimed key size, and for each supported associated data size from 0 through 256 bits in increments of 8 bits, the TOE must be tested by encrypting 10 test cases using all random data. In addition, for each key size, the TOE must be tested by encrypting 10 cases with associated data lengths of 65536 bits, if supported.

Variable Payload Test

For each claimed key size, and for each supported payload size from 0 through 256 bits in increments of 8 bits, the TOE must be tested by encrypting 10 test cases using all random data.

Variable Nonce Test

For each claimed key size, and for each supported IV/Nonce size from 64 through 104 bits in increments of 8 bits, the TOE must be tested by encrypting 10 test cases using all random data.

Variable Tag Test

For each claimed key size, and for each supported tag size from 32 through 128 bits in increments of 16 bits, the TOE must be tested by encrypting 10 test cases using all random data.

Decryption Verification Test

For each claimed key size, for each supported associated data size from 0 through 256 bits in increments of 8 bits, for each supported payload size from 0 through 256 bits in increments of 8 bits, for each supported IV/Nonce size from 64 through 104 bits in increments of 8 bits, and for each supported tag size from 32 through 128 bits in increments of 16 bits, the TOE must be tested by decrypting 10 test cases using all random data.

AES-GCM

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-GCM	AES in GCM mode with nonrepeating IVs using [**selection:** deterministic, RBG-based] IV construction; the tag must be of length [**selection:** 96, 104, 112, 120, or 128] bits.	256 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [**selection:** ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D] [GCM]

To test the TOE's implementation of AES-GCM authenticated encryption functionality the evaluator shall perform the Encryption Algorithm Functional Tests and Decryption Algorithm Functional Tests as described below using the following input parameters:

- *Key Size [256] bits*
- *Associated data size [0-65536] bits*
- *Payload size [0-65536] bits*
- *IV size [96] bits*
- *Tag size [96, 104, 112, 120, 128] bits*

Encryption Algorithm Functional Tests

The evaluator shall generate 15 test cases using random data for each combination of the above parameters as follows:

- *Each claimed key size,*
- *Each supported tag size,*
- *Four supported non-zero payload sizes, such that two are multiples of 128 bits and two are not multiples of 128 bits,*

- *Four supported non-zero associated data sizes, such that two are multiples of 128 bits and two are not multiples of 128 bits, and*
- *An associated data size of zero, if supported.*

Note that the IV size is always 96 bits.

The evaluator shall compare the output from each test case against results generated by a known- good implementation with the same input parameters.

Decryption Algorithm Functional Tests

The evaluator shall test the authenticated decrypt functionality of AES-GCM by supplying 15 test cases for the supported combinations of the parameters as described above. For each parameter combination the evaluator shall introduce an error into either the Ciphertext or the Tag such that approximately half of the cases are correct and half the cases contain errors.

FCS_COP.1/CMAC Cryptographic Operation - CMAC

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.5.1,***
- ***FCS_STG_EXT.3.1***

This component may also be included in the ST as if optional.

FCS_COP.1.1/CMAC

The TSF shall perform [*CMAC*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic algorithm*] and cryptographic key sizes [**selection**: *Cryptographic key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 16 provides the allowable choices for completion of the selection operations of FCS_COP.1/CMAC.

Table 16: Allowable choices for FCS_COP.1/CMAC

Identifier	Cryptographic algorithm	Cryptographic key sizes	List of standards
AES-CMAC	AES using CMAC mode	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *: ISO/IEC 9797-1:2011 Subclause 7.6, NIST SP 800-38B*] [CMAC]

Application Note: The use of 256-bit keys for AES algorithms is required by CNSA.

Evaluation Activities

FCS_COP.1/CMAC
TSS
The evaluator shall examine the TSS to verify that the IV consists of all zeros in accordance with the relevant standards.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

AES-CMAC

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CMAC	AES using CMAC mode	256 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [**selection:** ISO/IEC 9797-1:2011 (Subclause 7.6), NIST SP 800-38B] [CMAC]

To test the TOE's ability to generate MAC values using AES in CMAC mode the evaluator shall perform the CMAC Generation Test and CMAC Verification Test using the following input parameters:

- *Key Size [256] bits*
- *Message size [0-524288] bits in increments of 8*
- *MAC sizes [1-128] bits*

CMAC Generation Test

The evaluator shall generate eight test cases using random keys and data for each combination of the above parameters as follows:

- *For each claimed key size,*
- *For four message sizes as follows:*
  - *The smallest supported message size,*
  - *The largest supported message size,*
  - *Two sizes that are divisible by the block size, and*
  - *Two sizes that are not divisible by the block size*
- *For three MAC sizes as follows:*
  - *The smallest supported MAC size,*
  - *The largest supported MAC size, and*
  - *Some other supported MAC size*

The evaluator shall compare the output from each test case against results generated by a known- good implementation with the same input parameters.

CMAC Verification Test

The evaluator shall generate 20 test cases using random keys and data for each combination of the above parameters as follows:

- *For each claimed key size,*
- *For four message sizes as follows:*
  - *The smallest supported message size,*
  - *The largest supported message size,*
  - *Two sizes that are divisible by the block size, and*
  - *Two sizes that are not divisible by the block size*
- *For three MAC sizes as follows:*
  - *The smallest supported MAC size,*
  - *The largest supported MAC size, and*
  - *Some other supported MAC size*

The evaluator shall modify the tag such that 25% of the test cases in each group of 20 test cases should fail.

The evaluator shall determine that the verification fails for the test cases with modified inputs, and succeeds for those with unmodified inputs.

FCS_COP.1/Hash Cryptographic Operation - Hashing

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.5.1,***
- ***FDP_ITC_EXT.1.2,***
- ***FPT_ROT_EXT.2.1,***
- ***FPT_TUD_EXT.2.1,***
- ***FTP_ITC_EXT.1.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_COP.1/KeyedHash***
- ***FCS_COP.1/SigGen***
- ***FCS_COP.1/SigVer***

This component may also be included in the ST as if optional.

FCS_COP.1.1/Hash

The TSF shall perform [*cryptographic hashing*] in accordance with a specified cryptographic algorithm [**selection**: *SHA-256, SHA-384, SHA-512, SHA3-384, SHA3-512*] that meet the following: [**selection**: *ISO/IEC 10118-3:2018 [SHA, SHA3], FIPS PUB 180-4 [SHA], FIPS PUB 202 [SHA3]*].

Application Note: In accordance with CNSA:

- SHA-1 hash is no longer permitted to be used as a hash function,
- SHA3 hashes may be used only for internal hardware functionality such as boot integrity checks, and
- SHA-256 is permitted only for use as part of LMS and XMSS, or temporarily in certain non-CNSA cases as a PRF, but these non-CNSA uses are discouraged.

The hash selection should be consistent with the overall strength of the algorithm used for signature generation. For example, the TOE should choose SHA-384 for 3072-bit RSA, 4096-bit RSA, or ECC with P-384; and SHA-512 for ECC with P-521.

Evaluation Activities

FCS_COP.1/Hash
TSS
The evaluator shall examine the TSS to verify that if SHA-256 is selected, that it is being used only as part of LMS and XMSS, or as a PRF, not as a hash algorithm.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

SHA-256, SHA-384, SHA-512

To test the TOE's ability to generate hash digests using SHA2 the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Large Data Test for each claimed SHA2 algorithm.

Algorithm Functional Test

The evaluator shall generate a number of test cases equal to the block size of the hash (512 for SHA2-256; 1024 for the other SHA2 algorithms).

Each test case is to consist of random data of a random length between 0 and 65536 bits, or the largest size supported.

Each test case is to consist of random data of a random length between 0 and 65536 bits, or the largest size supported.

Monte Carlo Test

Monte Carlo tests begin with a single seed and run 100 iterations of the chained computation.

There are two versions of the Monte Carlo test for SHA-2. Either one is acceptable. For the Standard Monte Carlo test the message hashed is always three times the length of the initial seed.

```
For j = 0 to 99
        A = B = C = SEED
        For i = 0 to 999
                MSG = A || B || C
                MD = SHA(MSG)
                A = B
                B = C
                C = MD
        Output MD
        SEED = MD
```

For the alternate version of the Monte Carlo Test, the hashed message is always the same length as the seed.

```
INITIAL_SEED_LENGTH = LEN(SEED)
For j = 0 to 99
        A = B = C = SEED
        For i = 0 to 999
                MSG = A || B || C
                if LEN(MSG) >= INITIAL_SEED_LENGTH:
                        MSG = leftmost INITIAL_SEED_LENGTH bits of MSG
                else:
                        MSG = MSG || INITIAL_SEED_LENGTH - LEN(MSG) 0 bits
                MD = SHA(MSG)
                A = B
                B = C
                C = MD
        Output MD
        SEED = MD
```

The evaluator shall compare the output against results generated by a known-good implementation with the same input.

Large Data Test

The implementation must be tested against one test case each on large data messages of 1GB, 2GB, 4GB, and 8GB of data as supported. The data need not be random. It may, for example, consist of a repeated pattern of 64 bits.

The evaluator shall compare the output against results generated by a known-good implementation with the same input.

SHA3-384, SHA3-512 *To test the TOE's ability to generate hash digests using SHA3 the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Large Data Tests for each claimed SHA3 algorithm.*
Algorithm Functional Test

Generate a test case consisting of random data for every message length from 0 bits (or the smallest supported message size) to rate bits, where rate equals

- *832 for SHA3-384 and*
- *576 for SHA3-512.*

Additionally, generate tests cases of random data for messages of every multiple of (rate+1) bits starting at length rate, and continuing until 65535 is exceeded.

The evaluator shall compare the output against results generated by a known-good implementation with the same input.

Monte Carlo Test

Monte Carlo tests begin with a single seed and run 100 iterations of the chained computation.

For this Monte Carlo Test, the hashed message is always the same length as the seed.

```
MD[0] = SEED
INITIAL_SEED_LENGTH = LEN(SEED)
For 100 iterations
        For i = 1 to 1000
                MSG = MD[i-1];
                if LEN(MSG) >= INITIAL_SEED_LENGTH:
                        MSG = leftmost INITIAL_SEED_LENGTH bits of MSG
                else:
                        MSG = MSG || INITIAL_SEED_LENGTH - LEN(MSG) 0 bits
                MD[i] = SHA3(MSG)
        MD[0] = MD[1000]
        Output MD[0]
```

The evaluator shall compare the output against results generated by a known-good implementation with the same input.

Large Data Test

The implementation must be tested against one test case each on large data messages of 1GB, 2GB, 4GB, and 8GB of data as supported. The data need not be random. It may, for example, consist of a repeated pattern of 64 bits.

The evaluator shall compare the output against results generated by a known-good implementation with the same input.

FCS_COP.1/KeyedHash Cryptographic Operation - Keyed Hash

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.5.1*****,**
- ***FCS_STG_EXT.3.1*****,**
- ***FDP_ITC_EXT.1.2***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_CKM_EXT.8***
- ***FCS_IPSEC_EXT.1***

This component may also be included in the ST as if optional.

FCS_COP.1.1/KeyedHash

The TSF shall perform [*keyed hash message authentication*] in accordance with a specified cryptographic algorithm [**selection**: *Keyed Hash Algorithm*] and cryptographic key sizes [**selection**: *Cryptographic key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 17 provides the allowable choices for completion of the selection operations of FCS_COP.1/KeyedHash.

Table 17: Allowable choices for FCS_COP.1/KeyedHash

Keyed Hash Algorithm	Cryptographic key sizes	List of standards
HMAC-SHA-384	[**selection**: *384 (ISO, FIPS), 256 (FIPS)*] bits	[**selection**: *ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"), FIPS PUB 198-1*]
HMAC-SHA-512	[**selection**: *512 (ISO, FIPS), 384 (FIPS), 256 (FIPS)*] bits	[**selection**: *ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"), FIPS PUB 198-1*]

Application Note: The HMAC minimum key sizes in the table are specified in ISO/IEC 9797-2:2021, which requires that the minimum key size be equal to the digest size. The FIPS standard specifies no minimum or maximum key sizes, so if FIPS PUB 198-1 is selected, larger or smaller key sizes may be used. This is indicated by the parenthesized annotations in the Cryptographic Key Sizes column.

Evaluation Activities

FCS_COP.1/KeyedHash

TSS

The evaluator shall examine the TSS to ensure that the size of the key is sufficient for the desired security strength of the output.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

HMAC

Keyed Hash Algorithm	Cryptographic Key Sizes	List of Standards
HMAC-SHA-384	[**selection:** (ISO, FIPS) 384, (FIPS) 256] bits	[**selection:** ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"), FIPS PUB 198-1]
HMAC-SHA-512	[**selection:** (ISO, FIPS) 512, (FIPS) 384, 256] bits	[**selection:** ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"), FIPS PUB 198-1]

To test the TOE's ability to generate keyed hashes using HMAC the evaluator shall perform the Algorithm Functional Test for each combination of claimed HMAC algorithm the following parameters:

- *Hash function [SHA-384, SHA-512]*
- *Key length [8-65536] bits by 8s*
- *MAC length [32-[digest size of hash function (256, 384, 512)]] bits*

Algorithm Functional Test

For each supported Hash function the evaluator shall generate 150 test cases using random input messages of 128 bits, random supported key lengths, random keys, and random supported MAC lengths such that across the 150 test cases:

- *The key length includes the minimum, the maximum, a key length equal to the block size, and key lengths that are both larger and smaller than the block size.*
- *The MAC size includes the minimum, the maximum, and two other random values.*

The evaluator shall compare the output against results generated by a known-good implementation with the same input.

FCS_COP.1/KeyEncap Cryptographic Operation - Key Encapsulation

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.2.1***

FCS_COP.1.1/KeyEncap

The TSF shall perform [*key encapsulation*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic algorithm*] and cryptographic key sizes [**selection**: *Cryptographic key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 18 provides the allowable choices for completion of the selection operations of FCS_COP.1/KeyEncap.

Table 18: Allowable choices for FCS_COP.1/KeyEncap

Identifier	Cryptographic algorithm	Cryptographic key sizes	List of standards
KAS1	RSASVE	[**selection**: *3072*, *4096*, *6144*, *8192*] bits	NIST SP 800-56B Revision 2 (Section 8.2)
KTS-OAEP	RSA-OAEP	[**selection**: *3072*, *4096*, *6144*, *8192*] bits	NIST SP 800-56B Revision 2 (Sections 6.3 and 9)
ML-KEM	ML-KEM	Parameter set = ML-KEM-1024	NIST FIPS 203

Application Note: This SFR is claimed when "key encapsulation" is selected in FCS_CKM.2.1. NIST SP 800-57 Part 1 Revision 5 Section 5.6.2 specifies that the size of key used to protect the key being transported should be at least the security strength of the key it is protecting.

If this SFR is claimed, then FCS_CKM.1/AKG and FCS_CKM.6 must also be claimed.

KAS1 and KTS-OAEP with the selectable parameters are CNSA compliant. ML-KEM-1024 is CNSA compliant.

Evaluation Activities

FCS_COP.1/KeyEncap

TSS

The evaluator shall ensure that the TSS documents that the selection of the key size is sufficient for the security strength of the key encapsulated.

The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed and used in accordance with the relevant standards.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

KAS1 [RSASVE single-party]

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
KAS1	RSASVE	[**selection:** 3072, 4096, 6144, 8192] bits	NIST SP 800-56B Revision 2 (Section 8.2)

To test the TOE's implementation of the of KAS1 RSASVE Single-Party Key Encapsulation, the evaluator shall perform the Algorithm Functional Test and Validation Test using the following input parameters:

- *RSA Private key format [Basic with fixed public exponent, Prime Factor with fixed public exponent, Chinese Remainder Theorem with fixed public exponent, Basic with random public exponent, Prime Factor with random public exponent, Chinese Remainder Theorem with random public exponent]*
- *Modulo value [3072, 4096, 6144, 8192]*
- *Role [initiator, responder]*
- *Key confirmation supported [yes, no]*

The evaluator shall generate a test group (i.e. set of tests) for each parameter value of the above parameter type with the largest number of supported values. For example, if the TOE supports all six key formats, then the evaluator shall generate six test groups. Each of the above supported parameter values must be included in at least one test group.

Regardless of how many parameter values are supported, there must be at least two test groups.

Half of the test groups are designated as Algorithm Functional Tests (AFT) and the remainder are designated as Validation Tests (VAT). If there is an odd number of groups, then the extra group is designated randomly as either AFT or VAT.

If there are only two test groups, in addition to the above, one shall act as an initiator, and the other as a responder, if supported.

Algorithm Functional Test

For each test group designated as AFT, the evaluator shall generate 10 test cases using random data (except for a fixed public exponent, if supported). The resulting shared secrets shall be compared with those generated by a known-good implementation using the same inputs.

Validation Test

For each test group designated as VAT, the evaluator shall generate 25 test cases are using random data (except for a fixed public exponent, if supported). Of the 25 test cases:

- *Two test cases must have a shared secret with a leading nibble of 0s,*
- *Two test cases have modified derived key material,*
- *Two test cases have modified tags, if key confirmation is supported,*
- *Two test cases have modified MACs, if key confirmation is supported, and*
- *The remaining test cases are not modified.*

To determine correctness, the evaluator shall confirm that the resulting 25 shared secrets correspond as expected for both the modified and unmodified values.

KTS-OAEP [RSA-OAEP]

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
KTS-OAEP	RSA-OAEP	[**selection:** 3072, 4096, 6144, 8192] bits	NIST SP 800-56B Revision 2 (Sections 6.3 & 9)

To test the TOE's implementation of the of KTS-OAEP, the evaluator shall perform the Algorithm Functional Test and Validation Test using the following input parameters:

- *Roles [initiator, receiver]*
- *Private Key format [Basic with fixed public exponent, Prime Factor with fixed public exponent, Chinese Remainder Theorem with fixed public exponent, Basic with random public exponent, Prime Factor with random public exponent, Chinese Remainder Theorem with random public exponent]*
- *Supported modulus values [3072, 4096, 6144, 8192]*
- *Key confirmation supported [yes, no]*

The evaluator shall generate a test group (i.e. set of tests) for each parameter value of the above parameter type with the largest number of supported values. For example, if the TOE supports all six key formats, then the evaluator shall generate six test groups. Each of the above supported parameter values must be included in at least one test group.

Regardless of how many parameter values are supported, there must be at least two test groups.

Half of the test groups are designated as Algorithm Functional Tests (AFT) and the remainder are designated as Validation Tests (VAT). If there is an odd number of groups, then the extra group is designated randomly as either AFT or VAT.

If there are only two test groups, in addition to the above, one shall act as an initiator, and the other as a responder, if supported.

Algorithm Functional Test

For each test group designated as AFT, the evaluator shall generate 10 test cases using random data (except for a fixed public exponent, if supported). The resulting shared secrets shall be compared with those generated by a known-good implementation using the same inputs.

Validation Test

For each test group designated as VAT, the evaluator shall generate 25 test cases are using random data (except for a fixed public exponent, if supported). Of the 25 test cases:

- *Two test cases must have a shared secret with a leading nibble of 0s,*
- *Two test cases have modified derived key material,*
- *Two test cases have modified tags, if key confirmation is supported,*
- *Two test cases have modified MACs, if key confirmation is supported, and*
- *The remaining test cases are not modified.*

To determine correctness, the evaluator shall confirm that the resulting 25 shared secrets correspond as expected for both the modified and unmodified values.

ML-KEM Key Encapsulation

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
ML-KEM	ML-KEM	Parameter set = ML-KEM-1024	NIST FIPS PUB 203

To test the TOE's implementation of ML-KEM key encapsulation/decapsulation, the evaluator shall perform the Encapsulation Test and the Decapsulation Test using the following input parameters:

- *Encapsulation Parameters:*
  - *Parameter set [ML-KEM-1024]*
  - *Previously generated encapsulation key (ek)*
  - *Random value (m) [32 bytes]*
- *Decapsulation Parameters:*
  - *Parameter set [ML-KEM-1024]*
  - *Previously generated decapsulation key (dk)*
  - *Previously generated ciphertext (c) [32 bytes]*

The random value m is normally generated internally by ML-KEM.Encaps(ek). The encapsulation test will actually be testing the TSF's internal encapsulation function ML-KEM.Encaps_Internal(ek, m) as described in FIPS 203.

Encapsulation Test

For each supported parameter set the evaluator shall generate 25 test cases consisting of an encapsulation key ek and random value m. For each test case the valuator shall require the implementation under test to generate the corresponding shared secret k and ciphertext c. To determine correctness, the evaluator shall compare the resulting values with those generated using a known-good implementation using the same inputs.

Encapsulation Key Check (if supported)

The evaluator shall generate 10 encapsulation keys such that:

- *Five of the encapsulation keys are valid, and*
- *Five of the encapsulation keys are modified such that a value in the noisy linear system is encoded into the key as a value greater than Q.*

The evaluator shall invoke the TOE's Encapsulation Key Check functionality to determine the validity of the 10 keys. The unmodified keys should be determined valid, and the modified keys should be determined invalid.

Decapsulation Key Check (if supported)

The evaluator shall generate 10 decapsulation keys such that:

- *Five of the decapsulation keys are valid, and*

- *Five of the decapsulation keys are modified such that the concatenated values ek||H(ek) will no longer match by modifying H(ek) to be a different value.*

The evaluator shall invoke the TOE's Decapsulation Key Check functionality to determine the validity of the 10 keys. The unmodified keys should be determined valid, and the modified keys should be determined invalid.

Decapsulation Test

For each supported parameter set the evaluator shall use a single previously generated decapsulation key dk and generate 10 test cases consisting of valid and invalid ciphertexts c. For each test case the evaluator shall require the implementation under test to generate the corresponding shared secret k whether or not the ciphertext is valid. To determine correctness, the evaluator shall compare the resulting values with those generated using a known-good implementation using the same inputs.

FCS_COP.1/KeyWrap Cryptographic Operation - Key Wrapping

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.2.1,***
- ***FCS_STG_EXT.3.1,***
- ***FDP_ITC_EXT.1.2***

FCS_COP.1.1/KeyWrap

The TSF shall perform [*key wrapping*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic algorithm*] and cryptographic key sizes [**selection**: *Cryptographic key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 19 provides the allowable choices for completion of the selection operations of FCS_COP.1/KeyWrap.

Table 19: Allowable choices for FCS_COP.1/KeyWrap

Identifier	Cryptographic algorithm	Cryptographic key sizes	List of standards
AES-KW	AES in KW mode	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 19772:2020 (clause 6), NIST SP 800-38F (Section 6.2)*] [KW mode]
AES-KWP	AES in KWP mode	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] NIST SP 800-38F (Section 6.3) [KWP mode]
AES-CCM	AES in CCM mode with unpredictable, non-repeating nonce, minimum size of 64 bits	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C*] [CCM]
AES-GCM	AES in GCM mode with non-repeating IVs using [**selection**: *deterministic, RBG-based*], IV construction; the tag	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES]

	must be of length [**selection**: *96, 104, 112, 120, 128*] bits.	[**selection**: *ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D*] [GCM]

Application Note: This SFR is claimed when "key wrapping" is selected in FCS_CKM.2.1 or is used as a mechanism supporting verification of imported key integrity (FDP_ITC_EXT.1.2) or the integrity of stored key data (FCS_STG_EXT.3.1). NIST 800-57p1rev5 sec. 5.6.2 specifies that the size of key used to protect the key being transported should be at least the security strength of the key it is protecting.

Evaluation Activities

FCS_COP.1/KeyWrap

TSS

The evaluator shall ensure that the TSS documents that the selection of the key size is sufficient for the security strength of the key wrapped.

The evaluator shall examine the TSS to ensure that it describes the construction of any IVs, nonces, and MACs in conformance with the relevant specifications.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

For tests of AES-GCM and AES-CCM, see testing for FCS_COP.1/AEAD.

