Protection Profile for General-Purpose Computing Platforms

Version: 1.0-DRAFT
2021-09-21
National Information Assurance Partnership

Revision History

Version	Date	Comment
0.1	2020-11-09	Started
0.8	2021-06-11	First Draft of the First Draft
0.9	2021-09-21	First Draft for Public Comments

Contents

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 Ctiteria 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:

- NIAP, *BIOS Update for PC Client Devices protection Profile*, Version 1.0, 12 Feb 2013
- NIST SP800-147 *BIOS Protection Guidelines*, April 2011
- NIST SP800-147B *BIOS Protection Guidelines for Servers*, August 2014
- NIST SP800-193 *Platform Firmware Resiliency Guidelines*

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

- NIST SP800-155 (Draft) *BIOS Integrity Measurement Guidelines (Draft)*, December 2011
- 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.
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-Configuration)	A comprehensive set of security requirements for a product type that consists of at least one Base-PP and at least one PP-Module.
Protection Profile Module (PP-Module)	An implementation-independent statement of security needs for a TOE type complementary to one or more Base Protection Profiles.
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.
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.
Target of Evaluation (TOE)	The product under evaluation.

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.
Application (app)	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.
Cipher-based Message Authentication Code (CMAC)	A mode of AES that provides authentication, but not confidentiality.
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.
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 writes OS software. For the purposes of this document, vendors and developers are the same.
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 Operating Systems also lack the real-time constraint that defines Real Time Operating Systems (RTOS). RTOSes typically power routers, switches, and embedded devices.
General-Purpose Computing Platform (GPCP)	A physical computing platform designed to support general-purpose operating systems, virtualiation systems, and applications.
KECCAK Message Authentication Code (KMAC)	A variable-length keyed hash function described in NIST SP800-185.
Management Controller (MC)	Or Baseboard Management Controller. A small computer generally found on Server motherboards that performs management tasks on behalf of an Administrator.
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.
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. Sensitive data shall be identified in the OS's TSS by the ST author.
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.
User	In the context of a GPCP, a User is a person who is physically present and operating the platform. Users do not need to be authenticated by the platform to use the platform, but may authenticate to tenant software such as on Operating System.
Virtual Machine (VM)	A Virtual Machine is a virtualized hardware environment in which an operating system may execute.
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. For the purposes of this document, the VS consists of a Virtual Machine Manager (VMM), Virtual Machine abstractions, a management subsystem, and other components.

1.3 Compliant Targets of Evaluation

A general-purpose computing platform is a hardware device that is capable of hosting more than one different operating system, virtualization system, or bare-metal application. Typical platform implementations include servers, PC clients, laptops, and tablets.

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.

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 plaform 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.

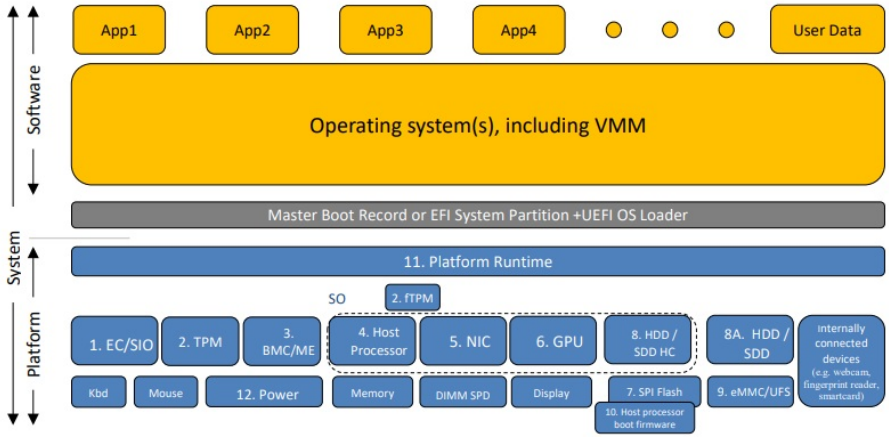


Figure 1: High-Level Architecture of a Generic Platform

Figure 1 (taken from NIST SP 800-193) shows a high-level system architecture for a 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.

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

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 for general-purpose computing platforms. The cases enumerated below add requirements to the baseline for GPCP due to additional threats or changes in assumptions about the operational environment. Use cases not listed below (e.g. consumer-grade desktop computers) need only be evaluated against the baseline requirements.

[USE CASE 1] Server-Class Platform, Basic

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 through a management controller who accesses the MC through a console or remotely.

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

For a the list of appropriate selections and acceptable assignment values for this configuration, see F.1 Server-Class Platform, Basic.

[USE CASE 2] Server-Class Platform, Enhanced

This use case adds physical protections to the base requirements for server-class hardware. Additional physical protections required because the platform us assumed to be minimally protected by the by the operational environment. This use case can also be invoked for servers in data centers where there are enhanced security requirements.

This use case adds requirements for audit and physical protections to the base mandatory SFRs. It removes the assumption that the TOE is physically protected by the OE.

For a the list of appropriate selections and acceptable assignment values for this configuration, see F.2 Server-Class Platform, Enhanced.

[USE CASE 3] Portable Clients (laptops, tablets), Basic

This use case includes the base requirements for portable clients or end-user devices.

This use case adds no requirements to the base mandatory SFRs.

For a the list of appropriate selections and acceptable assignment values for this configuration, see F.3 Portable Clients (laptops, tablets), Basic.

[USE CASE 4] Portable Clients (laptops, tablets), Enhanced

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

For a the list of appropriate selections and acceptable assignment values for this configuration, see F.4 Portable Clients (laptops, tablets), Enhanced.

[USE CASE 5] CSfC EUD

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

Although CSfC requires that users maintain physical control of EUDs at all times, this use case removes the assumption that the TOE is protected by the OE and adds requirements for audit and basic tamper detection and reporting.

For a the list of appropriate selections and acceptable assignment values for this configuration, see F.5 CSfC EUD.

[USE CASE 6] Tactical EUD

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

For a the list of appropriate selections and acceptable assignment values for this configuration, see F.6 Tactical EUD.

[USE CASE 7] 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 a the list of appropriate selections and acceptable assignment values for this configuration, see F.7 Enterprise Desktop clients.

[USE CASE 8] IoT Devices

IoT devices are field-located devices without human interfaces when in normal operation. In order to qualify for evaluation under this PP, the device must meet the basic criteria for a general-purpose platform, and not meet the requirements for a mobile device or network device.

For a the list of appropriate selections and acceptable assignment values for this configuration, see F.8 IoT Devices.

1.5 Roles

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

1. Tenant Software
2. Users (non-privileged users)
3. Administrators (privileged users)

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 is necessarily privileged on the platform.

Users are humans who interact with the platform through user interfaces. They generally have to authenticate thenselves to tenant software (e.g. an operating system), but generally not to the platform itself.

Administrators are humans who manage the platform through some kind of administrative interface. The interface may be local or remote to the platform.

In this context, Administrators manage the physical platform, not the OS (OS Admins would be classified as platform Users). Only Administrators need to be authenticated to the platform. For an EUD, this could generally 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 must authenticate themselves to the platform before the platform can allow them to perform administrative tasks.

Administrators are assumed to be acting in the best interests of the platform owner.

2 Conformance Claims

Conformance Statement

An ST must claim exact conformance to this PP, as defined in the CC and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs (dated May 2017).

CC Conformance Claims

This PP is conformant to Parts 2 (extended) and 3 (conformant) of Common Criteria Version 3.1, Revision 5.

PP Claim

This PP does not claim conformance to any Protection Profile.

Package Claim

This PP is Functional Package for Transport Layer Security (TLS), version 1.1 Conformant and Functional Package for Secure Shell (SSH), version 1.0 Conformant .

3 Security Problem Description

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. This threat is included only in the following use cases:

- Server-Class Platform, Enhanced
- Portable Clients (laptops, tablets), Enhanced
- CSfC EUD
- Tactical EUD
- IoT Devices

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 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 be physically protected in its operational environment and thus is not subject to physical attacks that could compromise its security or its ability to support the security of tenant workloads.

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

P.ENTERPRISE

If the OS is bound to a directory or management server, the configuration of the OS software must be capable of adhering to the enterprise security policies distributed by them.

4 Security Objectives

4.1 Security Objectives for the TOE

O.PHYSICAL_INTEGRITY
Protect the physical platform and interfaces from access by physical means such as probing, physical manipulation, fault-injection, side-channel analysis, environmental stress, or activating disabled features or pre-delivery services. This includes specification of:

- Requirements for tamper detection
- Requirements for disabling or protection of external debug interfaces
- Requirements for updateability
- Requirements for environmental shielding

O.ATTACK_DETECTION_AND_RESPONSE
Detected attempts to compromise the physical or logical integrity of the TOE must be indicated to a user or reported to an enterprise security authority. This includes specification of:

- Requirements for responses to particular detected events. (resilience, secure state, etc.)
- Requirements for basic auditing capabilities and protection of audit records.
- Requirements for secure transmission of audit records (if applicable)

O.MITIGATE_FUNDAMENTAL_FLAWS
The TOE must have a capability for mitigating or fixing fundamental flaws through update or some other technical or operational means. This includes specifications of:

- Requirements for updateabililty of TOE firmware.