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

AES-KW

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-KW	AES in KW mode	256 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [**selection:** ISO/IEC 19772:2020 (clause 6), NIST SP 800-38F (Section 6.2)] [KW mode]

To test the TOE's ability to wrap keys using AES in Key Wrap mode the evaluator shall perform the Algorithm Functional Tests using the following input parameters:

- *Key size [256] bits*
- *Keyword cipher type [cipher, inverse]*
- *Payload sizes [128-4096] bits by 64s*

Algorithm Functional Test

The evaluator shall generate 100 encryption test cases using random data for each combination of claimed key size, keyword cipher type, and six supported payload sizes such that the payload sizes include the minimum, the maximum, two that are divisible by 128, and two that are not divisible by 128.

The results shall be compared with those generated by a known-good implementation using the same inputs.

The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.

AES-KWP

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards

AES-KWP	AES in KWP mode	256 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] NIST SP 800-38F (Section 6.3) [KWP mode]

To test the TOE's ability to wrap keys using AES in Key Wrap with Padding mode with padding the evaluator shall perform the Algorithm Functional Tests using the following input parameters:

- *Key size [256] bits*
- *Keyword cipher type [cipher, inverse]*
- *Payload sizes [8-4096] bits by 8s*

Algorithm Functional Test

The evaluator shall generate 100 encryption test cases using random data for each combination of claimed key size, keyword cipher type, and six supported payload sizes such that the payload sizes include the minimum, the maximum, two that are divisible by 128, and two that are not divisible by 128.

The results shall be compared with those generated by a known-good implementation using the same inputs.

The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.

FCS_COP.1/SigGen Cryptographic Operation - Signature Generation

The inclusion of this selection-based component depends upon selection in:

- ***FCS_STG_EXT.3.1***

This component may also be included in the ST as if optional.

FCS_COP.1.1/SigGen

The TSF shall perform [*digital signature generation*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic algorithm*] and cryptographic key sizes [**selection**: *Cryptographic key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 20 provides the allowable choices for completion of the selection operations of FCS_COP.1/SigGen.

Table 20: Allowable choices for FCS_COP.1/SigGen

Identifier	Cryptographic algorithm	Cryptographic key sizes	List of standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection**: *3072*, *4096*, *6144*, *8192*] bits, hash [**selection**: *SHA-384*, *SHA-512*]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [**selection**: *3072*, *4096*, *6144*, *8192*] bits, hash [**selection**: *SHA-384*, *SHA-512*], Salt Length (*sLen*) such that [**assignment**: *0 ≤ sLen ≤ hLen (Hash Output Length)*] and Mask Generation Function = MGF1	RFC 8017 (Section 8.1) [PKCS#1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]

ECDSA	ECDSA	Elliptic Curve [**selection**: *P-384*, *P-521*], per-message secret number generation [**selection**: *extra random bits*, *rejection sampling*, *deterministic*] and hash function using[**selection**: *SHA-384*, *SHA-512*]	[**selection**: *ISO/IEC 14888-3:2018 (Subclause 6.6)*, *FIPS PUB 186-5 (Sections 6.3.1, 6.4.1)*] [ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]
LMS	LMS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Winternitz parameter = [**selection**: *1*, *2*, *4*, *8*] Tree height = [**selection**: *5*, *10*, *15*, *20*, *25*]	RFC 8554 [LMS] NIST SP 800-208 [parameters]
ML-DSA	ML-DSA Signature Generation	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.2)
XMSS	XMSS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Tree height = [**selection**: *10*, *16*, *20*]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]

Application Note: This SFR must be included in the ST if digital signature generation is a service provided by the TOE to tenant software, or if digital signature generation is used by the TOE itself to support or implement PP-specified security functionality.

Specifically, this SFR must be included if "*A digital signature of the stored key in accordance with FCS_COP.1/SigGen using an asymmetric key that is protected in accordance with FCS_STG_EXT.2*" is selected in FCS_STG_EXT.3.

If this SFR is included in the ST, then FCS_COP.1/Hash and FCS_RBG.1 must also be claimed.

Evaluation Activities

FCS_COP.1/SigGen

TSS

The evaluator shall examine the TSS and verify that any hash function is the appropriate security strength for the signing algorithm.

The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed and used in accordance with the relevant standards.

The evaluator shall examine the TSS to verify that the TOE has appropriate measures in place to ensure that hash-based signature algorithms do not reuse private keys

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

RSA-PKCS Signature Generation

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection:** 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]

To test the TOE's ability to perform RSA Digital Signature Generation using PKCS1-v1_5 signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

- *Modulus size [3072, 4096, 6144, 8192] bits*
- *Hash algorithm [SHA-384, SHA-512]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate three test cases using random data. The evaluator shall compare the results against those from a known-good implementation.

RSA-PSS Signature Generation

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
RSA-PSS	RSASSA-PSS	Modulus of size [**selection:** 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512], Salt Length (*sLen*) such that [**assignment:**0 ≤ *sLen* ≤ *hLen* (Hash Output Length)] and Mask Generation Function = MGF1	RFC 8017 (Section 8.2) [PKCS #1 v2.2] NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]

To test the TOE's ability to perform RSA Digital Signature Generation using PSS signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

- *Modulus size [3072, 4096, 6144, 8192] bits*
- *Hash algorithm [SHA-384, SHA-512]*
- *Salt length [Fixed based on implementation]*
- *Mask function [MGF1]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate three test cases using random data. The evaluator shall compare the results against those from a known-good implementation.

ECDSA Signature Generation

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
ECDSA	ECDSA	Elliptic Curve [**selection:** P-384, P-521], per-message secret number generation [**selection:** extra random bits, rejection sampling, deterministic] and hash function using [**selection:** SHA-384, SHA-512]	[**selection:** ISO/IEC 14888-3:2018 (Subclause 6.6), NIST FIPS PUB 186-5 (Sections 6.3.1, 6.4.1] [ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]

To test the TOE's ability to perform ECDSA Digital Signature Generation using extra random bits or rejection sampling for secret number generation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Elliptic Curve [P-384, P-521]*
- *Hash algorithm [SHA-384, SHA-512]*

To test the TOE's ability to perform ECDSA Digital Signature Generation using deterministic secret number generation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Elliptic Curve [P-384, P-521]*
- *Hash algorithm [SHA-384, SHA-512]*

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate 10 test cases using random data. The evaluator shall compare the results against those from a known-good implementation.

LMS Signature Generation

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
LMS	LMS	Private key size = [**selection:** 192 bits with [**selection:** SHA256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]] , Winternitz parameter = [**selection:** 1, 2, 4, 8], and tree height = [**selection:** 5, 10, 15, 20, 25]	RFC 8554 [LMS] NIST SP 800-208 [parameters]

To test the TOE's ability to generate cryptographic digital signature using LMS, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]*
- *Winternitz [1, 2, 4, 8]*
- *Tree height [5, 10, 15, 20, 25]*

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall generate 10 signatures. The evaluator shall verify the correctness of the implementation by comparing values generated by the TOE with those generated by a known good implementation using the same input parameters.

ML-DSA Signature Generation

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
ML-DSA	ML-DSA SigGen	Parameter set = ML-DSA-87	NIST FIPS PUB 204 (Section 5.2)

To test the TOE's ability to generate digital signatures using ML-DSA, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Parameter set [ML-DSA-87]*
- *Seed [32 random bytes] (for non-deterministic signature testing), or*
- *Seed [32 zero bytes] (for deterministic signature testing)*
- *Message to sign [8-65535] bytes*
- *Mu value (if generated externally)*
- *Previously generated private key (sk)*
- *Context (for external interface testing)*

The random seed is normally generated internally by ML-DSA.Sign(sk, Message, context). This test will actually be testing the TSF's internal key signature function ML-DSA.Sign_internal(sk, M', seed) as described in FIPS 204, unless the test setup allows capturing the seed.

Algorithm Functional Test

For each combination of supported parameter set and capabilities, the evaluator shall require the implementation under test to generate 15 signatures pairs using 15 different randomly generated 32-byte seed values. To determine correctness, the evaluator shall compare the resulting key pairs with those generated using a known-good implementation using the same inputs.

Known Answer Test for Rejection Cases

For each supported parameter set, the evaluator shall cause the TOE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in

the fourth row for each corresponding test case below.

The test values are defined as follows:

- *Seed is the seed to generate the key pair (pk, sk)*
- *Hash of keys is computed by SHA-256(pk||sk)*
- *Message is the message to be signed*
- *Hash of sig is computed by SHA-256(sig)*

ML-DSA-87 Test Cases for Rejection Cases

```
Test case 87-RC-01
Seed:                   E4F5AFCF697E0EC3C1BDEB66FAA903221E803902F9C3F716E1056A63D77DC250
Hash of Keys:   61618E8DDA6998072C8EB36974E03880D741CAF0BD523356DFC161E7C9E63934
Message:                F4F1C05004D5B946F69EAFE104C4020519086ADDB9582A20FDE887D13DFC36B1
Hash of sig:    B584E38FA442FC3C81A147D4BDBF058D73C822CAF5CA4C06B0110867F60A8001

Test case 87-RC-02
Seed:                   8B828D871254D6C57384A8E7025AA3F7160CAD1D2C754499DF3844426062C3DD
Hash of Keys:   BB64481317D6C0DBAD20C0C7EF11078AD54E5D574F4A07652115A95F77C655FA
Message:                0F9409C5A4930C25B83FC5B77FDB5BB49C75372DE724D9C1A77DB700CF0CF154
Hash of sig:    F86B49BE9DEB2B209BDEB4E922E5939E92D38E562C44BB09AFBD67323C345192

Test case 87-RC-03
Seed:                   E693D282CACB8CE65FD4D108DA7A373F097F0AA9713550BE242AAD5BD3E2E452
Hash of Keys:   B0BEAF56713A69BD4AB2CBEE006FA5001E7B41F3AE541E05F088933AA0CC78DF
Message:                24DABB9D57ADEBD560ED65D9451C5106D437061708F849BA53F3543CDF9AAAE0
Hash of sig:    DBF65CEFF9F96A74AAF6F3AB27B043231BE6AA04FBA2EEC987A24A00BDD6A08E

Test case 87-RC-04
Seed:                   4002163EB8EED01A8E0919BA8C07D291341EDCAE25B02B9779A2CFFE50561AF0
Hash of Keys:   FED1BE685C20ECB322FC40D41DEE7E0E98D0409FBF989CAE71B8AD2D58AD645E
Message:                EE316BB5EBED53325B4A55571C60657B53E353B51B831F4A0BBB28107EBA4BA8
Hash of sig:    3BE9B5545FDCED92547B3409C83B3312CCB5792A8EC3A4DA63BA692C79BEF17C

Test case 87-RC-05
Seed:                   9C7AD524F65854C27E565BCEDF8E86D650F13A40D0448F9AE10C05F10F777120
Hash of Keys:   0EA872CA5A4BEA94F4E8EF7ED31800727899A51059FDEE111E5CB15F0233B534
Message:                CE09831294AA96CAF684B9E667947B021C57B24C138EC7D4DA270694C82F2E08
Hash of sig:    3B9526CEE6587F2418BFE603ADB0F7DF0D69EBA31C9F9F005C60C993945EBD33

Test case 87-RC-06
Seed:                   2EB7676D4A28700DA7772A7A035EB495CAA6F842352A74824EF5FD891BC38B2A
Hash of Keys:   D5B73703A1DDC5BCB0D14AE39B193A25D6ADA6535827973181ADB0BE70435A5B
Message:                C2B3A0AC483A5517682285C205974B2A506946448A8F7D3E1934C155EFDFE922
Hash of sig:    375D598704B722C8A1FEF1626FD7738A532C06329AA4217357460E3B729660F8

Test case 87-RC-07
Seed:                   E4E80CCE8B26DF1B02B99949851EE2F907FE4F0CC34790352C76D5D91634D073
Hash of Keys:   84B7E61684A12698400B09EA332EA3C4FBCFA47FE37FD6AE725CBC5FA8A99D3F
Message:                89E6AB43C9CB1CC59C3986D53217A558357E62102A26F666F2B64CD1DBB7A536
Hash of sig:    7C4AABD163CAEF8F6EBFDA3E3EEBC0A9604675B0E991ABAFD284F1AE8BA07B2A

Test case 87-RC-08
Seed:                   5787262B803499223D4E5A8C1EE572E89F7A69B359B3F8505355B0BDEAB95E5C
Hash of Keys:   85AE1DE605A7B479C02730BF4B7DD6D0FD8FFE5C980893CA6DAD00BD8BD1CE68
Message:                D3230C4E061964BBFB17702432D5D36FC1EB3D1068F8CCAA84044776E3B5CC55
Hash of sig:    D3ABE460EE2DD9595F413CFE2780A319E4E4DFD6592995298A7AB0B82A5E2815

Test case 87-RC-09
Seed:                   CE099B99330537DD153052243FC32ACAD509A126AB982410258858567D410D79
Hash of Keys:   E04A9F15EDF8F078EB336CE624249EF2A8EDF2CDBF6A8276E9F5E92ED9B0BAE8
Message:                0035931762665F561A1B22176567E3B10FDE2441521F77030733A8E39312EEEE
Hash of sig:    3EEF413CB5EB179896ECA172D0DBFB9B251545DC561D61580BD5BBC8B6D734E1

Test case 87-RC-10
Seed:                   FC8F2929878CBD81E1CCC23913F290380120C043A4A8A251AEEBF09705B8E590
Hash of Keys:   7E2ECCA86F532E8E8092FEBB6E0007F92E7909AD2BCBE2E02AB375DAC9969E5E
Message:                D3C28875D2671C0EF23BFDC8869E8ECF8868D3F0561C3134D254F7479D0CE0E5
Hash of sig:    EB69A908EDCC04320A0B61AD57E21B044465F2037698636B64229CF2DB259789
```

Known Answer Test for Large Number of Rejection Cases (Total Rejection Count)

For each supported parameter set, the evaluator shall cause the TOE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in the fourth row of the corresponding test case below.

ML-DSA-87 Test Cases for Total Rejection Count

```
Test case 87-LN-01
Seed:                   98B6298051D92BF37293C93C97370747BF527B87B71F6C4264182F45155ADE4C
Hash of Keys:   04A135B5C9B7020332C7B16E7108E8FF7FC1EAE1C23C5FA0B5D5CED0FEEE7424
Message:                D7B0341269259083ABF3C8DC47559A19D57669B4486E0224F376DC43E577A3D8
Hash of sig:    58D72D76EC0FB65BFB9893C4479366B79DD7B8B7577E4291D13514FCC76C26DD

Test case 87-LN-02
```

```
Seed:                   DFB5BDD90F58571DCA962426C623F13D046BBE814D183886AC90D143EAD725A7
Hash of Keys:   2B6AB8CFCCCC41F759CAF01932E9413F5DC6D949BC827F739866929683FB155E
Message:                21005DB2B583CC826A9684BFFD0EE00AB97E0479FE4A1D266699337540145778
Hash of sig:    C93EA34E00FFFFC3ECEA072D5FB038A83B5539CAF7B831AEDCFA785E50B3CA5E

Test case 87-LN-03
Seed:                   5AD414E0DD0EF2FE685F342871875FDF06F503717A86C3B3466565ADD2096417
Hash of Keys:   BD9C2D52F3FC78DB17E682DA2E78947ECFC0898333838D60C892700B2B0DDA9F
Message:                29139C279816B25F2D6BB52C8247D163544F7BA332C3CF63359B9E23FBC56515
Hash of sig:    DB4BE2DE19FB40437BDB7E9B6578D665DB05B4E88C16907DF4546EBA9BE03AEA

Test case 87-LN-04
Seed:                   484DD2F406A4D15F49A91AD5FC3BDC1D0FF253622EB68F83D6E1C870D0E89E29
Hash of Keys:   A719DC9A77C91C46295555C2353BA0CBEA513DA9A92A5C34D2E949EFF46A12D8
Message:                6AD6E959F0EA60126364FB7C95FA71133F246A9265A11B4965EE78AB0CB5AF0E
Hash of sig:    5050D7A665074EC63D9F3966C1F01A1BFB18F9E83AE0B09F838BC1E2342ED6F4

Test case 87-LN-05
Seed:                   B25C1816F82D59940D5CB829BAC364AAD013C4C16415CE1CF6DCC2F15199B391
Hash of Keys:   ADBB2CD43F222640BD9FF4E61C80E63853E8DC1F759C581B7447C9C166EAA38E
Message:                824E47322895BFFE37B6B4AFC41CF6115C07EEC0C24EB81076C87A1B01AE8617
Hash of sig:    667ADA46073BC69D64DC47BB9A76DD0D78302E7415D87D5E816B05FB95F9E84D

Test case 87-LN-06
Seed:                   B2CE72B3560AF07E06465881F56ADA00262BA708D87B73F39E04E310F3B8A3E9
Hash of Keys:   FD9C4AC53AE803242A62DF933B8E8BAD6CE5207AC4A73683B6D9383B5E70B17A
Message:                A1501CC84C917E0D2D7C27C2AC382220BD8FFFE807DB38E37A9E429EC2781911
Hash of sig:    779553B195E11558EE59EF3942F5F6B446A2144600D1F4F50B300C6C56504760

Test case 87-LN-07
Seed:                   AB01D0E591B7DDCD3C03395AED808FA2763C0A486D44119D621BE0FD0B022B25
Hash of Keys:   93B6ADE34F78A4ADB36B2F6D2C51DB793E659E1243E80488AE1C03B65125D6D7
Message:                8DE8122D89D15FE84A4C34F6B59B2C4B11F33B6A053154D199B634F557FDF5F6
Hash of sig:    0483045999A79B583F403DB96A736F0F0B24E2DFBC4E5CFA9B50E3D910786F07

Test case 87-LN-08
Seed:                   15D60D3693762F82C9AC1DCB0576936651AC81D863842EDB91109C8EE83AE705
Hash of Keys:   2DF544E2E939AA717741C2437288FAEB308DEB8FF37A2652FAE34BAE8B84D779
Message:                F05946A6113905C34163AEF2246FD69016CE24A7BA40F8E7E42EDAC2D0A44605
Hash of sig:    F8383917AF79C8E540D2356AB05F08B465BF32DFEC444B787CE31BF48CC6C3DD

Test case 87-LN-09
Seed:                   21212285BED53B3411705DAF5F3BDDB6F0618EB571B36EE11A74053407A269F5
Hash of Keys:   737061155A9A03F11F9FEBBB940BED4DD54542C4A6212F89A5EB4EC2BE542782
Message:                FFE38246BF3DEFD9CAD15CC17CEA511C067D582E04227B479E32F9197CF91482
Hash of sig:    C4C12C58032052FB2D21F0C6A7388A63154FB85B74287D2859DE6C1C6F7F277B

Test case 87-LN-10
Seed:                   A2744470587C71BA43EC26DC390CE3531978F315993C653E5D3EFD2849D5D9F1
Hash of Keys:   B1BF37BFFB11531B6ADD697870D7DB2E2462D0A97A63F09C1D0038457C6D795A
Message:                9831A830231A160B9847203341A5F30BF3E87A2A482AEEA6886315C92B5C4E4C
Hash of sig:    46C669D2FEB643A38E54FF87B790CC33F44043A1B6B31DB9474D301328CA2A7F
```

XMSS Signature Generation

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
XMSS	XMSS	Private key size = [**selection:** 192 bits with [**selection:** SHA256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]], and tree height = [**selection:** 10, 16, 20]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]

To test the TOE's ability to generate digital signatures using XMSS, the evaluator shall perform the XMSS Key Generation Test using the following input parameters:

- *Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]*
- *Tree height [10, 16, 20]*

XMSS Key Generation Test

For each supported combination of the above parameters, the evaluator shall generate 10 signatures. The evaluator shall verify the correctness of the implementation by comparing values generated by the TOE with those generated by a known-good implementation using the same input parameters.

FCS_COP.1/SigVer Cryptographic Operation - Signature Verification

The inclusion of this selection-based component depends upon selection in:

- ***FDP_ITC_EXT.1.2*,**
- ***FPT_ROT_EXT.2.1*,**
- ***FPT_TUD_EXT.1.1***

This component may also be included in the ST as if optional.

FCS_COP.1.1/SigVer

The TSF shall perform [*digital signature verification*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic algorithm*] and cryptographic key sizes [**selection**: *Cryptographic key sizes*] that meet the following: [**selection**: *List of standards*] .

Table 21 provides the allowable choices for completion of the selection operations of FCS_COP.1/SigVer.

Table 21: Allowable choices for FCS_COP.1/SigVer

Identifier	Cryptographic algorithm	Cryptographic key sizes	List of standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection**: *3072, 4096, 6144, 8192*] bits and hash[**selection**: *SHA-384, SHA-512*]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [**selection**: *3072, 4096, 6144, 8192*] bits and hash[**selection**: *SHA-384, SHA-512*]	RFC 8017 (Section 8.1) [PKCS#1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]
ECDSA	ECDSA	Elliptic Curve [**selection**: *P-384, P-521*] using hash [**selection**: *SHA-384, SHA-512*]	[**selection**: *ISO/IEC 14888-3:2018 (Subclause 6.6), FIPS PUB 186-5 (Section 6.4.2)*][ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]
LMS	LMS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Winternitz parameter = [**selection**: *1, 2, 4, 8*] Tree height = [**selection**: *5, 10, 15, 20, 25*]	RFC 8554 [LMS] NIST SP 800-208 [parameters]
XMSS	XMSS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Tree height = [**selection**: *10, 16, 20*]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]
ML-DSA	ML-DSA Signature Verification	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.3)

Application Note: This SFR must be included in the ST if digital signature verification is a service provided by the TOE to tenant software, or if digital signature verification is used by the TOE itself to support or implement PP-specified security functionality.

Specifically, this SFR must be included if the ST Author chooses "*implement an authenticated platform firmware update mechanism as described in FPT_TUD_EXT.2*" or "*implement a delayed-authentication platform firmware update mechanism as described in FPT_TUD_EXT.3*" in FPT_TUD_EXT.1; or if the ST Author selects "*verification of a digital signature by trusted code/data*" in FPT_ROT_EXT.2.

If this SFR is included in the ST, then FCS_COP.1/Hash must also be claimed.

The ST Author should choose the algorithm implemented to perform verification of digital signatures. For the algorithm chosen, the ST Author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm. In particular, if ECDSA is selected as one of the signature algorithms, the key size specified must match the selection for the curve used in the algorithm.

For elliptic curve-based schemes, the key size refers to the binary logarithm (log2) of the order of the base point.

The TOE may contain a public key which is integrity protected (e.g., in hardware), in which case the FDP_ITC.1 and FDP_ITC.2 dependencies do not apply. In this case, no dependencies may be chosen. For signature verifications, private keys are not necessary, so there are no dependencies required for generating or destroying cryptographic keys.

If LMS or XMSS is claimed, then FCS_COP.1/XOF must also be claimed when SHAKE is utilized.

Evaluation Activities

FCS_COP.1/SigVer

TSS

The evaluator shall examine the TSS to verify that any one-time values such as nonces or masks are constructed and used in accordance with the relevant standards.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

RSA-PKCS Signature Verification

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection:** 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]

To test the TOE's ability to perform RSA Digital Signature Verification using PKCS1-v1_5 signature type, the evaluator shall perform Generated Data Test using the following input parameters:

- *Modulus size [3072, 4096, 6144, 8192] bits*
- *Hash algorithm [SHA-384, SHA-512]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate six test cases using a random message and its signature such that the test cases are modified as follows:

- *One test case is left unmodified*
- *For one test case the Message is modified*
- *For one test case the Signature is modified*

- *For one test case the exponent (e) is modified*
- *For one test case the IR is moved*
- *For one test case the Trailer is moved*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

RSA-PSS Signature Verification

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
RSA-PSS	RSASSA-PSS	Modulus of size [**selection:** 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]

To test the TOE's ability to perform RSA Digital Signature Verification using PSS signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

- *Modulus size [3072, 4096, 6144, 8192] bits*
- *Hash algorithm [SHA-384, SHA-512]*
- *Salt length [0-hash length]*
- *Mask function [MGF1]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate six test cases using random data such that the test cases are modified as follows:

- *One test case is left unmodified*
- *For one test case the Message is modified*
- *For one test case the Signature is modified*
- *For one test case the exponent (e) is modified*
- *For one test case the IR is moved*
- *For one test case the Trailer is moved*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

DSA Signature Verification

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
DSA	DSA	Domain parameters for (L, N) = [(3072, 256)] bits	FIPS PUB 186-4 (Section 4.7) [DSA Signature Verification]

To test the TOE's ability to perform DSA Digital Signature Verification, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *(L, N) = (3072, 256)*
- *Hash algorithm [SHA-384, SHA-512]*

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate 15 test cases consisting of messages and signatures such that the 15 test cases are modified as follows:

- *Three test cases are left unmodified*
- *For three test cases the Message is modified*
- *For three test cases the key is modified*
- *For three test cases the r value is modified*
- *For three test cases the s value is modified*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

ECDSA Signature Verification

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
ECDSA	ECDSA	Elliptic Curve [**selection:** P-384, P-521] and hash function using [**selection:** SHA-384, SHA-512]	[**selection:** ISO/IEC 14888-3:2018 (Subclause 6.6), NIST FIPS PUB 186-5 (Sections 6.3.1, 6.4.1] [ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]

To test the TOE's ability to perform ECDSA Digital Signature Verification, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Elliptic Curve [P-384, P-521]*
- *Hash algorithm [SHA-384, SHA-512]*

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate test cases consisting of messages and signatures such that the 21 test cases are modified as follows:

- *Three test cases are left unmodified*
- *For three test cases the Message is modified*
- *For three test cases the key is modified*
- *For three test cases the r value is modified*
- *For three test cases the s value is modified*
- *For three test cases the value r is zeroed*
- *For three test cases the value s is zeroed*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

LMS Signature Verification

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
LMS	LMS	Private key size = [**selection:** 192 bits with [**selection:** SHA256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]], Winternitz parameter = [**selection:** 1, 2, 4, 8], and tree height = [**selection:** 5, 10, 15, 20, 25]	RFC 8554 [LMS] NIST SP 800-208 [parameters]

To test the TOE's ability to verify cryptographic digital signature using LMS, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]*
- *Winternitz [1, 2, 4, 8]*
- *Tree height [5, 10, 15, 20, 25]*

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall generate 4 test cases consisting of signed messages and keys, such that

- *One test case is unmodified (i.e. correct)*
- *For one test case modify the message, i.e. the message is different*
- *For one test case modify the signature, i.e. signature is different*
- *For one test case modify the signature header so that it is a valid header for a different LMS parameter set.*

The TOE must correctly verify the unmodified test case and fail to verify the modified test cases.

XMSS Signature Verification

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards

XMSS	XMSS	Private key size = [**selection:** 192 bits with [**selection:** SHA256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]], and tree height = [**selection:** 10, 16, 20]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]

To test the TOE's ability to verify digital signatures using XMSS or XMSS MT, the evaluator shall perform the XMSS digital signature verification test using the following input parameters:

- *Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]*
- *Tree height [10, 16, 20]*

XMSS Digital Signature Verification Test

For each supported combination of the above parameters, the evaluator shall generate four test cases consisting of signed messages and keys, such that

- *One test case is unmodified (i.e. correct)*
- *For one test case modify the message, i.e. the message is different*
- *For one test case modify the signature, i.e. signature is different*
- *For one test case modify the signature header so that it is a valid header for a different XMSS parameter set*

The evaluator shall verify the correctness of the implementation by verifying that the TOE correctly verifies the unmodified test case and fails to verify the modified test cases.

ML-DSA Signature Verification

Identifier	Cryptographic Algorithm Parameters	Cryptographic Key Sizes	List of Standards
ML-DSA	ML-DSA SigVer	Parameter set = ML-DSA-87	NIST FIPS PUB 204 (Section 5.2)

To test the TOE's ability to validate digital signatures using ML-DSA, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Parameter set [ML-DSA-87]*
- *Previously generated signed Message [8-65535] bytes*
- *Mu value (if generated externally)*
- *Context (for external interface testing)*
- *Previously generated Public key (pk)*
- *Previously generated Signature*

Algorithm Functional Test

For each combination of supported parameter set and capabilities, the evaluator shall require the implementation under test to validate 15 signatures. Each group of 15 test cases is modified as follows:

- *Three test cases are left unmodified*
- *For three test cases the Signed message is modified*
- *For three test cases the component of the signature that commits the signer to the message is modified*
- *For three test cases the component of the signature that allows the verifier to construct the vector z is modified*
- *For three test cases the component of the signature that allows the verifier to construct the hint array is modified*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

FCS_COP.1/SKC Cryptographic Operation - Symmetric Key Cryptography

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.5.1,***
- ***FTP_ITC_EXT.1.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_IPSEC_EXT.1***
- ***FCS_STG_EXT.2***

This component may also be included in the ST as if optional.

FCS_COP.1.1/SKC

The TSF shall perform [*symmetric-key encryption/decryption*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic Algorithm*] and cryptographic key sizes [**selection**: *Cryptographic Key Sizes*] that meet the following: [**selection**: *List of Standards*] .

Table 22 provides the allowable choices for completion of the selection operations of FCS_COP.1/SKC.

Table 22: Allowable choices for FCS_COP.1/SKC

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CBC	AES in CBC mode with non-repeating and unpredictable IVs	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A*] [CBC]
XTS-AES	AES in XTS mode with unique tweak values that are consecutive non-negative integers starting at an arbitrary non-negative integer	512 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *IEEE Std. 1619-2018, NIST SP 800-38E*] [XTS]
AES-CTR	AES in Counter Mode with a non-repeating initial counter and with no repeated use of counter values across multiple messages with the same secret key	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *: ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A*] [CTR]

Application Note: This SFR must be included in the ST if symmetric-key cryptography is a service provided by the TOE to tenant software, or if the TOE itself uses SKC to support or implement PP-specified security functionality.

Specifically, this SFR must be included if the ST includes FCS_IPSEC_EXT.1 or FCS_STG_EXT.2, or includes any of the following selections:

- "*CTR_DRBG (AES)*" in FCS_RBG.1
- "*AES-**" in FCS_STG_EXT.3

Evaluation Activities

FCS_COP.1/SKC

TSS

The evaluator shall examine the TSS to ensure that it describes the construction of any IVs, tweak values, and counters in conformance with the relevant specifications.

If XTS-AES is claimed then the evaluator shall examine the TSS to verify that the TOE creates full-length keys by methods that ensure that the two key halves are distinct.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug

mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

AES-CBC

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CBC	AES in CBC mode with non-repeating and unpredictable IVs	256 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [**selection:** ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]

To test the TOE's ability to encrypt/decrypt data using AES in CBC mode, the evaluator shall perform Algorithm Functional Tests and Monte Carlo Tests using the following input parameters:

- *Key size [256] bits*
- *Direction [encryption, decryption]*

Algorithm Functional Tests

Algorithm Functional Tests are designed to verify the correct operation of the logical components of the algorithm implementation under normal operation using different block sizes. For AES-CBC, there are two types of AFTs:

Known-Answer Tests

For each combination of direction and claimed key size, the TOE must be tested using the GFSBox, KeySbox, VarTxt, and VarKey test cases listed in Appendixes B through E of The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), NIST, 15 November 2002.

Multi-Block Message Tests

For each combination of direction and claimed key size, the TOE must be tested against 10 test cases consisting of a random IV, random key, and random plaintext/ciphertext. The plaintext/ciphertext starts with a length of 16 bytes and increases by 16 bytes for each test case until reaching 160 bytes.

Monte Carlo Tests

Monte Carlo tests are intended to test the implementation under strenuous conditions. The TOE must process the test cases according to the following algorithm once for each combination of direction and key size:

```
Key[0] = Key
IV[0] = IV
PT[0] = PT
for i = 0 to 99 {
        Output Key[i], IV[i], PT[0]
        for j = 0 to 999 {
                if (j == 0) {
                        CT[j] = AES-CBC-Encrypt(Key[i], IV[i], PT[j])
                        PT[j+1] = IV[i]
                } else {
                        CT[j] = AES-CBC-Encrypt(Key[i], PT[j])
                        PT[j+1] = CT[j-1]
                }
        }
        Output CT[j]
        AES_KEY_SHUFFLE(Key, CT)
        IV[i+1] = CT[j]
        PT[0] = CT[j-1]
}
```

where

`AES_KEY_SHUFFLE`

is defined as:

```
If ( keylen = 128 )
        Key[i+1] = Key[i] xor MSB(CT[j], 128)
If ( keylen = 192 )
        Key[i+1] = Key[i] xor (LSB(CT[j-1], 64) || MSB(CT[j], 128))
If ( keylen = 256 )
        Key[i+1] = Key[i] xor (MSB(CT[j-1], 128) || MSB(CT[j], 128))
```

The above pseudocode is for encryption. For decryption, swap all instances of CT and PT.

The initial IV, key, and plaintext/ciphertext should be random.

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

XTS-AES

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
XTS-AES	AES in XTS mode with unique tweak values that are consecutive non-negative integers starting at an arbitrary non-negative integer	512 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [**selection:** IEEE Std. 1619-2018, NIST SP 800-38E] [XTS]

To test the TOE's ability to encrypt/decrypt data using AES in XTS mode, the evaluator shall perform the Single Data Unit Test and the Multiple Data Unit Test using the following input parameters:

- *Direction [encryption, decryption]*
- *Key size [512] bits*
- *Tweak value format [128-bit hex string, data unit sequence number]*

Single Data Unit Test

For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. The payload size and data unit size must be equal.

Multiple Data Unit Test

For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. Likewise, the data unit size is determined randomly for each test case from supported values within the range [128-65535] bits. The payload size and data unit size must not be equal.

The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

AES-CTR

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CTR	AES in Counter Mode with a non-repeating initial counter and with no repeated use of counter values across multiple messages with the same secret key.	256 bits	[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [**selection:** ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A] [CTR]

To test the TOE's ability to encrypt/decrypt data using AES in CTR mode, the evaluator shall perform the Algorithm Functional Test and the Counter Test using the following input parameters:

- *Direction [encryption, decryption]*
- *Key size [256] bits*

Algorithm Functional Tests

Algorithm Functional Tests are designed to verify the correct operation of the logical components of the algorithm implementation under normal operation using different block sizes. For AES-CTR, there are three types of AFTs:

Known-Answer Tests

For each combination of direction and claimed key size, the TOE must be tested using the GFSBox, KeySbox, VarTxt, and VarKey test cases listed in Appendixes B through E of The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), NIST, 15 November 2002.

Single Block Message Tests

For each combination of direction and claimed key, the evaluator shall generate 10 test cases with a data size of 128 bits.

Partial Block Message Tests

Monte Carlo tests are intended to test the implementation under strenuous conditions. The TOE must process the test cases according to the following algorithm once for each combination of direction and key size:

For each combination of direction and claimed key, the evaluator shall generate five test cases such that the data size is not a multiple of 128 bits.

The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.

Counter Test

The evaluator shall generate a single message of 1000 blocks (128000 bits) and either encrypt or decrypt it. Back-compute the IVs used. Verify that they are unique and increasing (encryption) or decreasing (decryption).

FCS_COP.1/XOF Cryptographic Operation - Extendable-Output Function

The inclusion of this selection-based component depends upon selection in:

- ***FCS_COP.1.1/SigVer***

FCS_COP.1.1/XOF

The TSF shall perform [*extendable-output function*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic Algorithm*] and parameters [**selection**: *Parameters*] that meet the following: [**selection**: *List of Standards*] .

Table 23 provides the allowable choices for completion of the selection operations of FCS_COP.1/XOF.

Table 23: Allowable choices for FCS_COP.1/XOF

Cryptographic Algorithm	Parameters	List of Standards
SHAKE	Functions = [SHAKE256]	NIST FIPS PUB 202 Section 6.2 [SHAKE]

Application Note: In accordance with CNSA, SHAKE is permitted to be used only as a component of LMS or XMSS. Therefore this component is claimed only if LMS or XMSS is claimed in FCS_COP.1/SigVer.

Evaluation Activities

FCS_COP.1/XOF

TSS

There are no additional TSS evaluation activities for this component.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The following tests are conditional based on the selections made in the SFR. The evaluator shall perform the following tests or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

SHAKE

Cryptographic Algorithm	Parameters	List of Standards

SHAKE	Function = [SHAKE256]	NIST FIPS PUB 202 Section 6.2 [SHAKE]

To test the TOE's implementation of the SHAKE Extendable Output Function the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Variable Output Test using the following input parameters:

- *Function [SHAKE256]*
- *Output length [16-65536] bits*

Algorithm Functional Test

For each supported function, generate test cases consisting of random data for every message length from 0 bits (if supported) to rate-1 bits, where rate equals

1088.

Additionally, generate tests cases of random data for messages of every multiple of (rate+1) bits starting at length rate, and continuing until 65535 is exceeded.

Monte Carlo Test

The Monte Carlo test takes in a single 128-bit message (SEED) and desired output length in bits, and runs 100 iterations of the chained computation. MaxOutBytes and MinOutBytes are the largest and smallest supported input and output sizes in bytes, respectively.

```
Range = maxOutBytes - minOutBytes + 1
OutputLen = maxOutBytes
For j = 0 to 99
MD[0] = SEED
For i = 1 to 1000
MSG[i] = 128 leftmost bits of MD[i-1]
if (MSG[i] < 128 bits)
Append 0 bits on rightmost side of MSG[i] til MSG[i] is 128
MD[i] = SHAKE(MSG[i], OutputLen * 8)

RightmostOutputBits = 16 rightmost bits of MD[i] as an inte
OutputLen = minOutBytes + (RightmostOutputBits % Range)
Output MD[1000], OutputLen
SEED = MD[1000]
```

Variable Output Test

This test measures the ability of the TOE to generate output digests of varying sizes.

The evaluator shall generate 512 test cases such that the input for each test case consists of 128- bits of random data, and the output length includes the minimum supported value, the maximum supported value, and 510 random values between the minimum and maximum digest sizes supported by the implementation.

FCS_HTTPS_EXT.1 HTTPS Protocol

The inclusion of this selection-based component depends upon selection in:

- ***FTP_ITC_EXT.1.1***

FCS_HTTPS_EXT.1.1

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

Application Note: This SFR is included in the ST if the ST Author selects "*TLS/HTTPS*" in FTP_ITC_EXT.1.1.

If this SFR is included in the ST, then the Functional Package for Transport Layer Security (TLS), version 2.1 must also be claimed.

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS.

Evaluation Activities

FCS_HTTPS_EXT.1
TSS

The evaluator shall check the TSS to ensure that it is clear on how HTTPS uses TLS to establish an administrative session, focusing on any client authentication required by the TLS protocol vs. security administrator authentication which may be done at a different level of the processing stack.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
Testing for this activity is done as part of the TLS testing; this may result in additional testing if the TLS tests are done at the TLS protocol level.

FCS_IPSEC_EXT.1 IPsec Protocol

The inclusion of this selection-based component depends upon selection in:

- ***FTP_ITC_EXT.1.1***

FCS_IPSEC_EXT.1.1

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

Application Note: This SFR must be included in the ST if the ST Author selects "*IPsec*" in FTP_ITC_EXT.1.1.

If this SFR is claimed, then FCS_COP.1/KeyedHash and FCS_RBG.1 must also be claimed.

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

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

FCS_IPSEC_EXT.1.2

The TSF shall implement [**selection**: *transport mode*, *tunnel mode*].

Application Note: If the TOE is used to connect to a VPN gateway for the purposes of establishing a secure connection to a private network, the ST Author should select tunnel mode. If the TOE uses IPsec to establish an end-to-end connection to another IPsec VPN Client, the ST Author should select transport mode. If the TOE uses IPsec to establish a connection to a specific endpoint device for the purpose of secure remote administration, the ST Author should select transport mode.

FCS_IPSEC_EXT.1.3

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

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [*AES-GCM-256 (as specified in RFC 4106)*] together with a Secure Hash Algorithm (SHA)-based HMAC.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol:

[*IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8784, RFC 8247, and RFC 4868 for hash functions*].

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the [*IKEv2*] protocol uses the cryptographic algorithms [*AES-GCM-256 as specified in RFC 5282*] and no other algorithm.

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [*IKEv2 SA lifetimes can be configured by [**selection**: an Administrator, a VPN Gateway] based on [**selection**: number of packets/number of bytes, length of time]*].

Application Note: The ST Author is afforded a selection that allows the ST Author to specify which entity is responsible for "configuring" the life of the SA. An implementation that allows an

administrator to configure the client or a VPN gateway that pushes the SA lifetime down to the client are both acceptable.

As far as SA lifetimes are concerned, the TOE can limit the lifetime based on the number of bytes transmitted, or the number of packets transmitted. Either packet-based or volume-based SA lifetimes are acceptable; the ST Author makes the appropriate selection to indicate which type of lifetime limits are supported.

For IKEv2, there are no hard-coded limits, therefore it is required that an administrator be able to configure the values. In general, instructions for setting the parameters of the implementation, including lifetime of the SAs, should be included in the operational guidance generated for AGD_OPE. It is appropriate to refine the requirement in terms of number of MB/KB instead of number of packets, as long as the TOE is capable of setting a limit on the amount of traffic that is protected by the same key (the total volume of all IPsec traffic protected by that key).

FCS_IPSEC_EXT.1.8

The TSF shall ensure that all IKE protocols implement DH groups [*20 (384-bit Random ECP), and [**selection**: 21 (521-bit Random ECP), 15 (3072-bit MODP), 16 (4196-bit MODP), 17 (6144-bit MODP), 18 (8192-bit MODP), no other DH groups]*].

Application Note: The selection is used to specify additional DH groups supported. This applies to IKEv2 exchanges. It should be noted that if any additional DH groups are specified, they must comply with the requirements (in terms of the ephemeral keys that are established) listed in FCS_CKM.1.

Since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the SAs, the assignments in FCS_IPSEC_EXT.1.9 and FCS_IPSEC_EXT.1.10 may contain multiple values. For each DH group supported, the ST Author consults Table 2 in 800-57 to determine the "bits of security" associated with the DH group. Each unique value is then used to fill in the assignment (for 1.9 they are doubled; for 1.10 they are inserted directly into the assignment). For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192. For FCS_IPSEC_EXT.1.9, then, the assignment would read "[224, 384]" and for FCS_IPSEC_EXT.1.10 it would read "[112, 192]" (although in this case the requirement should probably be refined so that it makes sense mathematically).

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in gx mod p) using the random bit generator specified in FCS_RBG.1, and having a length of at least [**assignment**: *(one or more) number(s) of bits that is at least twice the "bits of security" value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*] bits.

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**assignment**: *(one or more) "bits of security" value(s) associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*].

FCS_IPSEC_EXT.1.11

The TSF shall ensure that all IKE protocols perform peer authentication using a [**selection**: *RSA, ECDSA*] that use X.509v3 certificates that conform to RFC 4945 and [**selection, choose one of**: *Pre-shared Keys, no other method*].

Application Note: At least one public-key-based Peer Authentication method is required in order to conform to this PP. One or more of the public key schemes is chosen by the ST Author to reflect what is implemented. The ST Author also ensures that appropriate FCS requirements reflecting the algorithms used (and key generation capabilities, if provided) are listed to support those methods. Applicable claims from the Functional Package for X.509, version 1.0 are made to support X.509 validation functionality, most notably FIA_XCU_EXT.1 (mandatory requirement defining the TOE's use of certificates), FIA_X509_EXT.1 (X.509 certificate validation) and FIA_X509_EXT.2 (X.509 certificate authentication).

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the [**selection**: *IP address, Fully Qualified Domain Name (FQDN), user FQDN, Distinguished Name (DN)*] and [**selection, choose one of**: *no other reference identifier type, [**assignment**: other supported reference identifier types]*] contained in a certificate does not match the expected value(s) for the entity attempting to establish a connection.

Application Note: The TOE must support at least one of the following identifier types: IP address, Fully Qualified Domain Name (FQDN), user FQDN, or Distinguished Name (DN). In the future, the TOE will be required to support all of these identifier types. The TOE is expected to support as many

IP address formats (IPv4 and IPv6) as IP versions supported by the TOE in general. The ST Author may assign additional supported identifier types in the second selection.

FCS_IPSEC_EXT.1.13

The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

Application Note: At this time, only the comparison between the presented identifier in the peer's certificate and the peer's reference identifier is mandated by the testing below. However, in the future, this requirement will address two aspects of the peer certificate validation: 1) comparison of the peer's ID payload to the peer's certificate which are both presented identifiers, as required by RFC 4945 and 2) verification that the peer identified by the ID payload and the certificate is the peer expected by the TOE (per the reference identifier). At that time, the TOE will be required to demonstrate both aspects (i.e. that the TOE enforces that the peer's ID payload matches the peer's certificate which both match configured peer reference identifiers).

Excluding the DN identifier type (which is necessarily the Subject DN in the peer certificate), the TOE may support the identifier in either the Common Name or Subject Alternative Name (SAN) or both. If both are supported, the preferred logic is to compare the reference identifier to a presented SAN, and only if the peer's certificate does not contain a SAN, to fall back to a comparison against the Common Name. In the future, the TOE will be required to compare the reference identifier to the presented identifier in the SAN only, ignoring the Common Name.

Evaluation Activities

FCS_IPSEC_EXT.1

TSS

In addition to the TSS EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the TOE boundary includes a general-purpose operating system or mobile device, the evaluator shall examine the TSS to ensure that it describes whether the VPN client capability is architecturally integrated with the platform itself or whether it is a separate executable that is bundled with the platform.

Guidance

In addition to the AGD EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the configuration of the IPsec behavior is from an environmental source, most notably a VPN gateway (e.g., through receipt of required connection parameters from a VPN gateway), the evaluator shall ensure that the AGD contains any appropriate information for ensuring that this configuration can be properly applied.

Note in this case that the implementation of the IPsec protocol must be enforced entirely within the TOE boundary; i.e. it is not permissible for a software application TOE to be a graphical front-end for IPsec functionality implemented totally or in part by the underlying OS platform. The behavior referenced here is for the possibility that the configuration of the IPsec connection is initiated from outside the TOE, which is permissible so long as the TSF is solely responsible for enforcing the configured behavior. However, it is allowable for the TSF to rely on low-level platform-provided networking functions to implement the SPD from the client (e.g., enforcement of packet routing decisions).

Tests

As a prerequisite for performing the Test EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall do the following:

The evaluator shall minimally create a test environment equivalent to the test environment illustrated below. The traffic generator used to construct network packets should provide the evaluator with the ability manipulate fields in the ICMP, IPv4, IPv6, UDP, and TCP packet headers. The evaluator shall provide justification for any differences in the test environment.

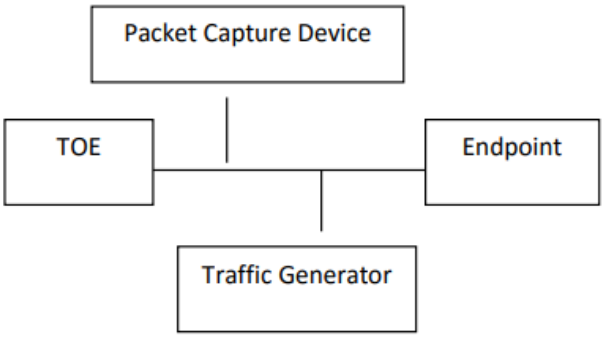


Figure 2: IPsec Test Environment

FCS_IPSEC_EXT.1.1

TSS

The evaluator shall examine the TSS and determine that it describes how the IPsec capabilities are implemented.