O.PROTECTED_FIRMWARE
TOE firmware can be modified only through a non-bypassable trusted update process. This ensures the integrity of firmware both during the update process and while at rest. This includes specification of:

- Requirements for invocation of the trusted update process
- Requirements for non-bypassability of the update process
- Requirements for detection and reporting of attempts to modify TOE firmware outside of the trusted update process.

O.UPDATE_INTEGRITY
Updates to TOE firmware must be authorized, authenticated, and properly validated prior to installation. This includes specification of:

- Requirements for protection of updates
- Requirements for authentication of update packages
- Requirements for management of updates
- Requirements for detection and reporting of update events, whether authorized or not.

O.STRONG_CRYPTOGRAPHY
Cryptography must meet the standards required for protection of National Security Systems data (in accordance with CNSSP 15, "Use of Public Standards for Secure Information Sharing"). This includes specification of:

- Requirements for use and configuration of cryptographic operations.
- Requirements for key generation
- Requirements for random bit generation support.

O.SECURITY_FUNCTIONALITY_INTEGRITY
The TOE should be able not be able to operate with failed or degraded security functionality in such a way as might compromise the security or integrity of the TOE or of TOE data.

- Requirements for detection of degraded or failed security functionality.
- Requirements for protection of platform credentials against compromise (secret keys, passwords, etc.)
- Requirements for secure destruction of platform credentials (if applicable)
- Requirements for quality of credentials (password lengths and the like)

O.TENANT_SECURITY
The TOE should provide capabilities and security services to tenant software to help tenants help themselves. This includes specification of:

- Requirements for providing cryptographic support to tenants (random bit generation, crypto support instructions).
- Requirements for support for separation of tenant workloads
- Requirements for supporting secure storage of tenant credentials
- Requirements for supporting secure boot of a tenant operating system

O.TRUSTED_CHANNELS
Certain types of network communications with the TOE must be protected with guaranteed confidentiality, integrity, and authenticity. Such traffic includes communications with remote administrators, audit servers, update servers, and credential managers. This objective includes specification of:

- Requirements for the use of secure network protocols
- Requirements for the use of public key certificates
- Requirements for the authentication of endpoints

O.CONFIGURATION_INTEGRITY
The TOE should detect or prevent unauthorized users from being able to modify the system configuration. This includes:

- Requirements for authentication of Administrators before application of configuration modifications.
- Requirements for detection and reporting of attempts to modify the TOE configuration.

O.AUTHORIZED_ADMINISTRATOR
The TOE must ensure that Administrative actions can be taken only by authorized Administrators. This includes specification of:

- Requirements specifying actions allowable for the Administrator role
- Requirements for processes for authenticating Administrators
- Requirements for protection of Administrator credentials
- Requirements for setup and operation of a secure channel for remote administration
- Requirements for management of administrator sessions.

4.2 Security Objectives for the Operational Environment

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

OE.PLATFORM
The OS relies on being installed on trusted hardware.

OE.PROPER_USER

The user of the OS is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy. Standard user accounts are provisioned in accordance with the least privilege model. Users requiring higher levels of access should have a separate account dedicated for that use.

OE.PROPER_ADMIN
The administrator of the OS is not careless, willfully negligent or hostile, and administers the OS within compliance of the applied enterprise security policy.

4.3 Security Objectives Rationale

This section describes how the assumptions, threats, and organizational security policies map to the security objectives.

Table 1: Security Objectives Rationale

Threat, Assumption, or OSP	Security Objectives	Rationale
T.PHYSICAL	O.PHYSICAL_INTEGRITY	The threat T.PHYSICAL is coun O.PHYSICAL_INTEGRITY as thi detection or prevention of atter the physical platform.
	O.ATTACK_DETECTION_AND_RESPONSE	The threat T.PHYSICAL is coun O.ATTACK_DETECTION_AND_ objective supports detection an attempts to compromise the TO
T.SIDE_CHANNEL_LEAKAGE	O.MITIGATE_FUNDAMENTAL_FLAWS	The threat T.SIDE_CHANNEL_ countered by O.MITIGATE_FUNDAMENTAL_ objective supports the remedy through update or other techni means.
T.PERSISTENCE	O.PROTECTED_FIRMWARE	The threat T.PERSISTENCE is O.PROTECTED_FIRMWARE as supports maintenance of platfo integrity.
T.UPDATE_COMPROMISE	O.UPDATE_INTEGRITY	The threat T.UPDATE_COMPRO by O.UPDATE_INTEGRITY as th supports ensuring that platform are authentic and properly vali installation.
	O.STRONG_CRYPTOGRAPHY	The threat T.UPDATE_COMPRO by O.STRONG_CRYPTOGRAPH supports use of proven, standar cryptographic mechanisms for updates are authentic and main
T.SECURITY_FUNCTIONALITY_FAILURE	O.SECURITY_FUNCTIONALITY_INTEGRITY	The threat T.SECURITY_FUNCTIONALITY countered by O.SECURITY_FUNCTIONALITY objective supports integrity and of security functionality.
T.TENANT_BASED_ATTACK	O.TENANT_SECURITY	The threat T.TENANT-BASED_A countered by O.TENANT_SECU objective supports tenant-based mechanisms to prevent tenant
T.NETWORK_BASED_ATTACK	O.TRUSTED_CHANNELS	The threat T.REMOTE_ATTACK O.TRUSTED_CHANNELS as thi integrity and confidentiality of t
T.UNAUTHORIZED_RECONFIGURATION	O.CONFIGURATION_INTEGRITY	The threat T.UNAUTHORIZED_RECONFIG countered by O.CONFIGURATI this provides for integrity of pla configuration.
T.UNAUTHORIZED_PLATFORM_ADMINISTRATOR	O.AUTHORIZED_ADMINISTRATOR	The threat T.UNAUTHORIZED_PLATFORM is countered by O.AUTHORIZED_ADMINISTRA provides for authentication of p administrative users.
A.PHYSICAL_PROTECTION	OE.PHYSICAL_PROTECTION	The operational environment ob OE.PHYSICAL_PROTECTION is A.PHYSICAL_PROTECTION.
A.ROT_INTEGRITY	OE.ROT_INTEGRITY	The operational environment ob OE.SUPPLY_CHAIN is realized A.ROT_INTEGRITY.
A.TRUSTED_ADMIN	OE.TRUSTED_ADMIN	The operational environment ob OE.TRUSTED_ADMIN is realize A.TRUSTED_ADMIN.
A.MFR_ROT	OE.TRUSTED_ADMIN	The operational environment ob OE.TRUSTED_ADMIN is realize A.TRUSTED_ADMIN.
A.TRUSTED_DEVELOPMENT_AND_BUILD_PROCESSES	OE.TRUSTED_ADMIN	The operational environment ob OE.TRUSTED_ADMIN is realize A.TRUSTED_ADMIN.

A.SUPPLY_CHAIN_SECURITY	OE.TRUSTED_ADMIN	The operational environment ob OE.TRUSTED_ADMIN is realize A.TRUSTED_ADMIN.
A.CORRECT_INITIAL_CONFIGURATION	OE.TRUSTED_ADMIN	The operational environment ob OE.TRUSTED_ADMIN is realize A.TRUSTED_ADMIN.
A.TRUSTED_USERS	OE.TRUSTED_ADMIN	The operational environment ob OE.TRUSTED_ADMIN is realize A.TRUSTED_ADMIN.
A.REGULAR_UPDATES	OE.TRUSTED_ADMIN	The operational environment ob OE.TRUSTED_ADMIN is realize A.TRUSTED_ADMIN.
P.ENTERPRISE	O.MANAGEMENT	The organizational security poli is enforced through the objectiv O.MANAGEMENT as this objec the enterprise and user assert r the OS.

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 (including replacing an assignment with a more restrictive selection) 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	
FPT_JTA_EXT.1	No events specified	
FPT_PPF_EXT.1	No events specified	
FPT_ROT_EXT.1	No events specified	
FPT_ROT_EXT.2	**[selection:** *Failure of integrity verification.*, *None***]**	
FPT_STM.1	No events specified	
FPT_TUD_EXT.1	No events specified	

5.1.2 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 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 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.*

5.1.3 Class: Protection of the TSF (FPT)

FPT_JTA_EXT.1 JTAG/Debug Port Access

FPT_JTA_EXT.1.1

The TSF shall allow access to JTAG or other debug ports only to an authorized administrator through platform firmware or through assertion of physical presence.

Application Note: This requirement means that JTAG ports may not be accessible to tenant software. For use cases that include the threat T.PHYSICAL, FPT_JTA_EXT.2 should also be included in the ST.

Evaluation Activities

FPT_JTA_EXT.1
TSS
The evaluator shall examine the TSS to determine how access to the JTAG or debug ports is denied to tenant software.

Guidance
The evaluator shall examine the guidance to ensure that it describes how administrators assert physical presence to the TSF.
Tests
- ***Test 1:*** *The evaluator shall attempt to access the debug port without authenticating as an administrator. The attempt should fail.*
- ***Test 2:*** *[conditional] If the TOE supports an Administrator role, the evaluator shall autenticate as an Administrator and then attempt to access the debug port. The attempt should succeed.*

FPT_PPF_EXT.1 Protection of Platform Firmware and Critical Data

FPT_PPF_EXT.1.1

The TSF shall allow modification of platform firmware 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. Software portions of 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 evauator 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 operational guidance 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
- ***Test 1:*** *The evaluator shall attempt to overwrite or modify the platform firmware without invoking one of the update mechanisms specified in FPT_TUD_EXT.1 (e.g., using a modified Linux boot loader such as GRUB that attempts to write to the memory where platform firmware is stored). The test succeeds if the attempts to overwrite platform firmware fail. The evaluator shall attempt at least two such tests--one that attempts to overwrite the first platform firmware that executes after boot, and 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 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).

Selection of "a separate management controller..." implies the existance 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.

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 verified prior to execution or use through: [**selection**:
- *computation and verification of a hash by trusted code/data,*
- *verification of a digital signature by trusted code/data,*
- *measurement and verification by trusted code/data,*
- *measurement and verification by an on-platform dedicated security component,*

- *measurement and verification by an off-platform entity*

].

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

Otherwise, 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.

FPT_ROT_EXT.2.2

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

1. Halt,
2. Notify an [**selection**: *administrator, user*] by [**selection**: *generating an audit event, **[assignment**: other notification method(s)]*], and
3. [**selection**:
   - *Stop all execution and shut down,*
   - *Initiate a Recovery process as specified in FPT_RVR_EXT.1*

   ]

[**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, of if "in accordance with administrator-configurable policy" is selected, then all administrator authentication requirements shall 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.4, and FAU_STG_EXT.1 must be included in the ST.

If "*computation and verification of a hash by trusted code/data*" is selected then FCS_COP.1/Hash must be included in the ST.

If "*verification of a digital signature by trusted code/data*" is selected then FCS_COP.1/SigVer must be included in the ST.

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 FMT_MOF_EXT.1 and FMT_SMF_EXT.1 must be included in the ST.

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 TSF 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 operational guidance 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 guidance 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 evaluators 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 timestamp 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.

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 guidance documentation to ensure it instructs the administrator how to set the time or indicates any configuration steps required for the TSF to receive time data from an external source.
KMD
There are no KMD evaluation activities for this component.
Tests
The evaluator shall perform the following tests:

- ***Test 1:*** *[conditional]: 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 was set correctly.*
- ***Test 2:*** *[conditional]: If the TSF receives time data from some source outside the TOE, the evaluator shall use the guidance documentation to configure the external time source (if applicable). The evaluator shall observe that the time has been set to the expected value.*

FPT_TUD_EXT.1 TOE Firmware Update

FPT_TUD_EXT.1.1

The TSF shall [**selection**:

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

].

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 must select "make no provision for platform firmware update."

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

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 "implement an authenticated platform update mechanism..." and include FPT_TUD_EXT.2 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 "implement an unauthenticated platform update mechanism..." and include FPT_TUD_EXT.3 in the ST.

Evaluation Activities

FPT_TUD_EXT.1
TSS
If the ST author selects "make no provision 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 potentially destroying or damaging the TOE or some of its components.

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

If the ST author selects "implement an unauthenticated platform firmware update mechanism..." then this requirement is satisfied if FPT_TUD_EXT.3 is satisfied.

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

5.1.4 TOE Security Functional Requirements Rationale

The following rationale provides justification for each security objective for the TOE, showing that the SFRs are suitable to meet and achieve the security objectives:

Table 3: SFR Rationale

Objective	Addressed by	Rationale
O.PHYSICAL_INTEGRITY	FPT_PHP.1	Supports the objective through passive detection of physical tampering.
	FPT_PHP.2	Supports the objective through detection and reporting of physical tampering.
	FPT_PHP.3	Supports the objective through resistence to physical tampering.
	FPT_TUD_EXT.1	Supports the objective through requiring that a TOE be either updateable or immutable.
	FPT_TUD_EXT.2	Supports the objective through specifying an authenticated firmware update mechanism.
	FPT_TUD_EXT.3	Supports the objective through

		specifying a firmware update mechanism with delayed authentication.
	FPT_TUD_EXT.4	Supports the objective through specifying a secure local firmware update mechanism.
	FPT_JTA_EXT.1	Supports the objective through restricting access to debug ports.
	FPT_JTA_EXT.2	Supports the objective through requiring debug ports to be disabled.
O.ATTACK_DETECTION_AND_RESPONSE	FAU_GEN.1	Supports the objective by requiring reporting of security-related audit events.
	FAU_SAR.1	Supports the objective by requiring that audit events be readable by an Administrator.
	FAU_STG.1	Supports the objective by requiring that audit records be protected from unauthorized deletion.
	FAU_STG.4	Supports the objective by requiring that audit records be protected from automatic deletion.
	FAU_STG_EXT.1	Supports the objective by requiring that audit records be off-loaded to an external IT entity.
	FPT_STM.1	Supports the objective by ensuring that audit events are marked with reliable time stamps.
	FPT_PHP.1	Supports the objective through passive detection of physical tampering.
	FPT_PHP.3	Supports the objective through resistence to physical tampering.
	FPT_ROT_EXT.2	Supports the objective by indicating integrity failures in platform firmware.
O.MITIGATE_FUNDAMENTAL_FLAWS	FPT_TUD_EXT.1	Supports the objective through requiring that a TOE be either updateable or immutable.
	FPT_TUD_EXT.2	Supports the objective through specifying an authenticated firmware update mechanism.
	FPT_TUD_EXT.3	Supports the objective through specifying a firmware update mechanism with delayed authentication.
	FPT_TUD_EXT.4	Supports the objective through specifying a secure local firmware update mechanism.
O.PROTECTED_FIRMWARE	FPT_TUD_EXT.1	Supports the objective through requiring that a TOE be either updateable or immutable.
	FPT_TUD_EXT.2	Supports the objective through specifying an authenticated firmware update mechanism.
	FPT_TUD_EXT.3	Supports the objective through specifying a firmware update mechanism with delayed authentication.
	FPT_TUD_EXT.4	Supports the objective through specifying a secure local firmware update mechanism.
	FPT_ROT_EXT.1	Supports the objective by ensuring that platform integrity is rooted in a trust anchor.
	FPT_ROT_EXT.2	Supports the objective by detecting integrity failures in platform firmware.
	FPT_PPF_EXT.1	Supports the objective by requiring that platform firmware be modifiable only through the update process.
	FPT_RVR_EXT.1	Supports the objective by specifying a means for recovery from a boot firmware failure.
O.UPDATE_INTEGRITY	FPT_TUD_EXT.1	Supports the objective through requiring that a TOE be either updateable or immutable.

	FPT_TUD_EXT.2	Supports the objective through specifying an authenticated firmware update mechanism.
	FPT_TUD_EXT.3	Supports the objective through specifying a firmware update mechanism with delayed authentication.
	FPT_TUD_EXT.4	Supports the objective through specifying a secure local firmware update mechanism.
	FPT_ROT_EXT.2	Supports the objective by validating the integrity of platform firmware prior to execution.
	FPT_PPF_EXT.1	Supports the objective by requiring that platform firmware be modifiable only through the update process.
O.STRONG_CRYPTOGRAPHY	FCS_CKM.1/AK	Supports the objective by specifying the requirements for generating asymmetric keys.
	FCS_CKM.1/SK	Supports the objective by specifying the requirements for generating symmetric keys.
	FCS_CKM.1/KEK	Supports the objective by specifying the requirements for generating key encryption keys.
	FCS_CKM.2	Supports the objective by specifying the requirements for cryptographic key establishment.
	FCS_CKM_EXT.5	Supports the objective by specifying the requirements for cryptographic key derivation.
	FCS_COP.1/KeyedHash	Supports the objective by specifying the requirements for keyed hashes.
	FCS_COP.1/KAT	Supports the objective by specifying the requirements for key agreement and transport.
	FCS_COP.1/KeyEnc	Supports the objective by defining the methods for encryption and decryption of keys.
	FCS_COP.1/PBKDF	Supports the objective by specifying the requirements for password-based key derivation.
	FCS_COP.1/SigGen	Supports the objective by specifying the requirements for digital signature generation.
	FCS_COP.1/SigVer	Supports the objective by specifying the requirements for digital signature verification.
	FCS_COP.1/SKC	Supports the objective by specifying the requirements for symmetric-key cryptography.
	FCS_RBG_EXT.1	Supports the objective by specifying the requirements for random-bit generation services.
	FCS_SLT_EXT.1	Supports the objective by specifying the requirements for cryptographic salt generation.
	FCS_COP.1/Hash	Supports the objective by specifying the requirements for cryptographic hashing.
O.SECURITY_FUNCTIONALITY_INTEGRITY	FPT_PPF_EXT.1	Supports the objective by requiring that platform firmware be modifiable only through the update process.
	FCS_CKM.4	Supports the objective by specifying the requirements for credential and key destruction.
	FCS_CKM_EXT.4	Supports the objective by specifying the timing for credential and key destruction.
	FCS_STG_EXT.1	Supports the objective by specifying the types of credential storage supported by the TOE.
	FCS_STG_EXT.2	Supports the objective by specifying the types of material that must be encrypted for storage.
	FCS_STG_EXT.3	Supports the objective by specifying the encryption requirements for