The evaluator shall ensure that the TSS describes at a high level the architectural relationship between the IPsec implementation and the rest of the TOE.

The evaluator shall ensure that the TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how the available rules and actions form the SPD using terms defined in RFC 4301 such as BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Guidance

The evaluator shall examine the AGD to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases – a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the AGD is consistent with the description in the TSS, and that the level of detail in the AGD is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

Tests

The evaluator uses the operational guidance to configure the TOE to carry out the following tests:

- *Test FCS_IPSEC_EXT.1.1:1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g., a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.*
- *Test FCS_IPSEC_EXT.1.1:2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios shall exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and operational guidance. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.*

FCS_IPSEC_EXT.1.2

TSS

The evaluator checks the TSS to ensure it states that an IPsec VPN can be established to operate in tunnel mode or transport mode (as selected).

Guidance

The evaluator shall confirm that the AGD contains instructions on how to configure the connection in each mode selected.

If both transport mode and tunnel mode are implemented, the evaluator shall review the AGD to determine how the use of a given mode is specified.

Tests
The evaluator shall perform the following test(s) based on the selections chosen:

- *Test FCS_IPSEC_EXT.1.2:1: [conditional] If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE/platform to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE/platform and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE/Platform to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.*
- *Test FCS_IPSEC_EXT.1.2:2: [conditional] If transport mode is selected, the evaluator uses the operational guidance to configure the TOE/platform to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE/platform and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE/platform to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.*

FCS_IPSEC_EXT.1.3
TSS
There are no additional TSS evaluation activities for this element.

Guidance
The evaluator shall check that the AGD provides instructions on how to construct or acquire the SPD and uses the AGD to configure the TOE for the following test.
If both transport mode and tunnel mode are implemented, the evaluator shall review the AGD to determine how the use of a given mode is specified.

Tests
The evaluator shall perform the following test:

The evaluator shall configure the SPD such that it has entries that contain operations that DISCARD, PROTECT, and (if applicable) BYPASS network packets. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches a BYPASS entry and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE-created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was not permitted to flow to any of the TOE's interfaces.

FCS_IPSEC_EXT.1.4
TSS
The evaluator shall examine the TSS to verify that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1/KeyedHash Cryptographic Operations (Keyed Hash Algorithms).

Guidance
The evaluator checks the Guidance to ensure it provides instructions on how the TOE is configured to use the algorithm selected in this component and whether this is performed through the TOE's default configuration (i.e., no configuration is necessary), direct configuration, configuration defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Tests
The evaluator shall perform the following test:

The evaluator shall configure the TOE/platform as indicated in the operational guidance configuring the TOE/platform to use the supported algorithm, attempt to establish a connection using ESP, and verify that the attempt succeeds.

FCS_IPSEC_EXT.1.5
TSS
The evaluator shall examine the TSS to verify that IKEv2 is implemented.

Guidance
The evaluator shall check the Guidance to ensure it instructs the administrator how to configure the TOE to support only IKEv2 (if necessary), and uses the guidance to configure the TOE to perform NAT traversal for the test below.

Tests
Tests are performed in conjunction with the other IPsec evaluation activities with the exception of the activities below:

- *Test FCS_IPSEC_EXT.1.5:1: The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 7296, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.*
- *Test FCS_IPSEC_EXT.1.5:2: The evaluator shall configure a remote peer to support IKEv1 only. If the TOE's supported versions of IKE is configurable, the evaluator shall follow the instructions specified in the operational*

guidance to ensure that only IKEv2 is supported. The evaluator shall then attempt to establish a connection between the TOE and that peer and verify the TSF rejects the connection attempt based on its lack of support for IKEv1.

FCS_IPSEC_EXT.1.6

TSS

The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv2 payload, and that the algorithm AES-GCM-256 is specified.

Guidance

The evaluator checks the Guidance to ensure it provides instructions on how the TOE is configured to use the algorithm selected in this component and whether this is performed through the TOE's default configuration (i.e., no configuration is necessary), direct configuration, configuration defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Tests

The evaluator shall use the operational guidance to configure the TOE (or to configure the Operational Environment to have the TOE receive configuration) to perform the following test for each ciphersuite selected:

The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation. The evaluator will confirm that the connection is successful by confirming that data can be passed through the connection once it is established. For example, the evaluator may connect to a webpage on the remote network and verify that it can be reached.

FCS_IPSEC_EXT.1.7

TSS

There are no additional TSS evaluation activities for this element.

Guidance

The evaluator shall check the Guidance to ensure it provides instructions on how the TOE configures the values for SA lifetimes. In addition, the evaluator shall check that the Guidance has the option for either the Administrator or VPN Gateway to configure Phase 1 SAs if time-based limits are supported. Currently there are no values mandated for the number of packets or number of bytes, the evaluator shall simply check the AGD to ensure that this can be configured if selected in the requirement.

Tests

When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

Each of the following tests shall be performed:

- *Test FCS_IPSEC_EXT.1.7:1: [conditional] The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.*
- *Test FCS_IPSEC_EXT.1.7:2: [conditional] The evaluator shall construct a test where an IKEv2 IKE_SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.*
- *Test FCS_IPSEC_EXT.1.7:3: [conditional] The evaluator shall perform a test similar to Test 2 for Child SAs, except that the lifetime will be 8 hours or less instead of 24 hours or less.*

FCS_IPSEC_EXT.1.8

TSS

The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Guidance

There are no additional Guidance evaluation activities for this element.

Tests

The evaluator shall perform the following test:

For each supported DH group, the evaluator shall test to ensure that IKEv2 can be successfully completed using that particular DH group.

FCS_IPSEC_EXT.1.9

TSS

The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x" (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this EP is used, and that the length of "x" and the nonces meet the stipulations in the requirement.

Guidance
There are no additional Guidance evaluation activities for this element.

Tests
There are no test activities for this element.

FCS_IPSEC_EXT.1.10
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.9.

FCS_IPSEC_EXT.1.11
TSS
The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication.

If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished depending on whether the TSF can generate a pre-shared key, accept a pre-shared key, or both.

The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN) and, if multiple fields are supported, the logical order comparison. If the ST Author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate.

Guidance
The evaluator shall check that the AGD describes how pre-shared keys are to be generated and established.

The evaluator ensures the AGD describes how to set up the TOE to use the cryptographic algorithms RSA and/or ECDSA.

In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the AGD also describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE as a trusted CA.

The evaluator shall also ensure that the AGD includes the configuration of the reference identifier(s) for the peer.

Tests
For efficiency's sake, the testing that is performed here has been combined with the testing for X.509 certificate validation defined by Functional Package for X.509, version 1.0, FCS_IPSEC_EXT.1.12, and FCS_IPSEC_EXT.1.13. The following tests shall be repeated for each peer authentication protocol selected in the FCS_IPSEC_EXT.1.11 selection above:

- *Test FCS_IPSEC_EXT.1.11:1: The evaluator shall have the TOE generate a public-private key pair, and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE and the peer VPN used to establish a connection) for its signature. The values for the DN (Common Name, Organization, Organizational Unit, and Country) will also be passed in the request. Alternatively, the evaluator may import to the TOE a previously generated private key and corresponding certificate.*
- *Test FCS_IPSEC_EXT.1.11:2: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.*
- *Test FCS_IPSEC_EXT.1.11:3: The evaluator shall test that the TOE can properly handle revoked certificates – conditional on whether CRL or OCSP is selected; if both are selected, then a test is performed for each method. For this current version of the PP-Module, the evaluator has to only test one up in the trust chain (future drafts may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the SA is established. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the TOE will not establish an SA.*
- *Test FCS_IPSEC_EXT.1.11:4: [conditional] The evaluator shall generate a pre-shared key and use it, as indicated in the operational guidance, to establish an IPsec connection with the VPN GW peer. If the generation of the pre-shared key is supported, the evaluator shall ensure that establishment of the key is carried out for an instance of the TOE generating the key as well as an instance of the TOE merely taking in and using the key.*

  For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:

- *Test FCS_IPSEC_EXT.1.11:5: For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKEv2 authentication succeeds.*
- *Test FCS_IPSEC_EXT.1.11:6: For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKEv2 authentication fails.*

The following tests are conditional:

- *Test FCS_IPSEC_EXT.1.11:7: [conditional] If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented certificate but to match the Common Name in the peer's presented certificate, and verify that the IKEv2 authentication fails.*
- *Test FCS_IPSEC_EXT.1.11:8: [conditional] If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKEv2 authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKEv2 authentication fails. To demonstrate a comparison of DN values, the evaluator shall change any one of the four DN values and verify that the IKEv2 authentication fails.*
- *Test FCS_IPSEC_EXT.1.11:9: [conditional] If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator must repeat test 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKE authentication fails.*
- *Test FCS_IPSEC_EXT.1.11:10: [conditional] If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKE peer.*

FCS_IPSEC_EXT.1.12
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.11.

FCS_IPSEC_EXT.1.13
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.11.

FCS_RBG.1 Random Bit Generation (RBG)

The inclusion of this selection-based component depends upon selection in:

- ***FTP_ITC_EXT.1.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_CKM.1/AKG***
- ***FCS_CKM.1/SKG***
- ***FCS_CKM_EXT.8***
- ***FCS_COP.1/SigGen***
- ***FCS_IPSEC_EXT.1***

This component may also be included in the ST as if optional.

FCS_RBG.1.1

The TSF shall perform deterministic random bit generation services using [**selection**: *DRBG Algorithm*] in accordance with [**selection**: *List of standards*] after initialization with a seed.

Table 24 provides the allowable choices for completion of the selection operations of FCS_RBG.1.

Table 24: Allowable choices for FCS_RBG.1.1

Identifier	DRBG Algorithm	List of standards
HASH_DRBG	Hash_DRBG with [**selection**: *SHA-384, SHA-512*]	[**selection**: *ISO/IEC 18031: 2011 (Section C.2.2), NIST SP 800-90A Revision 1 Section 10.1.1*]
HMAC_DRBG	HMAC_DRBG with [**selection**: *SHA-384, SHA-512*]	[**selection**: *ISO/IEC 18031: 2011 (Section C.2.3), NIST SP 800-90A Revision 1 Section 10.1.2*]
CTR_DRBG	CTR_DRBG with AES-CTR-256	[**selection**: *ISO/IEC 18031: 2011 (Section C.3.2), NIST SP800-90A Revision 1 Section 10.2.1*]

Application Note: CNSA requires the use of 256-bit AES and SHA-384 or SHA-512. SHA-256 and all SHA3 hashes are not allowed.

NIST SP 800-90A contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used and include the specific underlying cryptographic primitives used in the requirement or in the TSS. While any of the identified hash functions (SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.

This SFR must be included in the ST if random bits are provided by the TOE to tenant software, or if it is used by the TOE itself to support or implement PP-specified security functionality.

This SFR is also needed if the following SFRs are included in the ST: FCS_IPSEC_EXT.1, FCS_CKM.1/AKG, FCS_CKM.1/SKG, and FCS_COP.1/SigGen.

Also, this SFR must be claimed if "*KDF-CTR*," "*KDF-FB*," or "*KDF-DPI*" is selected in FCS_CKM.5.

If "HMAC_DRBG" is selected, then FCS_COP.1/KeyedHash must be claimed.

If "Hash_DRBG" is selected, then FCS_COP.1/Hash must be claimed.

If "CTR_DRBG" is selected, then FCS_COP.1/SKC must be claimed.

FCS_RBG.1.2

The TSF shall use a [**selection**: *TSF entropy source [**assignment**: name of entropy source], TSF interface for seeding*] for initialization and reseeding.

Application Note: For the selection in this requirement, the ST author selects "TSF entropy source" if a single entropy source is used as input to the DRBG. The ST author selects "multiple TSF noise sources" if a seed is formed from a combination of two or more entropy sources within the TOE boundary. If the TSF implements two or more separate DRBGs that are seeded in separate manners, this SFR should be iterated for each DRBG. It multiple distinct entropy sources exist such that each DRBG only uses one of them, then each iteration would select "TSF entropy source"; "multiple TSF entropy sources" is only selected if a single DRBG uses multiple entropy sources for its seed. The ST author selects "TSF interface for seeding" if entropy source data is generated outside the TOE boundary.

If "TSF entropy source" is selected, FCS_RBG.3 must be claimed.

If "multiple TSF entropy sources" is selected, FCS_RBG.4 and FCS_RBG.5 must be claimed.

If "TSF interface for seeding" is selected, FCS_RBG.2 must be claimed.

The security strength of the entropy used for seeding depends on the functions for which the TSF uses entropy. The security strength for the various functions is defined in Tables 2 and 3 of NIST SP 800-57A.

FCS_RBG.1.3

The TSF shall update the DRBG state by [**selection**: *reseeding, uninstantiating and re-instantiating*] using a [**selection**: *TSF entropy source [**assignment**: name of entropy source], TSF interface for obtaining entropy [**assignment**: name of the interface]*] in the following situations: [**selection**:

- *never*
- *on demand*
- *on the condition: [**assignment**: condition]*
- *after [**assignment**: time]*

] in accordance with [**assignment**: *list of standards*].

Application Note: If a reseeding is selected in the first selection and something other than "never" is selected in the third selection of FCS_RBG.1.3, but reseeding is not feasible, the TSF will uninstantiate RBGs, rather than produce output that is of insufficient quality. The listed standards should specify the reseed interval and procedure for uninstantiating and reseeding. The remaining selection allows the PP Author to require application-specific conditions for reseeding.

"Uninstantiate" means that the internal state of the DRBG is no longer available for use. In the second selection of FCS_RBG.1.3, "on demand" means that a TOE presents an interface to reseed as a TSFI (e.g., an API call). The interface causes the DRBG to reseed at the request of an authorized user, either with an internal source, an external source, or from input provided through the TSFI (e.g., the API call).

Evaluation Activities

FCS_RBG.1

TSS

If the ST specifies more than one DRBG, the evaluator shall examine the TSS to verify that it identifies the usage of each DRBG mechanism.

The evaluator shall examine the TSS to verify that it specifies the DRBG type, identifies entropy source(s) initializing and reseeding the DRBG, and the situations under which this may occur.

Guidance

The evaluator shall verify that the Guidance instructs the administrator how to configure the TOE to use the selected DRBG mechanism(s), if necessary, and provides information regarding how to instantiate/call the DRBG for RBG services.

If the ST claims that the DRBG state can be updated on demand, the evaluator shall verify that the guidance includes instructions for how to perform this operation.

Tests

The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

HASH_DRBG, HMAC_DRBG

Identifier	RBG Algorithm	List of Standards
HASH_DRBG	Hash_DRBG with [**selection:** SHA-384, SHA-512]	[**selection:** ISO/IEC 18031:2011 (Section C.2.2), NIST SP 800-90A Revision 1 (Section 10.1.1)]
HMAC_DRBG	HMAC_DRBG with [**selection:** SHA-384, SHA-512]	[**selection:** ISO/IEC 18031:2011 (Section C.2.3), NIST SP 800-90A Revision 1 (Section 10.1.2)]

To test the TOE's ability to generate random bits using Hash_DRBG or HMAC_DRBG, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Function [SHA-384, SHA-512]*
- *Max security strength [256] bits*
- *Entropy length [256-65536] bits*
- *Max Personalization String size [65536] bits*
- *Max additional string [65536] bits*
- *Nonce length [128-65536] bits*
- *Min returned bits length [256, 384, 512]*
- *Prediction resistance [on, off]*
- *Reseed [on, off]*

Algorithm Functional Test

The evaluator shall generate 16 test groups for each claimed function. Eight with prediction resistance enabled, and eight with prediction resistance turned off. For each of the eight test groups, four must have reseed enabled and four must have reseed turned off.

Each test group within the core sets of four test groups shall consist of 15 test cases each, such that minimum lengths and maximum lengths for the above parameters are used at least once across the four groups, and the rest are random lengths.

Correctness is determined by comparing the results from a known-good implementation using the same input parameters.

CTR_DRBG

Identifier	RBG Algorithm	List of Standards
CTR_DRBG	CTR_DRBG with AES-CTR-256	[**selection:** ISO/IEC 18031:2011 (Section C.3.2), NIST SP 800-90A Revision 1 (Section 10.2.1)]

To test the TOE's ability to generate random bits using CTR_DRBG, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Mode [AES-CTR-256]*
- *Max security strength [256] bits*
- *Entropy length [256-65536] bits*
- *Max Personalization String size [65536] bits*
- *Max additional string [65536] bits*

- *Nonce length [0-65536] bits*
- *Min returned bits length [128]*
- *Prediction resistance [on, off]*
- *Reseed [on, off]*
- *Derivation Function [true, false]*

Algorithm Functional Test

The evaluator shall generate 16 test groups for each claimed function and supported derivation function state (on or off). Eight with prediction resistance enabled, and eight with prediction resistance turned off. For each of the eight test groups, four must have reseed enabled and four must have reseed turned off.

Each test group within the core sets of four test groups shall consist of 15 test cases each, such that minimum supported lengths and maximum supported lengths for the above parameters are used at least once across the four groups, and the rest are random lengths.

Correctness is determined by comparing the results from a known-good implementation using the same input parameters.

FCS_RBG.2 Random Bit Generation (External Seeding)

The inclusion of this selection-based component depends upon selection in:

- ***FCS_RBG.1.2***

FCS_RBG.2.1

The TSF shall be able to accept a minimum input of [**assignment**: *minimum input length greater than zero*] from a TSF interface for the purpose of obtaining entropy.

Application Note: The ST author claims this requirement when a TOE uses an external source of entropy to initialize or reseed a DRBG. The TOE collects enough input from the external entropy source such that the total measured entropy of the input is sufficient to initialize or reseed a DRBG. The ST author ensures that the assignment is completed with the minimum length of the input sufficient to initialize or reseed a DRBG.

Evaluation Activities

FCS_RBG.2
TSS
The evaluator shall verify that the TSS documents the minimum amount of input expected from external entropy sources. If this SFR is iterated, the evaluator shall check that the TSS indicates the minimum input size for each external entropy source.

Guidance
The evaluator shall verify that the Guidance describes any settings, operational requirements, or user input necessary for the proper function of the entropy sources.

Tests
The evaluator shall exercise the interface to determine whether the interface can handle at least the number of bits claimed in the assignment.

FCS_RBG.3 Random Bit Generation (Internal Seeding - Single Source)

The inclusion of this selection-based component depends upon selection in:

- ***FCS_RBG.1.2***

FCS_RBG.3.1

The TSF shall be able to seed the RBG using a [**selection, choose one of**: *TSF software-based noise source*, *TSF hardware-based noise source*] [**assignment**: *name of noise source*] with a minimum of [**assignment**: *number of bits*] bits of min-entropy.

Application Note: The ST author claims this requirement when a TOE uses a single internal source of entropy to initialize or reseed a DRBG. Seeding a DRBG is the same as initializing a DRBG.

Hardware-based noise sources are entropy sources whose primary function is noise generation, such as ring oscillators, diodes, and thermal noise. While a TOE may use software to collect the noise from these hardware sources, these are not software-based.

Software-based noise sources generate noise as a byproduct of their normal operation. Examples of software-based noise sources can be user or system-based events such as reading the least significant bits from an event timer, etc.

The TOE collects enough input from the internal noise source such that the total measured entropy of the input is sufficient to initialize or reseed a DRBG. The ST author ensures that the assignment is completed with the number of bits of the input sufficient to initialize or reseed a DRBG.

Evaluation Activities

FCS_RBG.3
TSS
The evaluator shall verify that the TSS documents the types of entropy sources selected in FCS_RBG.3.1 and indicates the amount of entropy provided by these sources.

Guidance
The evaluator shall verify that the Guidance describes any settings, operational requirements, or user input necessary for the proper function of the entropy sources.

Tests
The evaluator shall exercise the interface to determine whether the interface can handle at least the number of bits claimed in the assignment.

FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)

The inclusion of this selection-based component depends upon selection in:
- ***FCS_RBG.1.2***

FCS_RBG.4.1

The TSF shall be able to seed the DRBG using [**selection**: *[**assignment**: number] TSF software-based entropy source(s), [**assignment**: number] TSF hardware-based entropy source(s)*].

Application Note: The ST author claims this requirement when a TOE uses two or more internal sources of entropy to initialize or reseed a DRBG. Seeding a DRBG is the same as initializing a DRBG. FCS_RBG.5 defines the mechanism by which these sources are combined to ensure sufficient minimum entropy.

Evaluation Activities

FCS_RBG.4
TSS
The evaluator shall verify that the TSS documents number and the types of entropy sources selected in FCS_RBG.4.1.

Guidance
The evaluator shall verify that the Guidance describes any settings, operational requirements, or user input necessary for the proper function of the noise sources.

Tests
There are no test activities for this component.

FCS_RBG.5 Random Bit Generation (Combining Entropy Sources)

The inclusion of this selection-based component depends upon selection in:

- ***FCS_RBG.1.2***

FCS_RBG.5.1

The TSF shall [**selection**: *hash, concatenate and hash, XOR, input into a linear feedback shift register, [**assignment**: combining operation]*] [**selection**: *output from TSF entropy source(s), input from TSF interface(s) for obtaining entropy*] resulting in a minimum of [**assignment**: *number of bits*] bits of min-entropy to create the entropy input into the derivation function as defined in [**selection**: *ISO/IEC 18031:2011, NIST SP 800-90A Revision 1*]

Application Note: The ST author claims this requirement when a TOE combines two or more sources of entropy to initialize or reseed a DRBG. Seeding a DRBG is the same as initializing a DRBG. The ST author ensures that the assignment is completed with the number of bits of the combined entropy sufficient to initialize or reseed a DRBG.

One can apply NIST SP 800-90B (or AIS-31) statistical tests against internal noise sources (aka raw entropy) to confirm the min-entropy of the noise sources either in aggregate or individually. One should not apply NIST SP 800-90B (or AIS-31) statistical tests against external noise sources since the TOE is unable to enforce entropy requirements or conditioning requirements against external sources of entropy. However, the TSS may include estimates for min-entropy from external sources that contribute to the overall entropy requirements for the DRBG.

Evaluation Activities

FCS_RBG.5
TSS
The evaluator shall examine the TSS and verify that it documents the types of noise sources selected in FCS_RBG.5.1 and indicates the amount of entropy provided by these sources in combination.

Guidance
The evaluator shall verify that the Guidance describes any settings, operational requirements, or user input necessary for the proper function of the noise sources.

Tests
The evaluator shall ensure that the combination of claimed entropy sources can provide the at least the number of bits of entropy claimed in the assignment.

FCS_RBG.6 Random Bit Generation Service

This component must also be included in the ST if any of the following use cases are selected:
- ***CSfC EUD***

This component may also be included in the ST as if optional.

FCS_RBG.6.1

The TSF shall provide a [**selection**: *hardware, software, [**assignment**: other interface type]*] interface to make the DRBG output, as specified in FCS_RBG.1 Random Bit Generation (RBG), available as a service to entities outside of the TOE.

Application Note: This SFR must be included in the ST if the TOE provides an entropy source accessible to tenant software.

This requirement ensures that the TOE makes available entropy to any tenant that requires it.

Evaluation Activities

FCS_RBG.6
TSS
The evaluator shall examine the TSS to verify that it describes how the DRBG output is made available to entities outside the TOE.

Guidance

The evaluator shall examine the Guidance to verify that it describes how to configure and use the claimed interface(s) so that DRBG output is available to entities outside of the TOE.

Tests
The evaluator shall perform the following test:

The evaluator shall invoke the entropy source(s) from tenant software. The evaluator shall verify that the tenant acquires values from the interface.

FCS_STG_EXT.2 Key Storage Encryption

The inclusion of this selection-based component depends upon selection in:
- ***FCS_STG_EXT.1.1***

FCS_STG_EXT.2.1

The TSF shall encrypt [*AKs, SKs, KEKs, and [**selection**: long-term trusted channel key material, all software-based key storage, no other keys]*] using Key Wrapping as defined in FCS_COP.1/KeyWrap.