		credential storage.
O.TENANT_SECURITY	FCS_ENT_EXT.1	Supports the objective by requiring that the TOE provide entropy to tenant software.
	FCS_STG_EXT.1	Supports the objective by specifying the types of credential storage supported by the TOE.
	FCS_STG_EXT.2	Supports the objective by specifying the types of material that must be encrypted for storage.
	FCS_STG_EXT.3	Supports the objective by specifying the encryption requirements for credential storage.
	FDP_TEE_EXT.1	Supports the objective by specifying the requirements for a trusted execution environment.
O.TRUSTED_CHANNELS	FCS_HTTPS_EXT.1	Supports the objective by specifying requirements for the HTTPS protocol.
	FCS_IPSEC_EXT.1	Supports the objective by specifying requirements for the IPSec protocol.
	FIA_X509_EXT.1	Supports the objective by specifying how X.509 certificate validation is performed.
	FIA_X509_EXT.2	Supports the objective by specifying how X.509 certificate authentication is performed.
	FTP_ITC_EXT.1	Supports the objective by specifying allowable trusted channel protocols.
	FTP_TRP.1	Supports the objective by specifying allowable uses for trusted channels.
O.CONFIGURATION_INTEGRITY	FMT_CFG_EXT.1	Supports the objective by requiring that default Administrator credentials be changed.
	FIA_UIA_EXT.1	Supports the objective by requiring Administrators be authenticated before making changes.
	FMT_MOF_EXT.1	Supports the objective by specifying that management functions be performed by Administrators.
	FMT_SMF_EXT.1	Supports the objective by specifying the management functions implemented by the TOE.
O.AUTHORIZED_ADMINISTRATOR	FMT_CFG_EXT.1	Supports the objective by requiring that default Administrator credentials be changed.
	FIA_TRT_EXT.1	Supports the objective by limiting the number of automated authentication attempts.
	FIA_AFL_EXT.1	Supports the objective by requiring that Administrators be authenticated.
	FIA_PMG_EXT.1	Supports the objective by specifying password complexity requirements.
	FIA_UAU.5	Supports the objective by specifying supported authentication mechanisms.
	FIA_UAU.7	Supports the objective by requiring that authentication factor feedback be suppressed.
	FIA_UIA_EXT.1	Supports the objective by requiring Administrators be authenticated before making changes.
	FIA_X509_EXT.1	Supports the objective by specifying how X.509 certificate validation is performed.
	FIA_X509_EXT.2	Supports the objective by specifying how X.509 certificate authentication is performed.
	FMT_MOF_EXT.1	Supports the objective by specifying that management functions be performed by Administrators.
	FMT_SMF_EXT.1	Supports the objective by specifying the management functions implemented by the TOE.

5.2 Security Assurance Requirements

The Security Objectives in were constructed to address threats identified in . 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 Assurance Activities to be performed are specified both in as well as in this section.
The general model for evaluation of OSs 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 Assurance Activities contained within , 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 Assurance Activities that are captured in also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the PP.

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 Assurance 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 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 in response to the functional requirements and the interfaces presented in the AGD documentation. No additional “functional specification” documentation is necessary to satisfy the assurance activities specified. The interfaces that need to be evaluated are characterized through the information needed to perform the assurance 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 AGD_OPE and AGD_PRE documentation. The developer may reference a website accessible to application developers and the evaluator. The assurance 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 assurance 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 assurance 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 assurance 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 assurance 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 assurance 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. 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 assurance 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. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

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

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 an identifier (such as a product name/version number) that specifically identifies 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 assurance activities are covered by the assurance 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 assurance 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., the developer, carriers(s)) and the steps that are performed (e.g., developer testing, carrier testing), including worst case time periods, before an update is made available to the public.

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. The evaluator will verify that this description addresses the entire application. 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, to include any third-party or carrier delays in deployment. 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 Assurance 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 assurance activities are covered by the assurance 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.

Application Note: The evaluator will test the OS on the most current fully patched version of the platform.

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. Public domain sources also include sites which provide free checking of files for viruses.

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 that may be included in the ST, but are not required in order to conform to this PP. However, applied modules, packages and/or use cases may refine specific requirements as mandatory.

The first type (A.1 Strictly Optional Requirements) are strictly optional requirements that are independent of the TOE implementing any function. If the TOE fulfills any of these requirements or supports a certain functionality, the vendor is encouraged to include the SFRs in the ST, but are not required in order to conform to this PP.

The second type (A.2 Objective Requirements) are objective requirements that describe security functionality not yet widely available in commercial technology. The requirements are not currently mandated in the body of this PP, but will be included in the baseline requirements in future versions of this PP. Adoption by vendors is encouraged and expected as soon as possible.

The third type (A.3 Implementation-Based Requirements) are dependent on the TOE implementing a particular function. If the TOE fulfills any of these requirements, the vendor must either add the related SFR or disable the functionality for the evaluated configuration.

A.1 Strictly Optional Requirements

A.1.1 Auditable Events for Strictly Optional Requirements

Table 4: Auditable Events for Optional Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_CKM.1/KEK	No events specified	
FCS_CKM.4	No events specified	
FCS_CKM_EXT.4	No events specified	
FCS_ENT_EXT.1	No events specified	
FCS_SLT_EXT.1	No events specified	
FDP_TEE_EXT.1	No events specified	
FIA_TRT_EXT.1	Authentication throttling triggered.	
FPT_JTA_EXT.2	No events specified	
FPT_PHP.1	Detection of intrusion.	
FPT_PHP.2	Detection of intrusion.	
FPT_PHP.3	Detection of attempted intrusion.	

A.1.2 Cryptographic Support (FCS)

FCS_CKM.1/KEK Cryptographic Key Generation (Key Encryption Key)

FCS_CKM.1.1/KEK

The TSF shall generate key encryption keys in accordance with a specified cryptographic key generation algorithm corresponding to [**selection**:

- *Asymmetric KEKs generated in accordance with FCS_CKM.1/AK identifier AK1,*
- *Symmetric KEKs generated in accordance with FCS_CKM.1/SK,*
- *Derived KEKs generated in accordance with FCS_CKM_EXT.5*

] and specified cryptographic key sizes [**assignment**: *cryptographic key sizes*] that meet the following: [**assignment**: *list of standards*].

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

KEKs protect KEKs and Symmetric Keys (SKs). DSCs should use key strengths commensurate with protecting the chosen symmetric encryption key strengths.

If Asymmetric KEKs generated in accordance with FCS_CKM.1/AK is selected, the selection-based SFR FCS_CKM.1/AK must be claimed by the TOE.

If Symmetric KEKs generated in accordance with FCS_CKM.1/SK is selected, the selection-based SFR FCS_CKM.1/SK must be claimed by the TOE.

If Derived KEKs generated in accordance with FCS_CKM_EXT.5 is selected, the selection-based SFR FCS_CKM_EXT.5 must be claimed by the TOE.

Evaluation Activities

FCS_CKM.1/KEK
TSS
The evaluator shall examine the key hierarchy section of the TSS to ensure that the formation of all KEKs is described and that the key sizes match that described by the ST author. The evaluator shall examine the key hierarchy section of the TSS to ensure that each KEK encrypts keys of equal or lesser security strength using one of the selected methods.

[conditional] If the KEK is generated according to an asymmetric key scheme, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_CKM.1/AK is invoked. The evaluator uses the description of the key generation functionality in FCS_CKM.1/AK or documentation available for the operational environment to determine that the key strength being requested is greater than or equal to 112 bits.

[conditional] If the KEK is generated according to a symmetric key scheme, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_CKM.1/SK is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1, or

the key derivation functionality in either FCS_CKM_EXT.5 or FCS_COP.1/PBKDF, depending on the key generation method claimed, to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.

[conditional] If the KEK is formed from derivation, the evaluator shall verify that the TSS describes the method of derivation and that this method is consistent with FCS_CKM_EXT.5.

Guidance
There are no guidance evaluation activities for this component.
KMD
The evaluator shall iterate through each of the methods selected by the ST and confirm that the KMD describes the applicable selected methods.
Tests
The evaluator shall iterate through each of the methods selected by the ST and perform all applicable tests from the selected methods.

FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM.4.1

The TSF shall destroy cryptographic keys and keying material 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**: a pseudo-random 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**:
  - *that employs a wear-leveling algorithm, the destruction shall be executed by a [**selection**:*
    - *single overwrite consisting of [**selection**: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [**assignment**: some value that does not contain any CSP]],*
    - *block erase*

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

    ]

  ]

that meets the following: [*no standard*].

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.

The platform must implement mechanisms to destroy cryptographic keys and key material contained in persistent storage when no longer needed. 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). 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.

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). Guidance documentation for the TOE requires users not to allow the TOE to leave the user's control while a session is active (and hence while the DEK is likely to be in plaintext in volatile memory).

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 collaborative Protection Profile for Dedicated Security Components the mappings of logical addresses to the erased memory locations are erased as well as the data itself.

Where different destruction methods are used for different data or different destruction situations then the different methods and the data/situations they apply to (e.g. different points in time, or power-loss situations) are described in the TSS (and the ST may use separate iterations of the SFR to aid clarity). The TSS includes a table describing all relevant keys and keying material (including authorization data) used in the implementation of the SFRs, stating the source of the data, all memory types in which the data is stored (covering storage both during and outside of a session, and both plaintext and non-plaintext forms of the data), and the applicable destruction method and time of destruction in each case.

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_EXT requirements, and not being any of the particular values listed as other selection options. The point of the phrase "does not contain any sensitive data" is to ensure that the overwritten data is carefully selected, and not taken from a general pool that might contain current or residual data (e.g. SDOs or intermediate key chain values) that itself requires confidentiality protection.

Evaluation Activities

FCS_CKM.4

TSS

The evaluator shall examine the TSS to ensure it lists all relevant keys and keying material (describing the source of the data, all memory types in which the data is stored (covering storage both during and outside of a session, and both plaintext and non-plaintext forms of the data)), 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 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 (e.g. that all keys in the key chain are accounted for). (Where keys are stored encrypted or wrapped under another key then this may need to be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions).