Application Note: This SFR is included in the ST if "software-based" is selected in FCS_STG_EXT.1.

Evaluation Activities

FCS_STG_EXT.2
TSS
The evaluator shall review the TSS to determine that the TSS describes the protection of symmetric keys, KEKs, long-term trusted channel key material, and software-based key storage as claimed in FCS_STG_EXT.2.1.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
There are no test activities for this component.

FCS_STG_EXT.3 Key Integrity Protection

The inclusion of this selection-based component depends upon selection in:
- ***FCS_STG_EXT.1.1***

FCS_STG_EXT.3.1

The TSF shall protect the integrity of any encrypted [*AKs, SKs, KEKs, and [**selection**: long-term trusted channel key material, all software-based key storage, no other keys]*] by using [**selection**:
- *Key wrapping accordance with FCS_COP.1/KeyWrap*
- *A keyed hash of the stored key in accordance with FCS_COP.1/KeyedHash*
- *A digital signature of the stored key in accordance with FCS_COP.1/SigGen using an asymmetric key that is protected in accordance with FCS_STG_EXT.2*
- *An immediate application of the key for decrypting the protected data followed by a successful verification of the decrypted data with previously known information*

].

Application Note: This SFR is included in the ST if "software-based" is selected in FCS_STG_EXT.1.

FCS_STG_EXT.3.2

The TSF shall verify the integrity of the [**selection**: *digital signature*, *MAC*] of the stored key prior to use of the key.

Application Note: This requirement is not applicable to derived keys that are not stored. It is not expected that a single key will be protected from corruption by multiple of these methods; however, a product may use one integrity-protection method for one type of key and a different method for other types of keys.

Evaluation Activities

FCS_STG_EXT.3
TSS
The evaluator shall examine the TSS and ensure that it contains a description of how the TOE protects the integrity of its keys.

Guidance
There are no additional Guidance evaluation activities for this component.

KMD
The documentation of the product's encryption key management should be detailed enough that, after reading, the evaluator will thoroughly understand the product's key management and how it meets the requirements to ensure the keys are adequately protected. This documentation should include an essay and diagrams. This documentation may be marked as developer proprietary.

Tests
There are no test activities for this component.

B.4 Class: User Data Protection (FDP)

FDP_ITC_EXT.1 Key/Credential Import

The inclusion of this selection-based component depends upon selection in:
- ***FCS_STG_EXT.1.2***

FDP_ITC_EXT.1.1

The TSF shall support importing keys/key material using [**selection**: *physically protected channels as specified in FTP_ITP_EXT.1, encrypted data buffers as specified in FTP_ITE_EXT.1, key distribution mechanisms as specified in FCS_CKM.2*].

FDP_ITC_EXT.1.2

The TSF shall verify the integrity of imported keys/key material using [**selection**: *cryptographic hash as specified in FCS_COP.1/Hash, keyed hash as specified in FCS_COP.1/KeyedHash, integrity-providing encryption algorithm as specified in FCS_COP.1/KeyWrap, digital signature as specified in FCS_COP.1/SigVer, integrity verification provided through FCS_CKM.2 key distribution mechanisms, integrity verification supported by FTP_ITC_EXT.1*].

Application Note: This SFR is included in the ST when "*importing keys/secrets into the TOE*" is selected in FCS_STG_EXT.1.

The way the TSF checks the integrity of the keys depends on the method of importation. For example, the encrypted data channel may provide data integrity as part of its service.

Evaluation Activities

FDP_ITC_EXT.1
TSS
The evaluator shall confirm the TSS contains descriptions of the supported methods the TSF uses to import keys and key material into the TOE. For each import method selected, the TSS shall describe integrity verification schemes employed.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
For each supported import method selected in FDP_ITC_EXT.1.1 and for each supported integrity verification method selected in FDP_ITC_EXT.1.2, used by the selected import method, provide one key/credential with valid integrity credentials, one with invalid integrity credentials (e.g. hash). The operations with invalid integrity credentials must result in error. The operations with valid integrity credentials must accept the keys/credentials.

B.5 Class: Identification and Authentication (FIA)

FIA_AFL_EXT.1 Authentication Failure Handling

The inclusion of this selection-based component depends upon selection in:

- ***FMT_SMR.1.1***

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Physically Secure Environment***
- ***Server-Class Platform, Enhanced Security Requirements***

FIA_AFL_EXT.1.1

The TSF shall consider password and [**selection**: ***[assignment**: other authentication mechanisms], no other authentication mechanisms*] as critical authentication mechanisms.

Application Note: This SFR is included in the ST if the "*Administrator*" role is selected in FMT_SMR.1, or if the "*Server-Class Platform, Basic*" or "*Server-Class Platform, Enhanced*" use cases are selected.

If this SFR is included in the ST, then FCS_CKM.6 must also be claimed.

This SFR specifies the actions to be taken in the event of multiple authentication failures.

This requirement applies to both critical and non-critical authentication mechanisms. The difference between the two is that excessive authentication failures of critical authentication mechanisms result in the actions defined in FIA_AFL_EXT.1.5.

If the TOE implements multiple Authentication Factor interfaces (for example, a DAR decryption interface, a lockscreen interface, an auxiliary boot mode interface), this element applies to all available interfaces. For example, a password is a critical authentication mechanism regardless of if it is being entered at the DAR decryption interface or at a lockscreen interface.

FIA_AFL_EXT.1.2

The TSF shall detect when a configurable positive non-zero integer within [**assignment**: *range of acceptable values for each authentication mechanism*] of [**selection, choose one of**: *unique*, *non-unique*] unsuccessful authentication attempts occur since the last successful authentication for each authentication mechanism.

Application Note: The positive integer is configured according to Table 3 in FMT_SMF.1.1.

An unique authentication attempt is defined as any attempt to verify an authentication attempt in which the input is different from a previous attempt. "Unique" must be selected if the authentication system increments the counter only for unique unsuccessful authentication attempts. For example, if the same incorrect password is attempted twice the authentication system increments the counter once. "Non-unique" must be selected if the authentication system increments the counter for each unsuccessful authentication attempt, regardless of whether the input is unique. For example, if the same incorrect password is attempted twice the authentication system increments the counter twice.

If the TOE supports multiple authentication mechanisms per FIA_UAU.5.1, this element applies to all authentication mechanisms. It is acceptable for each authentication mechanism to utilize an independent counter or for multiple authentication mechanisms to utilize a shared counter. The interaction between the authentication factors with regard to the authentication counter must be in accordance with FIA_UAU.5.2.

If the TOE implements multiple Authentication Factor interfaces (for example, a DAR decryption interface, a lockscreen interface, an auxiliary boot mode interface), this element applies to all available interfaces. However, it is acceptable for each Authentication Factor interface to be configurable with a different number of unsuccessful authentication attempts.

FIA_AFL_EXT.1.3

The TSF shall maintain the number of unsuccessful authentication attempts that have occurred upon power off if the minimum boot time of the system is shorter than the lockout time specified in FIA_AFL_EXT.1.5.

Application Note: The purpose of this requirement is to prevent hammering attacks focused on a device's pre-OS-firmware from bypassing the actions in FIA_AFL_EXT.1.5 by power cycling the system in order to zero the authentication failure count. The intention is to protect the pre-OS firmware without making assumptions as to boot duration per device. This purpose is achieved by default if the minimum reboot time of the system is greater than the timeout penalty specified in FIA_AFL_EXT.1.5.

If the actions specified in FIA_AFL_EXT.1.5 are device wipe or a non-time-limited lockout, or if the minimum reboot time is shorter than the specified lockout time, then the authentication failure count must be maintained across power cycles. The variation of boot duration of individual devices and the configurability of FIA_AFL_EXT.1.5 may create scenarios where some devices are compliant by default (specifically slow-booting servers and workstations) while other devices (specifically fast-booting desktops and notebooks) may need to implement this requirement.

The TOE may implement an Authentication Factor interface that precedes another Authentication Factor interface in the boot sequence (for example, a volume DAR decryption interface which precedes the lockscreen interface) before the user can access the GPCP. In this situation, because the user must successfully authenticate to the first interface to access the second, the number of unsuccessful authentication attempts need not be maintained for the second interface.

FIA_AFL_EXT.1.4

When the defined number of unsuccessful authentication attempts has exceeded the maximum allowed for a given authentication mechanism, all future authentication attempts shall be limited to other available authentication mechanisms, unless the given mechanism is designated as a critical authentication mechanism.

Application Note: See FIA_AFL_EXT.1.5 for exceeding the maximum failure threshold for critical authentication mechanisms.

In accordance with FIA_AFL_EXT.1.3, this requirement also applies after the TOE is powered off and powered back on.

FIA_AFL_EXT.1.5

When the defined number of unsuccessful authentication attempts for the last available authentication mechanism or a critical authentication mechanism has been surpassed, the TSF shall [**selection**:

- *perform a wipe of all protected data*
- *exclude the current Administrator from further authentication attempts*
- *exclude the current Administrator from further authentication attempts for [**assignment**: a period of time greater than zero seconds]*
- *exclude the current Administrator from further authentication attempts for [**assignment**: a period of time greater than the minimum boot time of the system]*

].

Application Note: The "current Administrator" is the entity attempting to authenticate to the TOE that has run afoul of the limit on authentication attempts. For platforms that support multiple Administrator identities, only the identity that has run afoul is punished. For platforms without such support, these actions are effectively applied to the authentication mechanism rather than a specific user.

Wipe is performed in accordance with FCS_CKM.6. Protected data is all non-TSF data, including all user or enterprise data. Some or all of this data may be considered sensitive data as well.

If the TOE implements multiple Authentication Factor interfaces (for example, a DAR decryption interface, a lockscreen interface, an auxiliary boot mode interface), this element applies to all available interfaces.

FIA_AFL_EXT.1.6

The TSF shall increment the number of unsuccessful authentication attempts prior to notifying the user that the authentication was unsuccessful.

Application Note: This requirement is to ensure that if power is cut to the device directly after an authentication attempt, the counter will be incremented to reflect that attempt.

Evaluation Activities

FIA_AFL_EXT.1

TSS

The evaluator shall ensure that the TSS describes that a value corresponding to the number of unsuccessful authentication attempts since the last successful authentication is kept for each Authentication Factor interface. The evaluator shall ensure that this description also includes if and how this value is maintained when the TOE loses power, either through a graceful powered off or an ungraceful loss of power. The evaluator shall ensure that if the value is not maintained, the interface is after another interface in the boot sequence for which the value is maintained.

If the TOE supports multiple authentication mechanisms, the evaluator shall ensure that this description also includes how the unsuccessful authentication attempts for each mechanism selected in FIA_UAU.5.1 is handled. The evaluator shall verify that the TSS describes if each authentication mechanism utilizes its own counter or if multiple authentication mechanisms utilize a shared counter. If multiple authentication mechanisms utilize a shared counter, the evaluator shall verify that the TSS describes this interaction.

The evaluator shall confirm that the TSS describes how the process used to determine if the authentication attempt was successful. The evaluator shall ensure that the counter would be updated even if power to the device is cut immediately following notifying the TOE user if the authentication attempt was successful or not.

Guidance
The evaluator shall verify that the AGD describes how the Administrator configures the maximum number of unique unsuccessful authentication attempts, and the lockout time period, if applicable.

The evaluator shall verify that the AGD describes how an Administrator may recover from authentication failure when another Administrator is locked out.

Tests
The evaluator shall configure the device with all authentication mechanisms selected in FIA_UAU.5.1, and configure a maximum number of unsuccessful authentication attempts for each mechanism.

- *Test FIA_AFL_EXT.1:1: The evaluator shall for each authentication mechanism make unsuccessful authentication attempts until the maximum is exceeded and verify that the number of failures corresponds to the configured maximum and that no further authentication attempts can be made using that mechanism.*
- *Test FIA_AFL_EXT.1:2: [conditional] If the mechanism is critical or if all authentication mechanisms are exhausted, then if "perform a wipe of all protected data" is selected in FIA_AFL_EXT.1.5 the evaluator shall verify that the wipe is implemented.*
- *Test FIA_AFL_EXT.1:3: [conditional] If the mechanism is critical or if all authentication mechanisms are exhausted, then if "exclude the current User/Administrator from further authentication attempts" is selected in FIA_AFL_EXT.1.5 the evaluator shall verify that the User/Administrator can make no further authentication attempts.*
- *Test FIA_AFL_EXT.1:4: [conditional] If the mechanism is critical or if all authentication mechanisms are exhausted, then if "exclude the current User/Administrator from further authentication attempts for a period of [assignment: greater than zero seconds] time" is selected in FIA_AFL_EXT.1.5 the evaluator shall verify that the User/Administrator can make no further authentication attempts until the specified time period has expired.*

FIA_PMG_EXT.1 Password Management

The inclusion of this selection-based component depends upon selection in:

- ***FMT_SMR.1.1***

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Physically Secure Environment***
- ***Server-Class Platform, Enhanced Security Requirements***

FIA_PMG_EXT.1.1

The TSF shall support the following for the Password Authentication Factor:

1. Passwords shall be able to be composed of any combination of [**selection**: *upper and lower case letters*, *[**assignment**: a character set of at least 52 characters]*], numbers, and special characters: [**selection**: *"!"*, *"@"*, *"#"*, *"$"*, *"%"*, *"^"*, *"&"*, *"*"*, *"("*, *")"*, *[**assignment**: other characters]*]
2. Password length of at least [**assignment**: *an integer greater than or equal to 14*] characters shall be supported.

Application Note: This SFR is included in the ST if the "*Administrator*" role is selected in FMT_SMR.1, or if the "*Server-Class Platform, Basic*" or "*Server-Class Platform, Enhanced*" use cases are selected.

While some corporate policies require passwords of 14 characters or better, the use of a Root Encryption Key for DAR protection and key storage protection and the anti-hammer requirement (FIA_TRT_EXT.1) addresses the threat of attackers with physical access using much smaller and less complex passwords.

The ST Author selects the character set: either the upper and lower case Basic Latin letters or another assigned character set containing at least 52 characters. The assigned character set must be well defined: either according to an international encoding standard (such as Unicode) or defined in the assignment by the ST Author. The ST Author also selects the special characters that are supported by TOE; they may optionally list additional special characters supported using the assignment.

Evaluation Activities

FIA_PMG_EXT.1
TSS
There are no additional TSS evaluation activities for this component.

Guidance
The evaluator shall examine the AGD to determine that it provides guidance to security administrators on the composition of strong passwords, and that it provides instructions on setting the minimum password length. The evaluator shall also perform the following tests. Note that one or more of these tests can be performed with a single test case.

Tests
The evaluator shall perform the following test:

The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

FIA_UAU.5 Multiple Authentication Mechanisms

The inclusion of this selection-based component depends upon selection in:

- ***FMT_SMR.1.1***

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Physically Secure Environment***
- ***Server-Class Platform, Enhanced Security Requirements***

FIA_UAU.5.1

The TSF shall provide [*password and [**selection**: certificate-based authentication, public key-based authentication, biometric authentication, no other authentication mechanism]*] to support user authentication.

Application Note: This SFR is included in the ST if the "*Administrator*" role is selected in FMT_SMR.1, or if the "*Server-Class Platform, Basic*" or "*Server-Class Platform, Enhanced*" use cases are selected.

A "user" in the context of this SFR is an Administrator.

The TSF must support a Password Authentication Factor and may optionally implement a biometric authentication factor.

The Password Authentication Factor is configured according to FIA_PMG_EXT.1.

If "*X.509 certificate-based authentication*" is selected, then the ST must include FIA_X509_EXT.1 and FIA_X509_EXT.2 from Functional Package for X.509, version 1.0.

If "*public key-based authentication*" is selected, then the ST must claim the Functional Package for Secure Shell (SSH), version 2.0.

FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the [**assignment**: *rules describing how the multiple authentication mechanisms provide authentication*].

Application Note: Rules regarding how the authentication factors interact in terms of unsuccessful authentication are covered in FIA_AFL_EXT.1.

Evaluation Activities

FIA_UAU.5
TSS
The evaluator shall ensure that the TSS describes each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication.

Specifically, for all authentication mechanisms specified in FIA_UAU.5.1, the evaluator shall ensure that the TSS describes the rules as to how each authentication mechanism is used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or biometric sample was entered), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism

can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used). If multiple Biometric Authentication Factors (BAF) are supported per FIA_UAU.5.1, the interaction between the BAFs must be described. For example, whether the multiple BAFs can be enabled at the same time.

Guidance
The evaluator shall verify that configuration information for each authentication mechanism is addressed in the AGD guidance.

Tests
The evaluator shall perform the following tests:

- *Test FIA_UAU.5:1: For each authentication mechanism selected in FIA_UAU.5.1, the evaluator shall enable that mechanism and verify that it can be used to authenticate the user at the specified authentication factor interfaces.*
- *Test FIA_UAU.5:2: For each authentication mechanism rule, the evaluator shall ensure that the authentication mechanism behaves accordingly.*

FIA_UAU.7 Protected Authentication Feedback

The inclusion of this selection-based component depends upon selection in:

- ***FMT_SMR.1.1***

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Physically Secure Environment***
- ***Server-Class Platform, Enhanced Security Requirements***

FIA_UAU.7.1

The TSF shall provide only [*obscured feedback*] to the user while the authentication is in progress.

Application Note: This SFR is included in the ST if the "*Administrator*" role is selected in FMT_SMR.1, or if the "*Server-Class Platform*, *Basic*" or "*Server-Class Platform*, *Enhanced*" use cases are selected.

This requirement applies to all authentication mechanisms specified in FIA_UAU.5.1 that provide feedback to a user or Administrator during authentication.

For authentication mechanisms that require the user or Administrator to enter a password or PIN, the TSF may briefly (1 second or less) display each character or provide an option to allow the user to unmask the user input; however, the user input must be obscured by default.

If a BAF is selected in FIA_UAU.5.1, the TSF must not display sensitive information regarding the biometric that could aid an adversary in identifying or spoofing the respective biometric characteristics of a given human user. While it is true that biometric samples, by themselves, are not secret, the analysis performed by the respective biometric algorithms, as well as output data from these biometric algorithms, is considered sensitive and must be kept secret. Where applicable, the TSF must not reveal or make public the reasons for authentication failure.

In the cases of SSH- or X.509-based authentication, the TSF must likewise not display sensitive information regarding the authentication factors that could aid an adversary in spoofing or circumventing the authentication protocols.

Evaluation Activities

FIA_UAU.7
TSS
The evaluator shall ensure that the TSS describes the means of obscuring the authentication information for all authentication methods specified in FIA_UAU.5.1.

Guidance
The evaluator shall verify that any configuration of this requirement is addressed in the AGD guidance and that user authentication input is obscured by default.

Tests

The evaluator shall perform the following tests:

- *Test FIA_UAU.7:1: The evaluator shall enter passwords on the device, including at least the Password Authentication Factor at lockscreen, and verify that the password is not displayed on the device.*
- *Test FIA_UAU.7:2: [conditional] For each Biometric Authentication Factor (BAF) selected in FIA_UAU.5.1, the evaluator shall authenticate by producing a biometric sample at lockscreen. As the biometric algorithms are performed, the evaluator shall verify that sensitive images, audio, or other information identifying the user are kept secret and are not revealed to the user. Additionally, the evaluator shall produce a biometric sample that fails to authenticate and verify that the reason(s) for authentication failure (user mismatch, low sample quality, etc.) are not revealed to the user. It is acceptable for the BAF to state that it was unable to physically read the biometric sample, for example, if the sensor is unclean or the biometric sample was removed too quickly. However, specifics regarding why the presented biometric sample failed authentication shall not be revealed to the user. [conditional] For each SSH- or X.509-based authentication mechanism, the evaluator shall examine whether the TSF displays sensitive information during the authentication process for both successful and failed authentication attempts.*

FIA_UIA_EXT.1 Administrator Authentication

The inclusion of this selection-based component depends upon selection in:

- ***FMT_SMR.1.1***

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Physically Secure Environment***
- ***Server-Class Platform, Enhanced Security Requirements***

FIA_UIA_EXT.1.1

The TSF shall require Administrators to be successfully authenticated using one of the methods in FIA_UAU.5 before allowing any TSF-mediated management function to be performed by that Administrator.

Application Note: This SFR is included in the ST if the "*Administrator*" role is selected in FMT_SMR.1, or if the "*Server-Class Platform, Basic*" or "*Server-Class Platform, Enhanced*" use cases are selected.

Ordinary unprivileged users of the platform need not authenticate to the platform, though they may well have to authenticate themselves to tenant software such as an Operating System.

The TSF-mediated management functions are listed in the management functions table (Table 3) in FMT_SMF.1.

Evaluation Activities

FIA_UIA_EXT.1
TSS
The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the platform. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a "successful logon."

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

Tests
There are no test activities for this component.

B.6 Class: Protection of the TSF (FPT)

FPT_FLS.1 Failure with Preservation of Secure State

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.5.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_CKM.1/AKG***
- ***FCS_CKM.1/SKG***
- ***FCS_COP.1/SigGen***
- ***FCS_IPSEC_EXT.1***

This component may also be included in the ST as if optional.

FPT_FLS.1.1

The TSF shall preserve a secure state when the following types of failures occur: [*DRBG self-test failure*].

Application Note: The intent of this requirement is to ensure that cryptographic services requiring random bit generation cannot be performed if a failure of a self-test defined in FPT_TST.1 occurs.

Evaluation Activities

FPT_FLS.1
TSS
The evaluator shall verify that the TSF describes how the TOE enters an error state in the event of a DRBG self-test failure.

Guidance
The evaluator shall verify that the guidance documentation describes the error state that results from a DRBG self-test failure and the actions that a user or administrator should take in response to attempt to resolve the error state.

Tests
There are no test activities for this component.

FPT_PHP.1 Passive detection of physical attack

This component must also be included in the ST if any of the following use cases are selected:

- ***Portable Clients (laptops, tablets), Enhanced Security Requirements***

This component may also be included in the ST as if optional, but may be mandatory in the future.

FPT_PHP.1.1

The TSF shall provide unambiguous detection of physical tampering that can compromise the TSF.

FPT_PHP.1.2

The TSF shall provide the capability to determine whether physical tampering with the TSF's devices or TSF's elements has occurred.

Application Note: This SFR should be included in the ST if the TOE implements the following use cases:

1. Portable Clients (laptops, tablets), Enhanced

The audit event "Detection of intrusion" should be claimed if the TOE is capable of generating an audit event in circumstances where an intrusion is detected.

Evaluation Activities

FPT_PHP.1
TSS
The evaluator shall examine the TSS to ensure it describes the methods used by the TOE to detect physical tampering and how tampering is indicated when detected.

Guidance

The evaluator shall ensure that the AGD describes how the TOE indicates to users and Administrators that it has detected tampering.

Tests
The evaluator shall verify that attempts to open the TOE enclosure result in indications consistent with the operational guidance. Such indications could include damaged tamper seals, logged events, or other physical or electronic manifestations.

FPT_PHP.2 Notification of Physical Attack

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Enhanced Security Requirements***

This component may also be included in the ST as if optional, but may be mandatory in the future.

FPT_PHP.2.1

The TSF shall provide unambiguous detection of physical tampering that can compromise the TSF.

FPT_PHP.2.2

The TSF shall provide the capability to determine whether physical tampering with the TSF's devices or TSF's elements has occurred.

FPT_PHP.2.3

For [**assignment**: *list of TSF devices/elements for which active detection is required*], the TSF shall monitor the devices and elements and notify [**assignment**: *a designated user or role*] when physical tampering with the TSF's devices or TSF's elements has occurred.

Application Note: This SFR should be included in the ST if the TOE implements the following use cases:

1. Server-Class Platform, Enhanced

FPT_PHP.2 is hierarchical to FPT_PHP.1 which means that all requirements of FPT_PHP.1 are also included as part of FPT_PHP.2. A TOE that conforms to FPT_PHP.2 therefore does not claim FPT_PHP.1.

The audit event "Detection of intrusion" should be claimed if the TOE is capable of generating an audit event in circumstances where an intrusion is detected.

Evaluation Activities

FPT_PHP.2
TSS
The evaluator shall examine the TSS to ensure it describes the methods used by the TOE to detect physical tampering and how the TOE will respond when physical tampering has been detected for each device/element specified in FPT_PHP.2.3.

Guidance
The evaluator shall ensure that the AGD describes how the TOE notifies users or Administrators that it has detected tampering.

Tests
The evaluator shall perform the following tests:

- *Test FPT_PHP.2:1: The evaluator shall verify that attempts to open the TOE enclosure result in indications consistent with the operational guidance. Such indications could include damaged tamper seals, logged events, or other physical or electronic manifestations.*
- *Test FPT_PHP.2:2: For each device/element listed in FPT_PHP.2.3, the evaluator shall verify that attempts to physically tamper with the device/element results in notification to the designated users or roles consistent with the operational guidance.*

FPT_PHP.3 Resistance to Physical Attack

This component must also be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Enhanced Security Requirements***
- ***Tactical EUD***

FPT_PHP.3.1

The TSF shall resist [**assignment**: *physical tampering scenarios*] to the [**assignment**: *list of TSF devices/elements*] by responding automatically such that the SFRs are always enforced.

Application Note: This SFR should be included in the ST if the TOE implements the following use cases:

1. Server-Class Platform, Enhanced
2. Tactical EUD

Evaluation Activities

FPT_PHP.3
TSS
The evaluator shall examine the TSS to ensure it describes the methods used by the TOE to detect physical tampering and how the TOE will respond when physical tampering has been detected such that SFRs are always enforced.

Guidance
The evaluator shall examine the AGD to ensure that it describes the expected response of the TOE when physical tampering is detected.

Tests
The evaluator shall perform the following test:

For each physical tampering scenario and device/element listed in FPT_PHP.3.1, the evaluator shall verify that tampering attempts result in a response from the TSF consistent with the operational guidance.

FPT_RVR_EXT.1 Platform Firmware Recovery

The inclusion of this selection-based component depends upon selection in:

- ***FPT_ROT_EXT.2.2,***
- ***FPT_TUD_EXT.2.5,***
- ***FPT_TUD_EXT.3.4***

FPT_RVR_EXT.1.1

The TSF shall implement a mechanism for recovering from boot firmware failure consisting of [**selection**:

- *the secure local update mechanism described in FPT_TUD_EXT.4*
- *installation of a known-good or recovery firmware image*
- *reversion to the prior firmware image*
- *installation of a recovery image that puts the TOE into a maintenance mode*

].

Application Note: This SFR must be included in the ST if:

- "*Initiate a recovery process as specified in FPT_RVR_EXT.1*" is selected in FPT_ROT_EXT.2.2,
- "*Initiate a recovery process as specified in FPT_RVR_EXT.1*" is selected in FPT_TUD_EXT.2.5,
- "*Initiate a recovery process as specified in FPT_RVR_EXT.1*" is selected in FPT_TUD_EXT.3.4,
- The TOE implements a recovery mechanism for firmware corruption not necessarily related to integrity or update failure.

If the ST Author selects "*the secure local update mechanism described in FPT_TUD_EXT.4*," then FPT_TUD_EXT.4 must be claimed in the ST.

As indicated above, in addition to integrity or update failure, the TOE may use a recovery mechanism to deal with non-security-related failures, such as a power outage during update or a power surge during normal operation.

The recovery process may be initiated automatically on failure, as the result of physically present user

action, or as the result of pre-configured policy. The action taken may depend on the nature of the failure as specified in FPT_ROT_EXT.2.2 and FPT_TUD_EXT.2.5.

Evaluation Activities

FPT_RVR_EXT.1
TSS
The evaluator shall examine the TSS section to confirm that it describes how the platform firmware recovery mechanism works and the conditions under which it is invoked.

Guidance
The evaluator shall examine the AGD to ensure that is describes how to configure the conditions under which the recovery mechanism is initiated (if configurable).

Tests
The evaluator shall perform the following tests:

- *Test FPT_RVR_EXT.1:1: To test this requirement, the evaluator shall trigger the recovery process either by forcing an update error or a boot integrity failure and observing that the recovery process has been initiated.*
- *Test FPT_RVR_EXT.1:2: The evaluator will engage with the recovery process as necessary, and after recovery will determine the version of the current firmware image. The test is passed if the resultant image is as expected in accordance with policy and the selections in FPT_RVR_EXT.1.1. If the recovery process uses the secure local update process as specified in FPT_TUD_EXT.4, then this test is satisfied by testing of that requirement.*

FPT_TST.1 TSF Self-Testing

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.5.1***

This component must be included in the ST if any of the following SFRs are included:

- ***FCS_CKM.1/AKG***
- ***FCS_CKM.1/SKG***
- ***FCS_COP.1/SigGen***
- ***FCS_IPSEC_EXT.1***

This component may also be included in the ST as if optional.

FPT_TST.1.1

The TSF shall run a suite of the following self-tests [**selection**: *during initial start-up, periodically during normal operation, at the request of the authorized user, at the conditions [**assignment**: conditions under which self-test should occur]*] to demonstrate the correct operation of [*TSF DRBG specified in FCS_RBG.1*]: [**assignment**: *DRBG health tests*].

FPT_TST.1.2

The TSF shall provide authorized users with the capability to verify the integrity of [*[DRBG seed/output data]*].

FPT_TST.1.3

The TSF shall provide authorized users with the capability to verify the integrity of [*[TSF DRBG specified in FCS_RBG.1]*].

Application Note: This SFR is a required dependency of FCS_RBG.1. It is intended to require that any DRBG implemented by the TOE undergo health testing to ensure that the random bit generation functionality has not been degraded. If the TSF supports multiple DRBGs, this SFR should be iterated to describe the self-test behavior for each.

Evaluation Activities

FPT_TST.1
TSS
The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF along with how they are run. This description should include an outline of what the tests are actually doing. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the DRBG is operating correctly.

Note that this information may also be placed in the entropy documentation specified by .

Guidance
If a self-test can be executed at the request of an authorized user, the evaluator shall verify that the operational guidance provides instructions on how to execute that self-test.

Tests
For each self-test, the evaluator shall verify that evidence is produced that the self-test is executed when specified by FPT_TST.1.1.

If a self-test can be executed at the request of an authorized user, the evaluator shall verify that following the steps documented in the operational guidance to perform the self-test will result in execution of the self-test.

FPT_TUD_EXT.2 Platform Firmware Authenticated Update Mechanism

The inclusion of this selection-based component depends upon selection in:

- ***FPT_TUD_EXT.1.1***

FPT_TUD_EXT.2.1

The TSF shall authenticate the source of all platform firmware updates using a digital signature algorithm specified in FCS_COP.1/**SigVer** and using a key store that contains [**selection**: *the public key*, *hash value of the public key*].

Application Note: This SFR must be included in the ST if "*an authenticated platform firmware update mechanism as described in FPT_TUD_EXT.2*" is selected in FPT_TUD_EXT.1.1.

The ST must include FCS_COP.1/Hash if "*hash value of the public key*" is selected.

FPT_TUD_EXT.2.2

The TSF shall allow installation of updates only if the digital signature has been successfully verified as specified in FCS_COP.1/**SigVer** and [**selection**: *the version number of the platform firmware update is more recent than the version number of the current installed platform firmware*, *no other conditions*].

Application Note: The ST Author should select "*the version number...* " if the TSF supports rollback prevention. That is, the TSF does not allow "update" to an older version of the platform firmware. In general, rollback should be permitted only through a secure local update mechanism at the express direction of an physically present Administrator or User.

FPT_TUD_EXT.2.3

The TSF shall include a platform firmware version identifier that is accessible by the update mechanism and includes information that enables the update mechanism to determine the relative order of updates.

FPT_TUD_EXT.2.4

The TSF shall provide an observable indication of the success or failure of the update operation.

Application Note: For success, this indication should include the version number of the newly installed firmware. Notification of failure could include generation of an audit event by a management subsystem, a beep code, an updated version number on a splash screen, or simple failure to continue functioning.

FPT_TUD_EXT.2.5

The TOE shall take the following actions if a platform firmware integrity, authenticity, or rollback-prevention check fails, or a platform firmware update fails for any other reason:

- Do not install the update,
- Generate an audit event (optional),

and [**selection, choose one of**:

- *Continue execution*
- *Halt*
- *Stop all execution and shut down*
- *Initiate recovery as specified in FPT_RVR_EXT.1*

]
[**selection, choose one of**:

- *automatically*

- *in accordance with administrator-configurable policy*
- *by express determination of a User*

].

Application Note: If "*generate an audit event*" is selected, then FAU_GEN.1 and the other audit requirements must be claimed.

If "*Initiate recovery as specified in FPT_RVR_EXT.1*" is selected, then FPT_RVR_EXT.1 must be included in the ST.

The platform firmware authenticated update mechanism employs digital signatures to ensure the authenticity of the firmware update image. The TSF includes a signature verification algorithm and a key store containing the public key needed to verify the signature on the firmware update image.

A hash of the public key may be stored if a copy of the public key is provided with firmware update images. In this case, the update mechanism hashes the public key provided with the update image, and ensures that it matches a hash which appears in the key store before using the provided public key to verify the signature of the update image. If the hash of the public key is selected, the ST Author may iterate the FCS_COP.1/Hash requirement to specify the hashing functions used.

An indication of success or failure can be generation of an audit event by a management subsystem, a beep code, an updated version number on a splash screen, or simple failure to continue functioning.

If the update mechanism generates audit events, the ST Author must make the appropriate selections from the audit events table (Table 9).

In the selection "by express determination of a User," the "User" could be more properly expressed as the "operator." It can be either a User or an Administrator, but the distinction is not important in the context of this requirement.

Evaluation Activities

FPT_TUD_EXT.2

TSS
The evaluator shall ensure that the TSS includes a comprehensive description of how the authentication of platform firmware updates is implemented by the TSF. The TSS should cover the initialization process and the activities that are performed to ensure that the digital signature of the update image is verified before modification of the firmware.

The evaluator shall examine the TSF to ensure that it describes the platform firmware version identifier and explains its meaning and encoding.

The evaluator shall also ensure that the TSS describes the actions taken by the TSF if an update image fails authentication.

Guidance
The evaluator shall examine the AGD to ensure that it describes the process for updating the platform firmware.

The evaluator shall examine the AGD to ensure that it documents the observable indications of update success or failure, and that it describes how to access the platform firmware version indicators.

Tests
The evaluator shall perform the following tests:

- *Test FPT_TUD_EXT.2:1: The evaluator determines the current version of the platform firmware, and obtains or produces a valid, authentic, and permissible update image of platform firmware. The evaluator initiates an update using this image through the process described in the operational guidance. After the process is complete, the evaluator checks the current firmware version to ensure that the new firmware version matches that of the update.*
- *Test FPT_TUD_EXT.2:2: The evaluator performs the same test, this time using a valid update image that is signed with an incorrect key. The update must fail.*
- *Test FPT_TUD_EXT.2:3: The evaluator performs the same test, this time using an update image that is corrupted but is signed with the correct key. The update must fail.*
- *Test FPT_TUD_EXT.2:4: The evaluator performs the same test, this time using a valid update image that is not signed. The update must fail.*
- *Test FPT_TUD_EXT.2:5: [conditional] If the TSF implements rollback protections, the evaluator performs the same test, this time using a valid, signed update image that has an earlier version number than the currently installed firmware. The update must fail.*

FPT_TUD_EXT.3 Platform Firmware Delayed-Authentication Update Mechanism

The inclusion of this selection-based component depends upon selection in:

- ***FPT_TUD_EXT.1.1***

FPT_TUD_EXT.3.1

The TSF shall allow execution or use of platform firmware updates only if new platform firmware is integrity- and authenticity-checked using the mechanism described in FPT_ROT_EXT.2 prior to its execution or use, and [**selection**: *the version number of the platform firmware update is more recent than the version number of the previously installed platform firmware*, *no other conditions*].

Application Note: This requirement must be included in the ST if "*implement a delayed-authentication platform firmware update mechanism as described in FPT_TUD_EXT.3*" is selected in FPT_TUD_EXT.1.1.

This update mechanism does not require an integrity or authenticity check prior to installation, but the newly installed platform firmware must have its integrity and authenticity verified prior to being executed or used. This update mechanism takes advantage of the existing FPT_ROT_EXT.2 requirement to avoid having to verify the integrity and authenticity of an update package at install time.

The ST Author should select "*the version number of the platform firmware update is more recent than the version number of the previously installed platform firmware*" if the TSF supports rollback prevention.

FPT_TUD_EXT.3.2

The TSF shall include an observable platform firmware version identifier that is accessible by the update mechanism and includes information that enables the update mechanism to determine the relative order of updates.

FPT_TUD_EXT.3.3

The TSF shall provide an observable indication of the success or failure of the update operation.

Application Note: For success, this should at least include an indication of the version number of the newly installed firmware. Notification of failure could include generation of an audit event by a management subsystem, a beep code, an updated version number on a splash screen, or simple failure to continue functioning.

FPT_TUD_EXT.3.4

The TOE shall take the following actions if a platform firmware update integrity, authentication, or rollback-prevention check fails, or a platform firmware update fails for any other reason:

- Do not install the update,
- Generate an audit event (optional),

and [**selection, choose one of**:

- *Continue execution*
- *Halt*
- *Stop all execution and shut down*
- *Initiate recovery as specified in FPT_RVR_EXT.1*

]
[**selection, choose one of**:

- *automatically*
- *in accordance with administrator-configurable policy*
- *by express determination of a User*

].

Application Note: If "*generate an audit event*" is selected, then FAU_GEN.1 and the other audit SFRs must be claimed.

If "*Initiate recovery as specified in FPT_RVR_EXT.1*" is selected, then FPT_RVR_EXT.1 must be included in the ST.

The platform firmware unauthenticated update mechanism installs platform firmware updates without first checking their integrity or authenticity. Instead, this mechanism either invokes a special authentication/integrity check on the firmware *in situ* after install or relies on the firmware checks required by FPT_ROT_EXT.2 to ensure the integrity and authenticity of the update image. In either case, the integrity and authenticity of the update must be verified before the updated firmware is executed or used.

Likewise, if the TSF implements rollback prevention, this check must be made before the newly installed firmware is executed.

In the selection "by express determination of a User," the "User" could be more properly expressed as the "operator." It can be either a User or an Administrator, but the distinction is not important in the context of this requirement.

Evaluation Activities

FPT_TUD_EXT.3

TSS
The evaluator shall ensure that the TSS includes a comprehensive description of how the authentication of platform firmware updates is implemented by the TSF. The TSS should cover the initialization process and the activities that are performed to ensure that the digital signature of the update image is verified before it is executed or used.

The evaluator shall examine the TSF to ensure that it describes the platform firmware version identifier and explains its meaning and encoding.

The evaluator shall also ensure that the TSS describes the actions taken by the TSF if an update image fails authentication, integrity, or rollback-prevention checks.

Guidance
The evaluator shall examine the AGD to ensure that it describes the process for updating the platform firmware.

The evaluator shall examine the AGD to ensure that it documents the observable indications of update success or failure, and that it describes how to access the platform firmware version indicators.

Tests
The evaluator shall perform the following tests:

- *Test FPT_TUD_EXT.3:1: The evaluator determines the current version of the platform firmware, and obtains or produces a valid, authentic, and permissible update image of platform firmware. The evaluator initiates an update using this image through the process described in the operational guidance. After the process is complete, the evaluator checks the current firmware version to ensure that the new firmware version matches that of the update.*
- *Test FPT_TUD_EXT.3:2: The evaluator performs the same test, this time using a inauthentic update image. The update code must fail to execute.*
- *Test FPT_TUD_EXT.3:3: The evaluator performs the same test, this time using an update image that is corrupted but is otherwise authentic. The update code must fail to execute.*
- *Test FPT_TUD_EXT.3:4: [conditional] If the TSF implements rollback protections, the evaluator performs the same test, this time using a valid, signed update image that is has an earlier version number than the currently installed firmware. The update code must fail to execute.*

FPT_TUD_EXT.4 Secure Local Platform Firmware Update Mechanism

The inclusion of this selection-based component depends upon selection in:

- ***FPT_RVR_EXT.1.1,***
- ***FPT_TUD_EXT.1.1***

FPT_TUD_EXT.4.1

The TSF shall provide a secure local update mechanism that requires an assertion of physical access to the TOE before installation of an update.

FPT_TUD_EXT.4.2

A user shall assert physical presence to the TSF through: [**selection**:

- *login to the TOE from a physically connected console or terminal*
- *physical connection of a jumper or cable*
- *connection to a debug port*
- *[**assignment**: description of other mechanism for asserting physical presence]*

].

Application Note: This requirement is included in the ST if "*the secure local update mechanism described in FPT_TUD_EXT.4*" is selected in FPT_RVR_EXT.1.1 or "*implement a secure local platform firmware update mechanism described in FPT_TUD_EXT.4*" is selected in FPT_TUD_EXT.1.1.

This requirement pertains to platform firmware update mechanisms that do not use the authentication-based update mechanism described in FPT_TUD_EXT.2 or the delayed-authentication described in

FPT_TUD_EXT.3. The secure local update mechanism ensures the authenticity and integrity of the firmware update image by requiring a user to be physically present at the TOE. An assertion of physical presence can take the form, for example, of requiring entry of a password at a boot screen, unlocking of a physical lock (e.g., a motherboard jumper), or inserting a USB cable before permitting platform firmware to be updated.

There is no requirement that the local update mechanism support rollback prevention.

The local update mechanism must be a designed mechanism. If update can be accomplished only through the physical removal and replacement of a part, then that is not a secure local update mechanism, and "make no provision for platform firmware update" should be selected in FPT_TUD_EXT.1.1.

FPT_TUD_EXT.4.3

The TSF shall include a platform firmware version identifier that is accessible by the update mechanism or to the user who asserts physical presence.

FPT_TUD_EXT.4.4

The TSF shall provide an observable indication of the success or failure of the update operation.

Application Note: For success, this indication should include the version number of the newly installed firmware. Notification of failure could be through a beep code, an indication on a splash screen, or simple failure to continue functioning.

Evaluation Activities

FPT_TUD_EXT.4
TSS
The evaluator shall check the TSS section to confirm that it clearly and thoroughly describes how the secure local update functionality is implemented.

Guidance
The evaluator shall examine the AGD to ensure that it describes instructions for using the local update mechanism, and how to validate that the update was successful.

Tests
The evaluator shall perform the following tests:

- *Test FPT_TUD_EXT.4:1: The evaluator tests the secure local update by following the instructions provided in the operational guidance to update the platform firmware image. The update must succeed.*
- *Test FPT_TUD_EXT.4:2: The evaluator next tries to update the platform firmware image without first asserting physical presence. The update must fail or be not possible.*

B.7 Class: Trusted Path/Channels (FTP)

FTP_ITC_EXT.1 Trusted Channel Communication

The inclusion of this selection-based component depends upon selection in:

- ***FCS_CKM.2.1***,
- ***FDP_ITC_EXT.1.1***,
- ***FMT_SMF.1.1***

FTP_ITC_EXT.1.1

The TSF shall use [**selection**:

- *TLS as conforming to the Functional Package for Transport Layer Security (TLS), version 2.1*
- *TLS/HTTPS as conforming to FCS_HTTPS_EXT.1*
- *IPsec as conforming to FCS_IPSEC_EXT.1*
- *SSH as conforming to the Functional Package for Secure Shell (SSH), version 2.0*

] protocols with [**selection, choose one of**:

- *X.509 certificate-based authentication of the remote peer*
- *non-certificate-based authentication of the remote peer*
- *no authentication of the remote peer*

] to provide a communication channel between itself and [**selection**:

- *audit servers (as required by FAU_STG.1.1 if selected)*
- *remote administrators (as required by FTP_TRP.1.1 if selected in FMT_MOF.1)*
- *[**assignment**: other capabilities]*
- *no other capabilities*

] that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

Application Note: This SFR is included in the ST if a trusted channel is used to offload audit data or if the platform is administered remotely. That is, if "*a trusted channel as specified in FTP_ITC_EXT.1*" is selected in FAU_STG.1.1, if "*physically protected channels as specified in FTP_ITP_EXT.1*" is selected in FDP_ITC_EXT.1.1, or if "*remotely*" is selected in Management Function 1 in FMT_SMF.1.1.

If the ST Author selects either "*TLS*" or "*HTTPS*," the TOE must be validated against the Functional Package for TLS. This PP does not mandate that a product implement TLS with mutual authentication, but if the product includes the capability to perform TLS with mutual authentication, then mutual authentication must be included within the TOE boundary.

If the ST Author selects "*SSH*," the TOE must conform to Functional Package for Secure Shell (SSH), version 2.0.

If the ST Author selects "*certificate-based authentication of the remote peer*," then the TOE must conform to Functional Package for X.509, version 1.0.

Claims from this package are only required to the extent that they are needed to support the functionality required by the trusted protocols that are claimed.

If the TSF implements a protocol that requires the validation of a certificate presented by an external entity, FIA_X509_EXT.1 and FIA_X509_EXT.2 will be claimed. FIA_TSM_EXT.1 may also be claimed if the TSF implements its own trust store. If the TSF implements a protocol that requires the presentation of any certificates to an external entity, FIA_XCU_EXT.2 will be claimed. FIA_X509_EXT.3 will also be claimed, along with any applicable dependencies, depending on how the certificates presented by the TOE are obtained.

"*No authentication of the remote peer*" should be selected only if the TOE is acting as a server in a non-mutual authentication configuration.

.

Evaluation Activities

FTP_ITC_EXT.1
TSS
The evaluator will review the TSS to determine that it lists all trusted channels the TOE uses for remote communications, including both the external IT entities and remote users that use the channel as well as the protocol that is used for each.

Guidance
The evaluator shall confirm that the AGD contains instructions for establishing connections to external IT entities and remote users.

Tests
The evaluator will configure the TOE to communicate with each external IT entity and type of remote user identified in the TSS. The evaluator will monitor network traffic while the VS performs communication with each of these destinations. The evaluator will ensure that for each session a trusted channel was established in conformance with the protocols identified in the selection.

FTP_ITE_EXT.1 Encrypted Data Communications

The inclusion of this selection-based component depends upon selection in:

- ***FDP_ITC_EXT.1.1***

FTP_ITE_EXT.1.1

The TSF shall encrypt data for transfer between the TOE and [**assignment**: *list of entities external to the TOE*] using a cryptographic algorithm and key size as specified in FCS_COP.1/SKC or FCS_COP.1/AEAD, and using [**selection**:

- *Pre-shared keys*
- *Keys established according to FCS_CKM.2*
- *Keys exchanged using a physically protected communication mechanism conformant with FTP_ITP_EXT.1*

].

Application Note: This SFR must be claimed if "*encrypted data buffers as specified in FTP_ITE_EXT.1*" is selected in FDP_ITC_EXT.1.

This requirement applies to encrypted data communications between the TOE and external entities that do not use a physically protected mechanism conforming to FTP_ITP_EXT.1, or a cryptographically protected data channel as conforming to FTP_ITC_EXT.1. For example, if data is transferred through encrypted buffers (or blobs), then this requirement applies. This requirement would apply, for example, for communications implemented through a shared data buffer.