The evaluator shall check that the TSS identifies any configurations or circumstances that may not conform to the key destruction requirement (see further discussion in the AGD section below). Note that reference may be made to the AGD for description of the detail of such cases where destruction may be prevented or delayed.

Where the ST specifies the use of "a value that does not contain any sensitive data" to overwrite keys, the evaluator shall examine the TSS to ensure that it describes how that pattern is obtained and used, and that this justifies the claim that the pattern does not contain any sensitive data.

Guidance

The evaluator shall check that the guidance documentation for the TOE requires users to ensure that the TOE remains under the user's control while a session is active.

A TOE may be subject to situations that could prevent or delay data destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and KMD). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer, identifying any additional mitigation actions for the user (e.g. there might be some operation the user can invoke, or the user might be advised to retain control of the device for some particular time to maximise the probability that garbage collection will have occurred).

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may implement wear-levelling and garbage collection. This may result in additional copies of the data that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and guidance documentation).

Where TRIM is used then the TSS or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table.

KMD

The evaluator shall examine the KMD to verify that it identifies and describes the interfaces that are used to service commands to read/write memory. The evaluator shall examine the interface description for each different media type to ensure that the interface supports the selections made by the ST author.

45 The evaluator shall examine the KMD to ensure that all keys and keying material identified in the TSS and KMD have been accounted for.

46 Note that where selections include 'destruction of reference to the key directly followed by a request for garbage collection' (for volatile memory) then the evaluator shall examine the KMD to ensure that it explains the nature of the destruction of the reference, the request for garbage collection, and of the garbage collection process itself.

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 evaluator shall perform the following tests:

- ***Test 1:** Applied to each key or keying material held as plaintext in volatile memory and subject to destruction by overwrite by the TOE (whether or not the plaintext value 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 dump the SDO/SDE memory of the TOE into a binary file.*
  3. *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.
  4. *Cause the TOE to perform normal cryptographic processing with the key from Step #1.*
  5. *Cause the TOE to destroy the key.*
  6. *Cause the TOE to stop execution but not exit.*
  7. *Cause the TOE to dump the SDO/SDE memory of the TOE into a binary file.*
  8. *Search the content 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 2:*** *Applied to each key and keying material held in non-volatile memory and subject to destruction by overwrite 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 clear the key.*
  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.
- ***Test 3:*** *Applied to each key and keying material held in non-volatile memory and subject to destruction by overwrite by the TOE.*
  1. *Record memory 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 clear the key. Record the value to be used for the overwrite of the key.*
  4. *Examine the memory from Step #1 to ensure the appropriate pattern (recorded in Step #3) is used.*

  The test succeeds if correct pattern is found in the memory location. If the pattern is not found, then the test fails.

FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction Timing

FCS_CKM_EXT.4.1

The TSF shall destroy all keys and keying material when no longer needed.

Application Note: This SFR must be included in the ST if FCS_CKM.4 is included in the ST.

The platform will have mechanisms to destroy keys, including intermediate keys and key material, by using an approved method, FCS_CKM.4. Examples of keys include intermediate keys, leaf keys, encryption keys, signing keys, verification keys, authentication tokens, and submasks. The DSC will have mechanisms to destroy keys and key material contained in persistent storage when no longer needed. Based on their implementation, vendors will explain when certain keys are no longer needed. An example in which key is no longer necessary includes a wrapped key whose password has changed. However, there are instances when keys are allowed to remain in memory, for example, a device identification key.

Evaluation Activities

FCS_CKM_EXT.4
TSS
The evaluator shall verify the TSS provides a high-level description of what it means for keys and key material to be no longer needed and when this data should be expected to be destroyed.
Guidance
There are no guidance evaluation activities for this component.
KMD
The evaluator shall verify that the KMD includes a description of the areas where keys and key material reside and when this data is no longer needed.

The evaluator shall verify that the KMD includes a key lifecycle that includes a description where key materials reside, how the key materials are used, how it is determined that keys and key material are no longer needed, and how the data is destroyed once it is no longer needed. The evaluator shall also verify that all key destruction operations are performed in a manner specified by FCS_CKM.4.

Tests
There are no test evaluation activities for this component

FCS_ENT_EXT.1 Entropy for Tenant Software

FCS_ENT_EXT.1.1

The TSF shall provide one or more mechanisms to make entropy that meets FCS_RBG_EXT.1 available to tenant software.

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 sufficient entropy to any tenant that requires it. Every entropy source need not provide high-quality entropy, but tenant software must have a means of acquiring sufficient entropy.

A hardware noise source is a component that produces data that cannot be explained by a deterministic rule, due to its physical nature. In other words, a hardware based noise source generates sequences of random numbers from a physical process that cannot be predicted. For example, a sampled ring oscillator consists of an odd number of inverter gates chained into a loop, with an electrical pulse traveling from inverter to inverter around the loop. The inverters are not clocked, so the precise time required for a complete circuit around the loop varies slightly as various physical effects modify the small delay time at each inverter on the line to the next inverter. This variance results in an approximate natural frequency that contains drift and jitter over time. The output of the ring oscillator consists of the oscillating binary value sampled at a

constant rate from one of the inverters – a rate that is significantly slower than the oscillator's natural frequency.

Evaluation Activities

FCS_ENT_EXT.1
TSS
The evaluator shall verify that the TSS documents the entropy sources implemented by the TOE. It is not necessary to document all the platform features that can be used by tenant software to contribute to entropy, rather only those features expressly provided as entropy sources.
Guidance
The evaluator shall examine the AGD to ensure that it describes how to configure entropy sources (if applicable) and how tenant software can access the sources.
Tests
The evaluator shall perform the following tests:

- ***Test 1:*** *The evaluator shall invoke the entropy source(s) from tenant software. The evaluator shall verify that the tenant acquires values from the interface.*

FCS_SLT_EXT.1 Cryptographic Salt Generation

FCS_SLT_EXT.1.1

The TSF shall use salts and nonces generated by an RBG as specified in FCS_RBG_EXT.1.

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.

Evaluation Activities

FCS_SLT_EXT.1
TSS
The evaluator shall ensure the TSS describes how salts are generated using the RBG.
Guidance
There are no AGD evaluation activities for this component.
KMD
There are no KMD evaluation activities for this component.
Tests
The evaluator shall confirm by testing that the salts obtained in the cryptographic operations that use the salts are of the length specified in FCS_SLT_EXT.1, are obtained from the RBG, and are fresh on each invocation.

Note: in general these tests may be carried out as part of the tests of the relevant cryptographic operations.

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 all "software-based key storage" in FCS_STG_EXT.2.

Support for protected storage for all symmetric keys and persistent secrets will be required in future revisions.

FCS_STG_EXT.1.2

The TSF shall support the capability of [**selection**: *importing keys/secrets into the TOE*, *causing the TOE to generate keys/secrets*] upon request of [**selection**: *a client application*, *an administrator*].

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 operational guidance 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 operational guidance to ensure that it describes the process for destroying keys that have been imported or generated.

KMD
There are no KMD evaluation activities for this component.
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 and gathering of evidence.

- ***Test 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 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.
- ***Test 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 evalutor shall verify that the tenant software is able to cause the deletion of only keys that were created or imported on its behalf.*

A.1.3 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 guidance to ensure that it describes the steps required for tenant software to invoke the TEE.

A.1.4 Identification and Authentication (FIA)

FIA_TRT_EXT.1 Authentication Throttling

FIA_TRT_EXT.1.1

The TSF shall limit automated 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. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

Application Note: This SFT 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 this delay totals at least 500 milliseconds over 10 attempts for all authentication mechanisms selected in FIA_UAU.5.1.
Guidance
There are no guidance evaluation activities for this component.
Tests
There are no test evaluation activities for this component.

A.1.5 Class: Protection of the TSF (FPT)

FPT_JTA_EXT.2 JTAG/Debug Port Disablement

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

- ***Server-Class Platform, Enhanced***
- ***Portable Clients (laptops, tablets), Enhanced***
- ***CSfC EUD***
- ***Tactical EUD***

FPT_JTA_EXT.2.1

The TSF shall [**selection**: *disable access through hardware, control access by a signing key*] to JTAG or other debug interfaces.

Application Note: This requirement means that access to JTAG must be disabled either through hardware or restricted through the use of a signing key. This requirement should be included in the ST for use cases that include the

threat T.PHYSICAL.

Evaluation Activities

FPT_JTA_EXT.2
TSS
If "disable access through hardware" is selected:
The evaluator shall examine the TSS to determine the location of the JTAG ports on the TSF, to include the order of the ports (i.e. Data In, Data Out, Clock, etc.).

If "control access by a signing key" is selected:
The evaluator shall examine the TSS to determine how access to the JTAG is controlled by a signing key. The evaluator shall examine the TSS to determine when the JTAG can be accessed, i.e. what has the access to the signing key.

Guidance
There are no guidance evaluation activities for this component.
Tests
The following test requires the developer to provide access to a test platform that provides the evaluator with chip level access.

If "disable access through hardware" is selected:
The evaluator shall connect a packet analyzer to the JTAG ports. The evaluator shall query the JTAG port for its device ID and confirm that the device ID cannot be retrieved.