Evaluation Activities

FTP_ITE_EXT.1
TSS
The evaluator shall review the TSS to determine that it lists all encryption mechanisms the TOE uses for protected external communications, along with the types of communications protected using each mechanism.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
The evaluator shall configure the TOE to communicate with each external entity identified in the TSS. The evaluator shall initiate a transaction that will result in data being transferred to the TOE through the mechanism and other data returned to the initiating entity through the mechanism. The evaluator must verify that the data returned to the entity was encrypted using the documented mechanism when received.

FTP_ITP_EXT.1 Physically Protected Channel

The inclusion of this selection-based component depends upon selection in:
- ***FDP_ITC_EXT.1.1,***
- ***FTP_ITE_EXT.1.1***

FTP_ITP_EXT.1.1

The TSF shall provide a physically protected communication channel between itself and [**assignment**: *list of other IT entities within the same platform*].

Application Note: This SFR focuses on persistent internal communication channels and does not apply to communication channels intended for use with external media, ephemeral devices, or peripherals. This SFR must be claimed if "*physically protected channels as specified in FTP_ITP_EXT.1*" is selected in either FDP_ITC_EXT.1, or if "*Keys exchanged using a physically protected communication mechanism conformant with FTP_ITP_EXT.1*" is selected in FTP_ITE_EXT.1.1.

Evaluation Activities

FTP_ITP_EXT.1
TSS
The evaluator shall review the TSS to determine that it lists all mechanisms the TOE uses for physically protected external communications, along with the types of communications protected using each mechanism.

Guidance
There are no additional Guidance evaluation activities for this component.

Tests
There are no test activities for this component.

FTP_TRP.1 Trusted Path

The inclusion of this selection-based component depends upon selection in:

- ***FMT_SMF.1.1***

FTP_TRP.1.1

The TSF shall **use a trusted channel as specified in FTP_ITC_EXT.1 to** provide a **trusted** communication path between itself and [*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 [*modification, disclosure*].

FTP_TRP.1.2

The TSF shall permit [*remote **Administrators***] to initiate communication via the trusted path.

FTP_TRP.1.3

The TSF shall require the use of the trusted path for [*[all remote administration actions]*].

Application Note: This SFR is included in the ST if "*remotely*" is selected in Management Function 1 of FMT_SMF.1.1.

Protocols used to implement the remote administration trusted channel must be selected in FTP_ITC_EXT.1.

This requirement ensures that authorized remote Administrators initiate all communication with the TOE via a trusted path, and that all communications with the TOE by remote Administrators is performed over this path. The data passed in this trusted communication channel are encrypted as defined in the protocol chosen in the first selection in FTP_ITC_EXT.1.

Evaluation Activities

FTP_TRP.1

TSS

The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Guidance

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

Tests

The evaluator shall also perform the following tests:

- *Test FTP_TRP.1:1: The evaluator shall ensure that communications using each specified (in the AGD) remote administration method is tested during the course of the evaluation, setting up the connections as described in the AGD and ensuring that communication is successful.*
- *Test FTP_TRP.1:2: For each method of remote administration supported, the evaluator shall follow the AGD to ensure that there is no available interface that can be used by a remote user to establish remote administrative sessions without invoking the trusted path.*
- *Test FTP_TRP.1:3: The evaluator shall ensure, for each method of remote administration, the channel data is not sent in plaintext.*
- *Test FTP_TRP.1:4: The evaluator shall ensure, for each method of remote administration, modification of the channel data is detected by the TOE.*

Additional evaluation activities are associated with the specific protocols.

Appendix C - Extended Component Definitions

This appendix contains the definitions for all extended requirements specified in the PP.

C.1 Extended Components Table

All extended components specified in the PP are listed in this table:

Table 25: Extended Component Definitions

Functional Class	Functional Components
Class: Cryptographic Support (FCS)	FCS_CKM_EXT Cryptographic Key Management FCS_HTTPS_EXT HTTPS Protocol FCS_IPSEC_EXT IPsec Protocol FCS_STG_EXT Cryptographic Key Storage
Class: Identification and Authentication (FIA)	FIA_AFL_EXT Authentication Failure Handling FIA_PMG_EXT Password Management FIA_TRT_EXT Authentication Throttling FIA_UIA_EXT Administrator Identification and Authentication
Class: Protection of the TSF (FPT)	FPT_JTA_EXT Debug Port Access FPT_PPF_EXT Protection of Platform Firmware FPT_ROT_EXT Platform Integrity FPT_RVR_EXT Platform Firmware Recovery FPT_TUD_EXT Platform Firmware Update
Class: Security Management (FMT)	FMT_CFG_EXT Secure by Default
Class: Trusted Path/Channels (FTP)	FTP_ITC_EXT Trusted Channel Communications FTP_ITE_EXT Encrypted Data Communications FTP_ITP_EXT Physically Protected Channel
Class: User Data Protection (FDP)	FDP_ITC_EXT Key Import FDP_TEE_EXT Trusted Execution Environment

C.2 Extended Component Definitions

C.2.1 Class: Cryptographic Support (FCS)

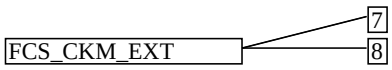
This PP defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.1 FCS_CKM_EXT Cryptographic Key Management

Family Behavior

This family defines requirements for management of cryptographic keys using mechanisms beyond what are specified in CC Part 2.

Component Leveling



FCS_CKM_EXT.7, Cryptographic Key Agreement, requires that cryptographic key agreement be performed in accordance with specified standards.

FCS_CKM_EXT.8, Password-Based Key Derivation, requires that password-based key derivation be performed in accordance with specified standards.

Management: FCS_CKM_EXT.7

There are no management functions foreseen.

Audit: FCS_CKM_EXT.7

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP, PP-Module, functional package or ST:

- minimal: Success and failure of the activity;
- basic: The object attribute(s), and object value(s) excluding any sensitive information.

FCS_CKM_EXT.7 Cryptographic Key Agreement

Hierarchical to: No other components.

Dependencies to:

[FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes],
[FCS_CKM.1 Cryptographic key generation, or
FCS_CKM.5 Cryptographic key derivation, or
FCS_CKM_EXT.8 Password-based key derivation] and
[FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation] and
FCS_CKM.6 Timing and event of cryptographic key destruction and
[FCS_COP.1 Cryptographic operation, or
no other dependencies].

FCS_CKM_EXT.7.1

The TSF shall derive shared cryptographic keys with input from multiple parties in accordance with specified cryptographic key agreement algorithms [**assignment**: *cryptographic algorithm*] and specified cryptographic parameters [**assignment**: *cryptographic parameters*] that meet the following: [**assignment**: *list of standards*].

Management: FCS_CKM_EXT.8

There are no management functions foreseen.

Audit: FCS_CKM_EXT.8

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP, PP-Module, functional package or ST:

- minimal: Success and failure of the activity;
- basic: The object attribute(s), and object value(s) excluding any sensitive information.

FCS_CKM_EXT.8 Password-Based Key Derivation

Hierarchical to: No other components.

Dependencies to:

[FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM_EXT.7 Cryptographic Key Agreement],
FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM_EXT.8.1

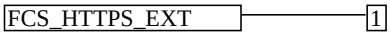
The TSF shall perform password-based key derivation functions in accordance with a specified cryptographic algorithm [HMAC-[**selection**: *SHA-384*, *SHA-512*], with iteration count of [**assignment**: *number of iterations*] using a randomly generated salt of length [**assignment**: *equal to or greater than 128*] and output cryptographic key sizes [**selection**: *256*, *384*, *512*] bits that meet the following standard: [NIST SP 800-132 (Section 5.3) [PBKDF2]].

C.2.1.2 FCS_HTTPS_EXT HTTPS Protocol

Family Behavior

This family defines requirements for protecting HTTP communications between the TOE and an external IT entity.

Component Leveling



FCS_HTTPS_EXT.1, HTTPS Protocol, defines requirements for the implementation of the HTTPS protocol.

Management: FCS_HTTPS_EXT.1

There are no management functions foreseen.

Audit: FCS_HTTPS_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure to establish an HTTPS session.
b. Establishment/termination of an HTTPS session.

FCS_HTTPS_EXT.1 HTTPS Protocol

Hierarchical to: No other components.

Dependencies to:
[FCS_TLSC_EXT.1 TLS Client Protocol, or
FCS_TLSC_EXT.2 TLS Client Protocol with Mutual Authentication, or
FCS_TLSS_EXT.1 TLS Server Protocol, or
FCS_TLSS_EXT.2 TLS Server Protocol with Mutual Authentication]

FCS_HTTPS_EXT.1.1

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

FCS_HTTPS_EXT.1.2

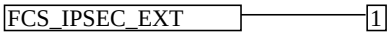
The TSF shall implement HTTPS using TLS.

C.2.1.3 FCS_IPSEC_EXT IPsec Protocol

Family Behavior

This family defines requirements for protecting communications using IPsec.

Component Leveling



FCS_IPSEC_EXT.1, IPsec Protocol, requires that IPsec be implemented as specified manner.

Management: FCS_IPSEC_EXT.1

The following actions could be considered for the management functions in FMT:

a. Managing the cryptographic functionality.

Audit: FCS_IPSEC_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure to establish an IPsec SA.
b. Establishment/Termination of an IPsec SA.

FCS_IPSEC_EXT.1 IPsec Protocol

Hierarchical to: No other components.

Dependencies to:
FCS_CKM.1 Cryptographic Key Generation
FCS_CKM.2 Cryptographic Key Establishment
FCS_COP.1 Cryptographic Operation
FCS_RBG.1 Random Bit Generation
FIA_X509_EXT.1 X.509 Certificate Validation

FCS_IPSEC_EXT.1.1

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

FCS_IPSEC_EXT.1.2

The TSF shall implement [**assignment**: *IPsec modes*].

FCS_IPSEC_EXT.1.3

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

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [**assignment**: *cryptographic algorithms*] together with a Secure Hash Algorithm (SHA)-based HMAC.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol: [**assignment**: *key exchange protocol*].

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the [**assignment**: *key exchange protocol*] protocol uses the cryptographic algorithms [**assignment**: *cryptographic algorithms*] and no other algorithm.

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [**assignment**: *key exchange protocol lifetime configuration rules*].

FCS_IPSEC_EXT.1.8

The TSF shall ensure that all IKE protocols implement DH groups [**assignment**: *DH Groups*].

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in gx mod p) using the random bit generator specified in FCS_RBG.1, and having a length of at least [**assignment**: *(one or more) number(s) of bits that is at least twice the "bits of security" value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*] bits.

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**assignment**: *(one or more) "bits of security" value(s) associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*].

FCS_IPSEC_EXT.1.11

The TSF shall ensure that all IKE protocols perform peer authentication using a [**assignment**: *IKE peer authentication algorithm*] that use X.509v3 certificates that conform to RFC 4945 and [**assignment**: *other IKE peer authentication mechanism*].

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the [**assignment**: *specific certificate reference identifier*] and [**assignment**: *other certificate reference identifier type*] contained in a certificate does not match the expected value(s) for the entity attempting to establish a connection.

FCS_IPSEC_EXT.1.13

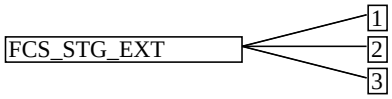
The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

C.2.1.4 FCS_STG_EXT Cryptographic Key Storage

Family Behavior

This family defines requirements for ensuring the protection of keys and secrets.

Component Leveling



FCS_STG_EXT.1, Protected Storage, requires the TSF to enforce protected storage for keys and secrets so that they cannot be accessed or destroyed without authorization.

FCS_STG_EXT.2, Key Storage Encryption, requires the TSF to ensure the confidentiality of stored data using a specified method.

FCS_STG_EXT.3, Key Integrity Protection, requires the TSF to ensure the integrity of stored data using a specified method.

Management: FCS_STG_EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to manage import and export keys/secrets to and from protected storage.

Audit: FCS_STG_EXT.1

There are no audit events foreseen.

FCS_STG_EXT.1 Protected Storage

Hierarchical to: No other components.

Dependencies to:
FCS_CKM.6 Timing and Event of Cryptographic Key Destruction

FCS_STG_EXT.1.1

The TSF shall provide [**assignment**: *protected storage type*] protected storage for asymmetric private keys and [**assignment**: *secrets to be stored*].

FCS_STG_EXT.1.2

The TSF shall support the capability of [**assignment**: *capability for acquiring keys*] upon request of [**assignment**: *entity requesting storage*].

FCS_STG_EXT.1.3

The TSF shall be capable of destroying keys/secrets in the protected storage upon request of [**assignment**: *authorized subject*].

Management: FCS_STG_EXT.2

There are no management functions foreseen.

Audit: FCS_STG_EXT.2

There are no audit events foreseen.

FCS_STG_EXT.2 Key Storage Encryption

Hierarchical to: No other components.

Dependencies to:
FCS_COP.1 Cryptographic Operation
FCS_STG_EXT.1 Protected Storage

FCS_STG_EXT.2.1

The TSF shall encrypt [**assignment**: *types of key material*] using [**assignment**: *cryptographic algorithm*].

Management: FCS_STG_EXT.3

There are no management functions foreseen.

Audit: FCS_STG_EXT.3

There are no audit events foreseen.

FCS_STG_EXT.3 Key Integrity Protection

Hierarchical to: No other components.

Dependencies to:
FCS_COP.1 Cryptographic Operation

FCS_STG_EXT.3.1

The TSF shall protect the integrity of any encrypted [**assignment**: *types of key material*] by using [**assignment**: *integrity protection mechanism*].

FCS_STG_EXT.3.2

The TSF shall verify the integrity of the [**selection**: *digital signature*, *MAC*] of the stored key prior to use of the key.

C.2.2 Class: Identification and Authentication (FIA)

This PP defines the following extended components as part of the FIA class originally defined by CC Part 2:

C.2.2.1 FIA_AFL_EXT Authentication Failure Handling

Family Behavior

This family defines requirements for the TOE's behavior when repeated failed attempts to gain authorization to access TSF data occur.

Component Leveling

FIA_AFL_EXT ——— 1

FIA_AFL_EXT.1, Authentication Failure Handling, requires the TSF to monitor authorization attempts, including counting and limiting the number of attempts at failed or passed authorizations. This extended component permits considerably more flexibility for dealing with multiple authentication mechanisms than FIA_AFL.

Management: FIA_AFL_EXT.1

The following actions could be considered for the management functions in FMT:

- Set authorization failure parameters

Audit: FIA_AFL_EXT.1

If FAU_GEN.1 is included in the ST, then the following audit events should be considered:

- Administrator authentication failures.

FIA_AFL_EXT.1 Authentication Failure Handling

Hierarchical to: No other components.

Dependencies to:
FCS_CKM.6 Timing and Event of Cryptographic Key Destruction
FIA_SMF.1 Specification of Management Functions

FIA_AFL_EXT.1.1

The TSF shall consider password and [**assignment**: *other authentication mechanisms*] as critical authentication mechanisms.

FIA_AFL_EXT.1.2

The TSF shall detect when a configurable positive non-zero integer within [**assignment**: *range of acceptable values for each authentication mechanism*] of [**selection, choose one of**: *unique*, *non-unique*] unsuccessful authentication attempts occur since the last successful authentication for each authentication mechanism.

FIA_AFL_EXT.1.3

The TSF shall maintain the number of unsuccessful authentication attempts that have occurred upon power off if the minimum boot time of the system is shorter than the lockout time specified in FIA_AFL_EXT.1.5.

FIA_AFL_EXT.1.4

When the defined number of unsuccessful authentication attempts has exceeded the maximum allowed for a given authentication mechanism, all future authentication attempts shall be limited to other available authentication mechanisms, unless the given mechanism is designated as a critical authentication mechanism.

FIA_AFL_EXT.1.5

When the defined number of unsuccessful authentication attempts for the last available authentication mechanism or a critical authentication mechanism has been surpassed, the TSF shall [**selection**:

- *perform a wipe of all protected data*
- *exclude the current Administrator from further authentication attempts*
- *exclude the current Administrator from further authentication attempts for [**assignment**: a period of time greater than zero seconds]*
- *exclude the current Administrator from further authentication attempts for [**assignment**: a period of time greater than the minimum boot time of the system]*

].

FIA_AFL_EXT.1.6

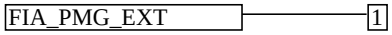
The TSF shall increment the number of unsuccessful authentication attempts prior to notifying the user that the authentication was unsuccessful.

C.2.2.2 FIA_PMG_EXT Password Management

Family Behavior

This family defines requirements for the composition of administrator passwords.

Component Leveling



FIA_PMG_EXT.1, Password Management, requires the TSF to support passwords with varying composition and length requirements.

Management: FIA_PMG_EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to configure password composition and length requirements for authorization of Administrators.
- Ability to manage authentication methods and change default authorization factors

Audit: FIA_PMG_EXT.1

There are no audit events foreseen.

FIA_PMG_EXT.1 Password Management

Hierarchical to: No other components.

Dependencies to: No other components.

FIA_PMG_EXT.1.1

The TSF shall support the following for the Password Authentication Factor:

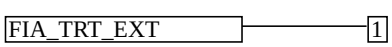
1. Passwords shall be able to be composed of any combination of [**assignment**: *characters sets*], numbers, and special characters: [**assignment**: *special characters*].
2. Password length of at least [**assignment**: *an integer greater than or equal to 14*] characters shall be supported.

C.2.2.3 FIA_TRT_EXT Authentication Throttling

Family Behavior

This family defines requirements for the limiting administrator authentication attempts.

Component Leveling



FIA_TRT_EXT.1, Authentication Throttling, requires that the TSF enforce a limit on authentication attempts.

Management: FIA_TRT_EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to configure an authentication throttling policy for the TOE.

Audit: FIA_TRT_EXT.1

The following should be considered for auditable events if FAU_GEN.1 is included in the ST:

- Authentication throttling is triggered.

FIA_TRT_EXT.1 Authentication Throttling

Hierarchical to: No other components.

Dependencies to:
FIA_UAU.5 Multiple Authentication Mechanisms

FIA_TRT_EXT.1.1

The TSF shall limit user authentication attempts by [**selection**: *preventing authentication via an external port*, *enforcing a delay between incorrect authentication attempts*] for all authentication mechanisms selected in FIA_UAU.5.1.

FIA_TRT_EXT.1.2

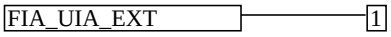
The minimum delay between incorrect authentication attempts shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

C.2.2.4 FIA_UIA_EXT Administrator Identification and Authentication

Family Behavior

This family defines requirements for ensuring that access to the TSF is not granted to unauthenticated subjects.

Component Leveling



FIA_UIA_EXT.1, Administrator Authentication, requires the TSF to ensure that all subjects attempting to perform TSF-mediated actions are authenticated prior to authorizing these actions to be performed.

Management: FIA_UIA_EXT.1

There are no management functions foreseen.

Audit: FIA_UIA_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. All use of the authentication mechanism.

FIA_UIA_EXT.1 Administrator Authentication

Hierarchical to: No other components.

Dependencies to:
FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UIA_EXT.1.1

The TSF shall require Administrators to be successfully authenticated using one of the methods in FIA_UAU.5 before allowing any TSF-mediated management function to be performed by that Administrator.

C.2.3 Class: Protection of the TSF (FPT)

This PP defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.1 FPT_JTA_EXT Debug Port Access

Family Behavior

This family defines requirements for access to debug ports during normal operation.

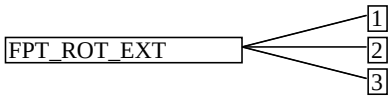
Component Leveling

C.2.3.2 FPT_ROT_EXT Platform Integrity

Family Behavior

This family defines requirements for platform firmware and hardware integrity.

Component Leveling



FPT_ROT_EXT.1, Platform Integrity Root, requires that the platform integrity be anchored in a root of trust.

FPT_ROT_EXT.2, Platform Integrity Extension, specifies how platform integrity is extended from the integrity root to other platform firmware.

FPT_ROT_EXT.3, Hardware component integrity, requires that the TOE support hardware supply chain integrity.

Management: FPT_ROT_EXT.1

There are no management functions foreseen.

Audit: FPT_ROT_EXT.1

There are no audit events foreseen.

FPT_ROT_EXT.1 Platform Integrity Root

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_ROT_EXT.1.1

The integrity of platform firmware shall be rooted in [**assignment**: *platform firmware root of trust*].

Management: FPT_ROT_EXT.2

The following actions could be considered for the management functions in FMT:

- Configuration of action to take on integrity failure.

Audit: FPT_ROT_EXT.2

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Failure of integrity verification.

FPT_ROT_EXT.2 Platform Integrity Extension

Hierarchical to: No other components.

Dependencies to: FPT_ROT_EXT.1 Platform Integrity Root

FPT_ROT_EXT.2.1

The integrity of all mutable platform firmware outside of the platform integrity root specified in FPT_ROT_EXT.1 shall be verified prior to execution or use through: [**assignment**: *method for extending the platform integrity root*].

FPT_ROT_EXT.2.2

The TOE shall take the following actions if an integrity check specified in FPT_ROT_EXT.2.1 fails: [**selection**:

- *Stop all execution, or*
- *Notify an [**selection**: Administrator, User] by [**selection**: generating an audit event, [**assignment**: other notification method(s)]], and [**selection**:*
  - *Stop all execution*
  - *Shut down, or*
  - *Initiate a recovery process as specified in FPT_RVR_EXT.1*
  - *Skip all instructions that failed the integrity check and continue execution*

  *][**selection**:*
  - *automatically*
  - *in accordance with Administrator-configurable policy*
  - *by express determination of an [**selection**: Administrator, User]*

  ]

].

Management: FPT_ROT_EXT.3

The following actions could be considered for the management functions in FMT:

- Configuration of action to take on integrity failure.

Audit: FPT_ROT_EXT.3

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Detection of attempted intrusion.

FPT_ROT_EXT.3 Hardware component integrity

Hierarchical to: No other components.

Dependencies to:
FPT_ROT_EXT.1 Platform Integrity Root

FPT_ROT_EXT.3.1

Outside of the integrity root specified in FPT_ROT_EXT.1, the integrity of [**assignment**: *critical platform hardware components*] shall be verified prior to execution or use through: [**assignment**: *method for ensuring integrity of platform hardware components*].

FPT_ROT_EXT.3.2

The TOE shall take the following actions if an integrity check specified in FPT_ROT_EXT.3.1 fails:

1. Halt,
2. Notify an [**selection**: *Administrator*, *User*] by [**assignment**: *notification method*], and
3. [**selection, choose one of**:
   - *Stop all execution and shut down*
   - *Continue execution without the integrity-compromised component*
   - *Continue execution*

   ]
   [**selection, choose one of**:
   - *in accordance with administrator-configurable policy*
   - *by express determination of an [**selection**: Administrator, User]*

   ]

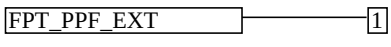
.

C.2.3.3 FPT_PPF_EXT Protection of Platform Firmware

Family Behavior

This family defines requirements for protecting platform firmware from unauthorized update.

Component Leveling



FPT_PPF_EXT.1, Protection of Platform Firmware and Critical Data, requires that the TSF prevent platform firmware from being modified outside of the update mechanisms defined in FPT_TUD_EXT.

Management: FPT_PPF_EXT.1

There are no management functions foreseen.

Audit: FPT_PPF_EXT.1

There are no audit events foreseen.

FPT_PPF_EXT.1 Protection of Platform Firmware and Critical Data

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_PPF_EXT.1.1

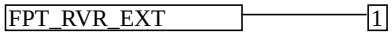
The TSF shall allow modification of platform firmware and critical data only through the update mechanisms described in FPT_TUD_EXT.1.

C.2.3.4 FPT_RVR_EXT Platform Firmware Recovery

Family Behavior

This family defines requirements for recovering from a firmware integrity failure.

Component Leveling



FPT_RVR_EXT.1, Platform Firmware Recovery, defines mechanisms for recovering from a platform firmware integrity failure.

Management: FPT_RVR_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of action to take on integrity failure.

Audit: FPT_RVR_EXT.1

There are no audit events foreseen.

FPT_RVR_EXT.1 Platform Firmware Recovery

Hierarchical to: No other components.

Dependencies to: FPT_TUD_EXT.4 Secure Local Update Mechanism

FPT_RVR_EXT.1.1

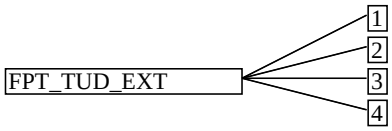
The TSF shall implement a mechanism for recovering from boot firmware failure consisting of [**assignment**: *recovery mechanism*].

C.2.3.5 FPT_TUD_EXT Platform Firmware Update

Family Behavior

This family defines requirements for updating platform firmware.

Component Leveling



FPT_TUD_EXT.1, TOE Firmware Update, requires that the TSF support update of platform firmware.

FPT_TUD_EXT.2, Platform Firmware Authenticated Update Mechanism, specifies the requirements for authenticated update of platform firmware.

FPT_TUD_EXT.3, Platform Firmware Delayed-Authentication Update Mechanism, specifies the requirements for delayed-authentication update of platform firmware.

FPT_TUD_EXT.4, Secure Local Platform Firmware Update Mechanism, specifies the requirements for secure local update of platform firmware.

Management: FPT_TUD_EXT.1

The following actions could be considered for the management functions in FMT:

- Initiation of the update process.

Audit: FPT_TUD_EXT.1

There are no audit events foreseen.

FPT_TUD_EXT.1 TOE Firmware Update

Hierarchical to: No other components.

Dependencies to:
FPT_TUD_EXT.2 Platform Firmware Authenticated Update Mechanism
FPT_TUD_EXT.3 Platform Firmware Delayed-Authentication Update Mechanism
FPT_TUD_EXT.4 Secure Local Platform Firmware Update Mechanism

FPT_TUD_EXT.1.1

The TSF shall implement [**assignment**: *update mechanism*].

Management: FPT_TUD_EXT.2

The following actions could be considered for the management functions in FMT:

- Configuration of action to take on an update failure.

Audit: FPT_TUD_EXT.2

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Failure of update authentication/integrity check/rollback
- Failure of update operation
- Success of update operation

FPT_TUD_EXT.2 Platform Firmware Authenticated Update Mechanism

Hierarchical to: No other components.

Dependencies to:
FCS_COP.1 Cryptographic Operations

FPT_TUD_EXT.2.1

The TSF shall authenticate the source of all platform firmware updates using a digital signature algorithm specified in FCS_COP.1 and using a key store that contains [**selection**: *the public key*, *hash value of the public key*].

FPT_TUD_EXT.2.2

The TSF shall allow installation of updates only if the digital signature has been successfully verified as specified in FCS_COP.1 and [**assignment**: *additional constraints on updates*].

FPT_TUD_EXT.2.3

The TSF shall include a platform firmware version identifier that is accessible by the update mechanism and includes information that enables the update mechanism to determine the relative order of updates.

FPT_TUD_EXT.2.4

The TSF shall provide an observable indication of the success or failure of the update operation.

FPT_TUD_EXT.2.5

The TOE shall take the following actions if a platform firmware integrity, authenticity, or rollback-prevention check fails, or a platform firmware update fails for any other reason:

- Do not install the update,
- Generate an audit event (optional),

and [**selection, choose one of**:

- *Continue execution*
- *Halt*
- *Stop all execution and shut down*
- *Initiate recovery as specified in FPT_RVR_EXT.1*

] [**selection, choose one of**:

- *automatically*
- *in accordance with Administrator-configurable policy*
- *by express determination of a User*

].

Management: FPT_TUD_EXT.3

The following actions could be considered for the management functions in FMT:

- Configuration of action to take on an update failure.

Audit: FPT_TUD_EXT.3

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Failure of update authentication/integrity check/rollback
- Failure of update operation
- Success of update operation

FPT_TUD_EXT.3 Platform Firmware Delayed-Authentication Update Mechanism

Hierarchical to: No other components.

Dependencies to: FPT_ROT_EXT.2 Platform Integrity Extension

FPT_TUD_EXT.3.1

The TSF shall allow execution or use of platform firmware updates only if new platform firmware is integrity- and authenticity-checked using the mechanism described in FPT_ROT_EXT.2 prior to its execution or use, and [**assignment**: *additional constraints on update*].

FPT_TUD_EXT.3.2

The TSF shall include an observable platform firmware version identifier that is accessible by the update mechanism and includes information that enables the update mechanism to determine the relative order of updates.

FPT_TUD_EXT.3.3

The TSF shall provide an observable indication of the success or failure of the update operation.

FPT_TUD_EXT.3.4

The TOE shall take the following actions if a platform firmware update integrity, authentication, or rollback-prevention check fails, or a platform firmware update fails for any other reason:

- Do not install the update,
- Generate an audit event (optional),

and [**selection, choose one of**:

- *Continue execution*
- *Halt*
- *Stop all execution and shut down*
- *Initiate recovery as specified in FPT_RVR_EXT.1*

]
[**selection, choose one of**:

- *automatically*

- *in accordance with administrator-configurable policy*
- *by express determination of a User*

].

Management: FPT_TUD_EXT.4

There are no management functions foreseen.

Audit: FPT_TUD_EXT.4

There are no audit events foreseen.

FPT_TUD_EXT.4 Secure Local Platform Firmware Update Mechanism

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_TUD_EXT.4.1

The TSF shall provide a secure local update mechanism that requires an assertion of physical access to the TOE before installation of an update.

FPT_TUD_EXT.4.2

A user shall assert physical presence to the TSF through: [**assignment**: *method for asserting physical presence*].

FPT_TUD_EXT.4.3

The TSF shall include a platform firmware version identifier that is accessible by the update mechanism or to the user who asserts physical presence.

FPT_TUD_EXT.4.4

The TSF shall provide an observable indication of the success or failure of the update operation.

C.2.4 Class: Security Management (FMT)

This PP defines the following extended components as part of the FMT class originally defined by CC Part 2:

C.2.4.1 FMT_CFG_EXT Secure by Default

Family Behavior

This family defines requirements for secure by default configuration of the TOE.

Component Leveling

FMT_CFG_EXT ——— 1

FMT_CFG_EXT.1, Secure by Default Configuration, requires that default Administrator credentials be changed immediately after first use.

Management: FMT_CFG_EXT.1

There are no management functions foreseen.

Audit: FMT_CFG_EXT.1

There are no audit events foreseen.

FMT_CFG_EXT.1 Secure by Default Configuration

Hierarchical to: No other components.

Dependencies to:

FIA_UAU.1 Timing of Authentication
FMT_SMR.1 Security Roles

FMT_CFG_EXT.1.1

The TSF shall enforce that Administrator credentials be changed immediately after first use when configured with default Administrator credentials or with no Administrator credentials.

C.2.5 Class: Trusted Path/Channels (FTP)

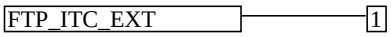
This PP defines the following extended components as part of the FTP class originally defined by CC Part 2:

C.2.5.1 FTP_ITC_EXT Trusted Channel Communications

Family Behavior

This family defines requirements for protection of data in transit between the TOE and its operational environment.

Component Leveling



FTP_ITC_EXT.1, Trusted Channel Communication, requires the TSF to implement one or more cryptographic protocols to secure connectivity between the TSF and various external entities.

Management: FTP_ITC_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure the cryptographic functionality.

Audit: FTP_ITC_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Initiation of the trusted channel.
b. Termination of the trusted channel.
c. Failures of the trusted path functions.

FTP_ITC_EXT.1 Trusted Channel Communication

Hierarchical to: No other components.

Dependencies to: No dependencies.

FTP_ITC_EXT.1.1

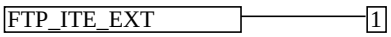
The TSF shall use [**assignment**: *trusted channel protocols*] protocols with [**assignment**: *authentication mechanism*] to provide a communication channel between itself and [**assignment**: *external IT entities*] that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

C.2.5.2 FTP_ITE_EXT Encrypted Data Communications

Family Behavior

This family defines requirements for encryption of TSF data that is transmitted to an external entity over an insecure channel.

Component Leveling



FTP_ITE_EXT.1, Encrypted Data Communications, requires the TSF to encrypt data in the specified manner using key data that is provided to an external entity in the specified manner.

Management: FTP_ITE_EXT.1

There are no management functions foreseen.

Audit: FTP_ITE_EXT.1

There are no audit events foreseen.

FTP_ITE_EXT.1 Encrypted Data Communications

Hierarchical to: No other components.

Dependencies to:
FCS_COP.1 Cryptographic Operation

FTP_ITE_EXT.1.1

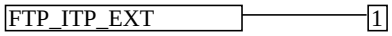
The TSF shall encrypt data for transfer between the TOE and [**assignment**: *list of entities external to the TOE*] using a cryptographic algorithm and key size as specified in FCS_COP.1, and using [**assignment**: *key establishment mechanism*].

C.2.5.3 FTP_ITP_EXT Physically Protected Channel

Family Behavior

This family defines requirements for use of physically protected communications mechanisms.

Component Leveling



FTP_ITP_EXT.1, Physically Protected Channel, requires the TSF to use a physically protected channel for transmission of data to an external entity.

Management: FTP_ITP_EXT.1

There are no management functions foreseen.

Audit: FTP_ITP_EXT.1

There are no audit events foreseen.

FTP_ITP_EXT.1 Physically Protected Channel

Hierarchical to: No other components.

Dependencies to: No dependencies.

FTP_ITP_EXT.1.1

The TSF shall provide a physically protected communication channel between itself and [**assignment**: *list of other IT entities within the same platform*].

C.2.6 Class: User Data Protection (FDP)

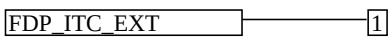
This PP defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.6.1 FDP_ITC_EXT Key Import

Family Behavior

This family defines requirements for importing cryptographic keys and credentials into the TOE.

Component Leveling



FDP_ITC_EXT.1, Key/Credential Import, requires the TSF to implement one or more means for importing keys and credentials into the TOE, which are not addressed by the FDP_ITC component.

Management: FDP_ITC_EXT.1

There are no management functions foreseen.

Audit: FDP_ITC_EXT.1

There are no audit events foreseen.

FDP_ITC_EXT.1 Key/Credential Import

Hierarchical to: No other components.

Dependencies to:
FCS_COP.1 Cryptographic Operation
FCS_STG_EXT.1 Key Storage Encryption
FCS_CKM.2 Key Distribution
FTP_ITE_EXT.1 Encrypted Data Communications
FTP_ITP_EXT.1 Physically Protected Channel

FDP_ITC_EXT.1.1

The TSF shall support importing keys/key material using [**assignment**: *import mechanism*].

FDP_ITC_EXT.1.2

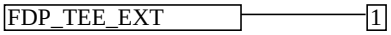
The TSF shall verify the integrity of imported keys/key material using [**assignment**: *integrity verification method*].

C.2.6.2 FDP_TEE_EXT Trusted Execution Environment

Family Behavior

This family defines requirements for Trusted Execution Environments implemented by the TOE for the use of tenant software.

Component Leveling



FDP_TEE_EXT.1, Trusted Execution Environment for Tenant Software, requires the TSF to implement a trusted execution environment for the use of tenant software.

Management: FDP_TEE_EXT.1

There are no management functions foreseen.

Audit: FDP_TEE_EXT.1

There are no audit events foreseen.

FDP_TEE_EXT.1 Trusted Execution Environment for Tenant Software

Hierarchical to: No other components.

Dependencies to: No dependencies.

FDP_TEE_EXT.1.1

The TSF shall implement a trusted execution environment that conforms to the following standard: [**assignment**: *Trusted Execution Environment standard*] and make this TEE available to tenant software.

Appendix D - Implicitly Satisfied Requirements

This appendix lists requirements that should be considered satisfied by products successfully evaluated against this PP. These requirements are not featured explicitly as SFRs and should not be included in the ST. They are not included as standalone SFRs because it would increase the time, cost, and complexity of evaluation. This approach is permitted by [CC] Part 1, 8.3 Dependencies between components.

This information benefits systems engineering activities which call for inclusion of particular security controls. Evaluation against the PP provides evidence that these controls are present and have been evaluated.

Table 26: Implicitly Satisfied Requirements

Requirement	Rationale for Satisfaction
FIA_UAU.1 – Timing of Authentication	FMT_CFG_EXT.1 has a dependency on FIA_UAU.1 because it cannot exist unless the TOE supports an authentication mechanism.

Appendix E - Entropy Documentation and Assessment

E.1 Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the random comes from, where it is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

E.2 Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how to ensure that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

E.3 Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

E.4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

Appendix F - Equivalency Guidelines

F.1 Introduction

The purpose of equivalence in PP-based evaluations is to find a balance between evaluation rigor and commercial practicability--to ensure that evaluations meet customer expectations while recognizing that there is little to be gained from requiring that every variation in a product or platform be fully tested. Generally, if a product is found to be compliant with a PP on a particular platform, then all equivalent products on equivalent platforms are also considered to be compliant with the PP. In this case, since the GPCP is itself a platform, only equivalent GPCP products are considered in the analysis.

A vendor can make a claim of equivalence if the vendor believes that a particular instance of their product implements PP-specified security functionality in a way equivalent to the implementation of the same functionality on another instance of their product on which the functionality was tested. The product instances can differ in version number or feature level (model). Equivalence can be used to reduce the testing required across claimed evaluated configurations. It can also be used during Assurance Maintenance to reduce testing needed to add more evaluated configurations to a certification.

These equivalency guidelines do not replace Assurance Maintenance requirements or NIAP Policy #5 requirements for CAVP certificates. Nor may equivalence be used to leverage evaluations with expired certifications.

This appendix provides guidance for determining whether products are equivalent for purposes of evaluation against the GPCPPP. This guidance differs from that provided in other PPs in that a GPCP is itself a platform, and thus the distinction between product and platform is somewhat blurred. This equivalency analysis is adjusted to reflect this.

For a GPCP, equivalence has two aspects:

1. ***Product Equivalence:*** To be considered equivalent, GPCPs must be produced by the same vendor and support the same tenant software.
2. ***Technical Equivalence:*** GPCPs may be considered equivalent if there are no differences between them with respect to their implementations of PP-specified security functionality.

The equivalency determination is made in accordance with these guidelines by the validator and scheme using information provided by the evaluator/vendor.

F.2 Approach to Equivalency Analysis

There are two scenarios for performing equivalency analysis. One is when a product has been certified and the vendor wants to show that a later product should be considered certified due to equivalence with the earlier product. The other is when multiple product variants are going though evaluation together and the vendor would like to reduce the amount of testing that must be done. The basic rules for determining equivalence are the same in both cases. But there is one additional consideration that applies to equivalence with previously certified products. That is, the product with which equivalence is being claimed must have a valid certification in accordance with scheme rules and the Assurance Maintenance process must be followed. If a product's certification has expired, then equivalence cannot be claimed with that product.

When performing equivalency analysis for a GPCP, the evaluator/vendor should first use the factors and guidelines for Product Equivalence to determine the set of products to be further considered.

Each non-equivalent product for which compliance is claimed must be fully tested.

"Differences in PP-Specified Security Functionality" Defined
PP-specified security functionality implemented by the TOE that differs in actual implementation between versions or product models break equivalence for that functionality. Likewise, the TOE invokes PP-specified security functionality differently in different versions or models of the TOE, then equivalence is broken for that functionality.

F.3 Specific Guidance for Determining Product Equivalence

Product Equivalence attempts to determine whether different feature levels or versions of the same product are equivalent for purposes of PP testing. For example, if a product has a "basic" edition and an "enterprise" edition, is it necessary to test both models? Or does testing one model provide sufficient confidence that both models are compliant?

Table 27: Factors for Determining Product Equivalence

Factor	Same/Different	Guidance

Product Type	Different	Products in different product classes are not equivalent. Servers and EUDs are not equivalent.
Product Vendors	Different	Products manufactured by different vendors are not equivalent.
PP-Specified Functionality	Same	If differences between products affect only non-PP-specified functionality, then the models are equivalent.
	Different	If PP-specified security functionality is affected by the differences between products, then the products are not equivalent and must be tested separately. It is necessary to test only the functionality affected by the differences. If only differences are tested, then the differences must be enumerated, and for each difference the Vendor must provide an explanation of why each difference does or does not affect PP-specified functionality. If the products are fully tested separately, then there is no need to document the differences.

F.4 Technical Equivalence

Platform equivalence is based primarily on processor architecture and instruction sets.

Technical equivalence is based primarily on processor architecture, instruction sets, and firmware versions. It is determined on a per-SFR basis.

Platforms with different processor architectures and instruction sets are not equivalent. Processors with the same architecture that have instruction sets that are subsets or supersets of each other are not disqualified from being equivalent. If PP-specified security functionality takes the same code paths when executing on different processors of the same family, then the processors can be considered equivalent with respect to that functionality.

For example, if for some PP-specified security functionality, one code path is followed on platforms that support the AES-NI instruction and another on platforms that do not, then those two platforms are not equivalent with respect to that functionality. But if the same path is followed whether or not the platform supports AES-NI, then the platforms are equivalent with respect to that functionality.

Platforms that run the same versions of the same firmware are considered equivalent with respect to any PP-specified security functionality implemented by that firmware. If firmware versions are different, then more in-depth analysis is required to determine whether the security functionality is implemented equivalently.

The platforms are equivalent if they are equivalent with respect to all PP-specified security functionality.

Table 28: Factors for Determining Technical Equivalence

Factor	Same/Different/None	Guidance
Processor Vendors	Different	Functionality implemented through processors manufactured by different vendors is not equivalent.
Processor/Chipset Architecture	Different	Functionality implemented through processors with different processor and chipset architectures are not equivalent.
Firmware Versions	Same	Functionality implemented through equivalent processors by the same version of firmware is considered equivalent.
PP-Specified Functionality	Same	For PP-specified security functionality implemented through equivalent processors and different firmware versions, the platforms are equivalent with respect to the functionality if execution of the functionality follows the same code paths on both platforms.
PP-Specified Functionality	Different	For PP-specified security functionality implemented through equivalent processors and different firmware versions, the platforms are not equivalent with respect to the functionality if execution of the functionality follows different code paths on both platforms.

F.5 Level of Specificity for Tested and Claimed Equivalent Configurations

In order to make equivalency determinations, the vendor and evaluator must agree on the equivalency claims. They must then provide the scheme with sufficient information about the TOE instances and platforms that were evaluated, and the TOE instances and platforms that are claimed to be equivalent.

The ST must describe all configurations evaluated down to processor manufacturer, model number, and microarchitecture version.

Appendix G - Use Case Templates

G.1 Server-Class Platform, Physically Secure Environment

The configuration for *[USE CASE 1] Server-Class Platform, Physically Secure Environment* modifies the base requirements as follows:

- Include FAU_GEN.1 in the ST
- Include FAU_SAR.1 in the ST
- Include FAU_STG.1 in the ST
- Include FAU_STG.2 in the ST
- Include FAU_STG.5 in the ST

G.2 Server-Class Platform, Enhanced Security Requirements

The configuration for *[USE CASE 2] Server-Class Platform, Enhanced Security Requirements* modifies the base requirements as follows:

- Include FAU_GEN.1 in the ST
- Include FAU_SAR.1 in the ST
- Include FAU_STG.1 in the ST
- Include FAU_STG.2 in the ST
- Include FAU_STG.5 in the ST
- Include FPT_PHP.2 in the ST
- Include FPT_PHP.3 in the ST

G.3 Portable Clients (laptops, tablets), Enhanced Security Requirements

The configuration for *[USE CASE 3] Portable Clients (laptops, tablets), Enhanced Security Requirements* modifies the base requirements as follows:

- Include FPT_PHP.1 in the ST

G.4 CSfC EUD

The configuration for *[USE CASE 4] CSfC EUD* modifies the base requirements as follows:

- Include FAU_GEN.1 in the ST
- Include FAU_SAR.1 in the ST
- Include FAU_STG.1 in the ST
- Include FCS_STG_EXT.1 in the ST
- Include FCS_RBG.6 in the ST

G.5 Tactical EUD

The configuration for *[USE CASE 5] Tactical EUD* modifies the base requirements as follows:

- Include FPT_PHP.3 in the ST
- Include FIA_AFL_EXT.1 in the ST

G.6 Enterprise Desktop Clients

The configuration for *[USE CASE 6] Enterprise Desktop Clients* modifies the base requirements as follows:

- Include FAU_GEN.1 in the ST
- Include FAU_SAR.1 in the ST
- Include FAU_STG.1 in the ST
- Include FAU_STG.2 in the ST
- Include FAU_STG.5 in the ST

Appendix H - Acronyms

Table 29: Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
AK	Asymmetric Key
ANSI	American National Standards Institute
API	Application Programming Interface
BAF	Biometric Authentication Factor
Base-PP	Base Protection Profile
BMC	Baseboard Management Controller
BOM	Bill of Materials
CC	Common Criteria
CEM	Common Evaluation Methodology
CMAC	Cipher-based Message Authentication Code
CN	Common Names
cPP	Collaborative Protection Profile
CRL	Certificate Revocation List
CSfC	Commercial Solutions for Classified
CSP	Critical Security Parameters
DAR	Data-at-Rest
DH	Diffie-Hellman Key Exchange
DN	Distinguished Name
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
DTLS	Datagram Transport Layer Security
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
EP	Extended Package
EUD	End-User Device
FIPS	Federal Information Processing Standards
FP	Functional Package
FQDN	Fully Qualified Domain Name
GPCP	General-Purpose Computing Platform

HMAC	Hash-based Message Authentication Code
HTTPS	Hypertext Transfer Protocol Secure
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
ITSEF	Information Technology Security Evaluation Facility
JTAG	Joint Test Action Group
KDF	Key-Derivation Function
KMAC	KECCAK Message Authentication Code
MAC	Message Authentication Code
MC	Management Controller
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OE	Operational Environment
OEM	Original Equipment Manufacturer
OID	Object Identifier
OMTP	Open Mobile Terminal Platform
OS	Operating System
PBKDF	Password-based Key-Derivation Function
PKCS	Public Key Cryptography Standards
PP	Protection Profile
PP-Configuration	Protection Profile Configuration
PP-Module	Protection Profile Module
RBG	Random Bit Generator
RFC	Request for Comment
RNG	Random Number Generator
RoT	Root of Trust
SA	Security Association
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SK	Symmetric Key

SPD	Security Policy Database
SSH	Secure Shell
ST	Security Target
SWID	Software Identification
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification
USB	Universal Serial Bus
VPN	Virtual Private Network
VS	Virtualization System
XCCDF	eXtensible Configuration Checklist Description Format
XOR	Exclusive Or

Appendix I - Bibliography

Table 30: Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and general model, CCMB-2022-11-001, CC:2022, Revision 1, November 2022. • Part 2: Security functional requirements, CCMB-2022-11-002, CC:2022, Revision 1, November 2022. • Part 3: Security assurance requirements, CCMB-2022-11-003, CC:2022, Revision 1, November 2022. • Part 4: Framework for the specification of evaluation methods and activities, CCMB-2022-11-004, CC:2022, Revision 1, November 2022. • Part 5: Pre-defined packages of security requirements, CCMB-2022-11-005, CC:2022, Revision 1, November 2022.
[CEM]	Common Methodology for Information Technology Security Evaluation - • Evaluation methodology, CCMB-2022-11-006, CC:2022, Revision 1, November 2022.
[ERR]	Errata and Interpretation for CC:2022 (Release 1) and CEM:2022 (Release 1), Version 1.1, CCMB-2024-02-002, 22 July 2024.
[OMB]	Reporting Incidents Involving Personally Identifiable Information and Incorporating the Cost for Security in Agency Information Technology Investments, OMB M-06-19, July 12, 2006.