FPT_PHP.1 Passive detection of physical attack

This component must be included in the ST if any of the following use cases are selected:
- ***Portable Clients (laptops, tablets), Enhanced***
- ***CSfC EUD***

FPT_PHP.1.1
The TSF shall provide unambiguous detection of physical tampering that might compromise the TSF.

Application Note: This SFR should be included in the ST if the TOE implements the following use cases:

1. Portable Clients (laptops, tablets), Enhanced
2. CSfC EUD
3. IoT Devices

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.

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 the TOE will respond when physical tampering has been detected.

Guidance
The evaluator shall ensure that the Guidance describes how the TOE indicates to Users and Administrators that it has detected tampering.
KMD
There are no KMD evaluation activities for this component.
Tests
TODO: 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 be included in the ST if any of the following use cases are selected:
- ***Server-Class Platform, Enhanced***
- ***CSfC EUD***

FPT_PHP.2.1
The TSF shall provide unambiguous detection of physical tampering that might compromise the TSF.

Application Note: This SFR should be included in the ST if the TOE implements the following use cases:

1. Server-Class Platform, Enhanced
2. CSfC EUD

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.

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 Guidance describes how the TOE notifies Users or Administrators that it has detected tampering.
KMD
There are no KMD evaluation activities for this component.
Tests

- ***Test 1: TODO: 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 2: TODO: 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 be included in the ST if any of the following use cases are selected:

- ***Server-Class Platform, Enhanced***
- ***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 (TODO: **if supported**)
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
There are no AGD evaluation activities for this component.
KMD
There are no KMD evaluation activities for this component.
Tests

- ***Test 1: TODO: For each physical tampering scenario and device/element listed in FPT_PHP.3.1, the evaluator shall verify that tampering attempts shall result in a response from the TSF consistent with the operational guidance.***

A.2 Objective Requirements

A.2.1 Auditable Events for Objective Requirements

Table 5: Auditable Events for Objective Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FPT_ROT_EXT.3	Detection of attempted intrusion.	

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 upcoming 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, 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**:
   - *Stop all execution and shut down,*
   - *Continue execution without the integrity-compromised component,*
   - *Continue execution*

   ]

[**selection**:

- *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, of 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.4, and FAU_STG_EXT.1 must be included in the ST.

If "*in accordance with administrator-configurable policy*" is selected then FMT_MOF_EXT.1 and FMT_SMF_EXT.1 must be included in the ST.

Evaluation Activities

FPT_ROT_EXT.3

A.3 Implementation-Based Requirements

This PP does not define any Implementation-Based 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 6: Auditable Events for Selection-based Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	No events specified	
FAU_STG.1	No events specified	
FAU_STG.4	No events specified	
FAU_STG_EXT.1	On failure of logging function, capture record of failure and record upon restart of logging function.	
FCS_CKM.1/AK	No events specified	
FCS_CKM.1/SK	No events specified	
FCS_CKM.2	No events specified	
FCS_CKM_EXT.5	No events specified	
FCS_COP.1/Hash	No events specified	
FCS_COP.1/KeyedHash	No events specified	
FCS_COP.1/KAT	No events specified	
FCS_COP.1/KeyEnc	No events specified	
FCS_COP.1/PBKDF	No events specified	
FCS_COP.1/SigGen	No events specified	
FCS_COP.1/SigVer	No events specified	
FCS_COP.1/SKC	No events specified	
FCS_HTTPS_EXT.1	Failure to establish a HTTPS Session.	Reason for failure. Non-TOE endpoint of connection (IP address) for failures.
FCS_HTTPS_EXT.1	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).
FCS_IPSEC_EXT.1	Establishment/Termination of an IPsec SAA.	Non-TOE endpoint of connection (IP address).
FCS_RBG_EXT.1	Failure of the randomization process	
FIA_AFL_EXT.1	Failed attempt at Administrator authentication.	
FIA_PMG_EXT.1	No events specified	
FIA_UAU.5	No events specified	
FIA_UAU.7	No events specified	
FIA_UIA_EXT.1	Administrator authentication attempts.	Provided user identity, origin of the attempt (e.g. console, remote IP address).
FIA_UIA_EXT.1	All use of the identification and authentication mechanism.	Provided user identity, origin of the attempt (e.g. console, remote IP address).
FIA_UIA_EXT.1	**[selection:** *Start and end of administrator session., None*]	Start time and end time of administrator session.
FIA_X509_EXT.1	Failure to validate a certificate.	Reason for failure.
FIA_X509_EXT.2	No events specified	
FMT_MOF_EXT.1	No events specified	
FMT_SMF_EXT.1	No events specified	
FPT_RVR_EXT.1	No events specified	
FPT_TUD_EXT.2	**[selection:** *Failure of update authentication/integrity check/rollback, None*]	Version numbers of the current firmware and of the attempted update
FPT_TUD_EXT.2	**[selection:** *Failure of update operation, None*]	Version numbers of the current firmware and of the attempted update

FPT_TUD_EXT.2	**[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
FPT_TUD_EXT.3	**[selection:** *Failure of update operation, None***]**	Version numbers of the current firmware and of the attempted update
FPT_TUD_EXT.3	**[selection:** *Success of update operation, None***]**	Version numbers of the new and old firmware images.
FPT_TUD_EXT.4	No events specified	
FTP_ITC_EXT.1	Initiation of the trusted channel.	User ID and remote source (IP Address) if feasible.
FTP_ITC_EXT.1	Termination of the trusted channel.	User ID and remote source (IP Address) if feasible.
FTP_ITC_EXT.1	Failures of the trusted path functions.	User ID and remote source (IP Address) if feasible.
FTP_TRP.1	Initiation of the trusted channel.	User ID and remote source (IP Address) if feasible.
FTP_TRP.1	Termination of the trusted channel.	User ID and remote source (IP Address) if feasible.
FTP_TRP.1	Failures of the trusted path functions.	User ID and remote source (IP Address) if feasible.

B.2 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, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4.***
>
> ***This component must also be included in the ST if any of the following use cases are selected:***
>
> - ***Server-Class Platform, Basic***
> - ***Server-Class Platform, Enhanced***
> - ***CSfC EUD***
> - ***Enterprise Desktop clients***

FAU_GEN.1.1

The TSF shall be able to generate an audit record 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 4 for Strictly Optional requirements,*
   - *Specifically defined auditable event in Table 5 for Objective requirements,*
   - *Specifically defined auditable event in Table 6 for Selection-based requirements,*
   - *no additional auditable events,*
   - *Additional information for the listed in Table 7,*
   - *Additional information defined in the audit table for the*

   ]

FAU_GEN.1.2

The TSF shall record within each audit record 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 4 for Strictly Optional SFRs,***
   - ***Additional information defined in Table 5 for Objective SFRs,***
   - ***Additional information defined in Table 6 for Selection-Based SFRs,***
   - ***Additional information for the listed in Table 7,***
   - ***Additional information defined in the audit table for the ,***
   - ***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_PHP.1.2, FPT_ROT_EXT.2.2, FPT_TUD_EXT.2.5, or FPT_TUD_EXT.3.4; or if the TOE supports the Server (basic or enhanced), CSfC EUD, or Enterprise Desktop use cases.

Appropriate entries from Table 4, Table 5, and Table 6 should be included in the ST if the associated SFRs and selections are included.

The following table contains the events enumerated in the auditable events table for the TLS Functional Package. Inclusion of these events in the ST is subject to

selection above, inclusion of the corresponding SFRs in the ST, and support in the FP as represented by a selection in the table below.

Table 7: Auditable Events for the TLS Functional Package

FCS_TLSC_EXT.1	Failure to establish a session.	Reason for failure.
FCS_TLSC_EXT.1	Failure to verify presented identifier.	Presented identifier and reference identifier.
FCS_TLSC_EXT.1	Establishment/termination of a TLS session.	Non-TOE endpoint of connection.
FCS_TLSS_EXT.1	Failure to establish a session.	Reason for failure.
FCS_DTLSC_EXT.1	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate.
FCS_DTLSS_EXT.1	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate.

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. The evaluator shall check to make sure that every audit event type mandated by the PP-Configuration is described in the TSS.

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

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

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4.

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

- ***Server-Class Platform, Enhanced***
- ***CSfC EUD***
- ***Enterprise Desktop clients***

FAU_SAR.1.1

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

FAU_SAR.1.2

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

Evaluation Activities

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

Guidance
The evaluator shall review the operational guidance 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 Storage Protection

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4.

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

- ***Server-Class Platform, Enhanced***
- ***CSfC EUD***

- ***Enterprise Desktop clients***

FAU_STG.1.1

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

FAU_STG.1.2

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

Evaluation Activities

FAU_STG.1
TSS
The evaluator shall ensure that the TSS lists the location 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 AGD describes the steps necessary for an authorized administrator to delete audit records.

Tests
- ***Test 1:*** *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 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.4 Prevention of Audit Data Loss

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4.

This component must also be included in the ST if any of the following use cases are selected:
- ***Server-Class Platform, Enhanced***
- ***CSfC EUD***
- ***Enterprise Desktop clients***

FAU_STG.4.1

The TSF shall overwrite the oldest stored audit records if the audit trail is full.

Evaluation Activities

FAU_STG.4
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
There are no guidance evaluation activities for this component.
Tests
There are no test evaluation activities for this component.

FAU_STG_EXT.1 Off-Loading of Audit Data

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4.

This component must also be included in the ST if any of the following use cases are selected:
- ***Server-Class Platform, Enhanced***
- ***CSfC EUD***
- ***Enterprise Desktop clients***

FAU_STG_EXT.1.1

The TSF shall be able to tranfer generated audit data to an external IT entity using [**selection**:
- *a trusted channel as specified in FTP_ITC_EXT.1,*
- *removeable media requiring physical access to the platform*

].

Application Note: The ST author must select "trusted channel" and include 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 must select "removeable media" if the TOE supports offload of audit data using removeable media such as thumb drives or disks.

Evaluation Activities

FAU_STG_EXT.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 operational guidance 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.

If "removeable media" is selected, the evaluator shall ensure that the guidance describes the process for accessing audit data and copying it to media.

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:

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

If "removeable 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.

B.3 Cryptographic Support (FCS)

FCS_CKM.1/AK Cryptographic Key Generation (Asymmetric Keys)

The inclusion of this selection-based component depends upon selection in FCS_CKM.1.1/KEK.

FCS_CKM.1.1/AK

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

The rows of Table 8 provide the allowable values for completion of the assignments for FCS_CKM.1.1/AK.

Table 8: Supported Methods for Asymmetric Key Generation

Identifier/Key Type	Key Sizes	List of Standards
RSA	[**selection**: *2048 bit, 3072-bit*]	FIPS PUB 186-4 (Section B.3)
ECC-N	[**selection**: *256 (P-256), 384 (P-384), 521 (P-521)*]	FIPS PUB 186-4 (Section B.4 & D.1.2)
ECC-B	[**selection**: *256 (brainpoolP256r1), 384 (brainpoolP384r1), 512 (brainpoolP512r1)*]	RFC5639 (Section 3) (Brainpool Curves)
DSA	DSA Bit lengths of p and q respectively (L, N) [**selection**: *(1024, 160), (2048, 224), (2048, 256), (3027, 256)*]	FIPS 186-4 Appendix B.1
Curve25519	256 bits	RFC 7748

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 "Asymmetric KEKs..." is selected in FCS_CKM.1/KEK.

This requirement is included for the purposes of encryption and decryption operations only. To support ITE protected communications requirement for the transfer of encrypted data, this requirement mandates implementation compliance to FIPS 186-4 only. Implementations according to FIPS 186-2 or FIPS 186-3 will not be accepted.

This requirement must be claimed by the TOE if at least one of FCS_CKM.1 or FCS_CKM.1/KEK chooses a selection related to generation of asymmetric keys.

Evaluation Activities

FCS_CKM.1/AK
TSS
The evaluator shall examine the TSS to verify that it describes how the TOE generates an asymmetric key based on the methods selected from cPP Table 13: "Supported Methods for Asymmetric Key Generation". The evaluator shall examine the TSS to verify that it describes how the TOE invokes the methods selected in the ST from the same table. The evaluator shall examine the TSS to verify that it identifies the usage for each row identifier (key type, key size, and list of standards) selected in the ST.
Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key types for all uses identified in the ST.
KMD
If the TOE uses the generated key in a key chain/hierarchy then the evaluator shall confirm that the KMD describes:

- *If AK1 is selected, then the KMD describes which methods for generating p and q are used*
- *How the key is used as part of the key chain/hierarchy.*

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.

AK1: RSA Key Generation
The below tests are derived from The 186-4 RSA Validation System (RSA2VS), Updated 8 July 2014, Section 6.2, from the National Institute of Standards and Technology.

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

FIPS 186-4 Key Pair generation specifies 5 methods for generating the primes p and q.

These are:

1. *Random Primes:*
   - *Provable primes*
   - *Probable primes*
2. *Primes with Conditions:*
   - *Primes p1, p2, q1, q2, p and q shall all be provable primes.*
   - *Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes*
   - *Primes p1, p2, q1, q2, p and q shall all be probable primes.*

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

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

If the TOE generates Random Probable Primes 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 >= squareroot(2)*(2^(nlen/2 -1))*
- *q >= squareroot(2)*(2^(nlen/2 -1))*
- *2^(nlen/2) < d < LCM(p-1,q-1)*
- *e*d = 1 mod LCM(p-1,q-1)*

AK2 & AK3: ECC Key Generation with NIST and Brainpool Curves

These tests are derived from The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS), Updated 18 Mar 2014, Section 6.

ECC Key Generation Test

For each selected curve, and for each key pair generation method as described in FIPS 186-4, section B.4, 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_EXT.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'.

Public Key Validation (PKV) Test

For each supported curve, the evaluator shall generate 12 private/public key pairs using the key generation function of a known good implementation and modify six of the public key values so that they are incorrect, leaving six values unchanged (i.e., correct). To determine correctness, the evaluator shall submit the 12 key pairs to the public key validation (PKV) function of the TOE and shall confirm that the results correspond as expected to the modified and unmodified values.

AK4: DSA Key Generation using Finite-Field Cryptography (FFC)

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

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

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

and two ways to generate the cryptographic group generator g:

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

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

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

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

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

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

- *g != 0,1*
- *q divides p-1*

- *g^q mod p = 1*
- *g^x mod p = y*

for each FFC parameter set and key pair.

AK5: Curve25519 Key Generation

The evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated as specified in RFC 7748 using an approved random bit generator (RBG) and shall be written in littleendian order (least significant byte first. To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

Note: Assuming the PKV function of the good implementation will (using little-endian order):

- *Confirm the private and public keys are 32-byte values*
- *Confirm the three least significant bits of the first byte of the private key are zero*
- *Confirm the most significant bit of the last byte is zero*
- *Confirm the second most significant bit of the last byte is one*
- *Calculate the expected public key from the private key and confirm it matches the supplied public key*

The evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify 5 of the public key values so that they are incorrect, leaving five values unchanged (i.e. correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

FCS_CKM.1/SK Cryptographic Key Generation (Symmetric Encryption Key)

The inclusion of this selection-based component depends upon selection in FCS_CKM.1.1/KEK.

FCS_CKM.1.1/SK

The TSF shall generate **symmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [**assignment**: *cryptographic key generation algorithm*] and specified cryptographic key sizes [**assignment**: *cryptographic key sizes*] that meet the following: [**assignment**: *list of standards*].

The rows of Table 9 provide the allowable values for completion of the assignments for FCS_CKM.1.1/SK.

Table 9: Supported Methods for Symmetric Key Generation

Identifier	Key Type	Cryptographic Key Generation Algorithm	Key Sizes	List of Standards
RSK	[**selection**: *symmetric key, submask, authorization value*]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	[**selection**: *128, 192, 256, 512*] bits	NIST SP 800-133 (Section 7.1) with ISO 18031 as an approved RBG in addition to those in NIST SP 800-133 (Section 5).
DSK [**assignment**: *Identifier from Table 16: Key Derivation Functions*]	[**assignment**: *Key Type from Table 16: Key Derivation Functions*]	Derived from a Key Derivation Function as specified in FCS_CKM_EXT.5 [**assignment**: *Key Derivation Algorithm from Table 16: Key Derivation Function*]	[**assignment**: *Key sizes from Table 16: Key Derivation Functions*]	[**assignment**: *List of Standards from Table 16: Key Derivation Functions*]
PBK	[**selection**: *submask, authentication token, authorization value*]	Derived from a Password Based Key Derivation Function as specified in FCS_COP.1/PBKDF	[**assignment**: *Key sizes as specified in FCS_COP.1/PBKDF*]	[**assignment**: *Standards as specified in FCS_COP.1/PBKDF*]

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 "Symmetric KEKs..." is selected in FCS_CKM.1/KEK.

The selection of key size 512 bits is for the case of XTS-AES using AES-256. In the case of XTSAES for both AES-128 and AES-256, the developer is expected to ensure that the full key is generated using direct generation from the RBG as in NIST SP 800-133 section.

The ST author selects at least one algorithm from the RSK row if the ST supports creating keys directly from the output of the RBG without further conditioning, at least one algorithm from the DSK row should be selected if the ST supports key derivation functions which are usually seeded from RBG and then further conditioned to the appropriate key size, and at least one algorithm from the PBK row should be selected if the ST supports keys derived from passwords.

If DSK is selected, the selection-based SFR FCS_CKM_EXT.5 must be claimed by the TOE.

If PBK is selected, the selection-based SFR FCS_COP.1/PBKDF must be claimed by the TOE.

This requirement must be claimed by the TOE if at least one of FCS_CKM.1 or FCS_CKM.1/KEK chooses a selection related to generation of symmetric keys.

Evaluation Activities

FCS_CKM.1/SK
TSS
The evaluator shall examine the TSS to verify that it describes how the TOE obtains an SK through direct generation as specified in FCS_RBG_EXT.1, FCS_COP.1/KDF, or FCS_COP.1/PBKDF. The evaluator shall review the TSS to verify that it describes how the ST invokes the functionality described by FCS_RBG_EXT.1 and FCS_COP.1/PBKDF where applicable.

[conditional] If the symmetric key is generated by an RBG, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.
Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key types for all uses identified in the ST.
KMD
The evaluator shall confirm that the KMD describes, as applicable:

- *The RBG interface and how the ST uses it in symmetric key generation*
- *The KDF interface and how the ST uses it in symmetric key generation*
- *The PBKDF interface and how the ST uses it in symmetric key generation*
- *If the TOE uses the generated key in a key chain/hierarchy then the KMD shall describe how the ST uses the key as part of the key chain/hierarchy.*

Tests
For each selected key generation method, the evaluator shall configure the selected generation capability. 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. The evaluator shall perform the tests as described for FCS_COP.1/KDF and FCS_COP.1/PBKDF.

FCS_CKM.2 Cryptographic Key Establishment

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

FCS_CKM.2.1

The TSF shall establish cryptographic keys in accordance with a specified cryptographic key establishment method: [**selection**:

- *RSA-based key establishment schemes that meet the following: NIST Special Publication 800-56B Revision 2, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography",*
- *RSA-based key establishment schemes that meet the following: RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 8017, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2",*
- *Elliptic curve-based key establishment schemes that meet the following: [**selection**:*
  - *NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography",*
  - *RFC 7748, "Elliptic Curves for Security"*

  ],
- *Finite field-based key establishment schemes that meet the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography",*
- *Elliptic Curve Integrated Encryption Scheme (ECIES) that meets the following: [**selection**:*
  - *ANSI X9.63 - Public Key Cryptography for the Financial Services Industry Key Agreement and Key Transport Using Elliptic Curve Cryptography,*
  - *IEEE 1363a - Standard Specification for Public-Key Cryptography - Amendment 1: Additional Techniques,*
  - *ISO/IEC 18033-2 - Information Technology - Security Techniques - Encryption Algorithms - Part 2: Asymmetric Ciphers,*
  - *SECG SEC1 - Standards for Efficient Cryptography Group Elliptic Curve Cryptography, section 5.1 Elliptic Curve Integrated Encryption Scheme*

  ]

] ~~that meets the following: [*assignment: list of standards*]~~.

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

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

The ST author selects all key establishment schemes used for the selected cryptographic protocols.

The RSA-based key establishment schemes are described in Section 8 of NIST SP 800-56B Revision 2 [NIST-RSA]; however, Section 8 relies on implementation of other sections in SP 800-56B Revision 2.

The elliptic curves used for the key establishment scheme correlate with the curves specified in FCS_CKM.1/AK.

The selections in this SFR must be consistent with those for FCS_COP.1/KAT.

Evaluation Activities

FCS_CKM.2
TSS
The evaluator shall examine the TSS to ensure that ST supports at least one key establishment scheme. The evaluator also ensures that for each key establishment scheme selected by the ST in FCS_CKM.2.1 it also supports one or more corresponding methods selected in FCS_COP.1/KAT. If the ST selects RSA in FCS_CKM.2.1, then the TOE must support one or more of "KAS1," or "KAS2," "KTS-OAEP," from FCS_COP.1/KAT. If the ST selects elliptic curve-based, then the TOE must support one or more of "ECDH-NIST" or "ECDH-BPC" from FCS_COP.1/KAT.

If the ST selects Diffie-Hellman-based key establishment, then the TOE must support "DH" from FCS_COP.1/KAT.

Guidance
The evaluator shall verify that the guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme.
KMD
There are no KMD evaluation activities for this component.
Tests
Testing for this SFR is performed under the corresponding functions in FCS_COP.1/KAT.

FCS_CKM_EXT.5 Cryptographic Key Derivation

The inclusion of this selection-based component depends upon selection in FCS_CKM.1.1/KEK, FCS_COP.1.1/KeyEnc.

FCS_CKM_EXT.5.1

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

The rows of Table 18 provide the allowable values for completion of the assignments for FCS_CKM_EXT.5.1.

Table 10: Key Derivation Functions

Identifier	Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KeyDrv1	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Counter Mode using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 (Section 5.1) (KDF in Counter Mode) [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]
KeyDrv2	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Feedback Mode using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 (Section 5.2) (KDF in Feedback Mode) [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]
KeyDrv3	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Double Pipeline Iteration Mode using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 (Section 5.3) (KDF in n Double Pipeline Iteration Mode) [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]
KeyDrv4	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Intermediary keys	[**selection**: *exclusive OR (XOR), SHA-256, SHA-512*]	[**selection**: *128, 192, 256*]bits	[**selection**: *ISO-HASH, FIPS-SHA*]
KeyDrv5	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Concatenated keys	KDF in [**selection**: *Counter Mode, Feedback Mode, Double Pipeline Iteration Mode*] using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 [selection: (Section 5.1) (KDF in Counter Mode); (Section 5.2) (KDF in Feedback Mode); (Section 5.3) (KDF in Double-Pipeline Iteration Mode)] [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]
KeyDrv6	[**selection**: *symmetric*	Two keys	[**selection**: *AES-CCM, AES-GCM,*	[**selection**: *128, 192,*	[selection: see List of Standards in

	key, initialization vector, authentication token, authorization value, HMAC key, KMAC key]		*AES-CBC, AES-KWP, AES-KW, CAM-CBC, CAM-CCM, CAM-GCM*] from FCS_COP.1/SKC Symmetric Key table	*256*]bits	FCS_COP.1/SKC Symmetric Key table]
KeyDrv7	[**selection**: *symmetric key, secret IV, seed*]	Shared secret, salt, output length, fixed information	[**selection**: *hash function from FCS_COP.1/Hash, keyed hash from FCS_COP.1/HMAC*]	[**selection**: *128, 192, 256*]bits	(NIST-KDRV) sec 4 [**assignment**: *See List of Standards in FCS_COP.1/Hash and FCS_COP.1/HMAC*]
KeyDrv8	[**selection**: *symmetric key, secret IV, seed*]	Shared secret, salt, IV, output length, fixed information	[**assignment**: *keyed hash from FCS_COP.1/HMAC*]	[**selection**: *128, 192, 256*]bits	(NIST-KDRV) sec 5 [**assignment**: *see List of Standards in FCS_COP.1/Hash and FCS_COP.1/HMAC*]

Application Note: This SFR must be included in the ST if key derivation 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 "Derived KEKs..." is selected in FCS_CKM.1/KEK, or if "XOR" is selected in FCS_COP.1/KeyEnc.

Note that Camellia algorithms do not support 192-bit key sizes.

The interface referenced in the requirement could take different forms, the most likely of which is an application programming interface to an OS kernel. There may be various levels of abstraction. For Authorization Factor Submasks, the key size to be used in the HMAC falls into a range between L1 and L2 defined in ISO/IEC 10118 for the appropriate hash function (for example for SHA-256 L1 = 512, L2 =256) where L2 = k = L1.

General note: in order to use a NIST SP 800-108 conformant method of key derivation, the TOE is permitted to implement this with keys as derived as indicated in Key Derivation Functions table above, and with the algorithms as indicated in the same table.

NIST SP 800-131A Rev 1 allows the use of SHA-1 in these use cases.

KeyDrv5, KeyDrv6, and the XOR option in KeyDrv4 will create an “inverted key hierarchy” in which the TSF will combine two or more keys to create a third key. These same KDFs may also use a submask key as input, which could be an authorization factor or derived from a PBKDF. In these cases the ST author must explicitly declare this option and should present a reasonable argument that the entropy of the inputs to the KDFs will result in full entropy of the expected output.

If keys are combined, the ST author shall describe which method of combination is used in order to justify that the effective entropy of each factor is preserved.

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 is not required to be part of the TSS; it can be submitted as a separate document and marked as developer proprietary.

SP 800-56C specifies a two-step key derivation procedure that employs an extraction-thenexpansion technique for deriving keying material from a shared secret generated during a key establishment scheme. The Randomness Extraction step as described in Section 5 of SP 800-56C is followed by Key Expansion using the key derivation functions defined in SP 800-108.

This requirement must be claimed by the TOE if at least one of FCS_CKM.1/KEK, FCS_CKM.1/SK, or FCS_COP.1/KeyEnc chooses a selection related to key derivation.

If at least one of KeyDrv4, KeyDrv5, or KeyDrv6 is selected AND password-based key derivation is used to create at least one of the inputs, the selection-based SFR FCS_COP.1/PBKDF must also be claimed.

Evaluation Activities

FCS_CKM_EXT.5
TSS
The evaluator shall check that the TSS includes a description of the key derivation functions and shall check that this uses a key derivation algorithm and key sizes according to the specification selected in the ST out of the table as provided in the cPP table per row. The evaluator shall confirm that the TSS supports the selected methods.

If KeyDrv5 is selected, the evaluator shall verify that the TSS shows that the total length of the concatenated keys used as input to the KDF is greater than or equal to the length of the output from the KDF.

[conditional] If key combination is used to form a KEK, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR, a KDF, or encryption.

[conditional] If a KDF is used to form a KEK, the evaluator shall ensure that the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108.

[conditional] If key concatenation is used to derive KEKs (KeyDrv5), the evaluator shall ensure the TSS includes a description of the randomness extraction step, including the following:

- *The description must include how an approved untruncated MAC function is being used for*

the randomness extraction step and the evaluator must verify the TSS describes that the output length (in bits) of the MAC function is at least as large as the targeted security strength (in bits) of the parameter set employed by the key establishment scheme (see Tables 1-3 of SP 800-56C).

- *The description must include how the MAC function being used for the randomness extraction step is related to the PRF used in the key expansion and verify the TSS description includes the correct MAC function:*
  - *If an HMAC-hash is used in the randomness extraction step, then the same HMAC-hash (with the same hash function hash) is used as the PRF in the key expansion step.*
  - *If an AES-CMAC (with key length 128, 192, or 256 bits) is used in the randomness extraction step, then AES-CMAC with a 128-bit key is used as the PRF in the key expansion step.*
- *The description must include the lengths of the salt values being used in the randomness extraction step and the evaluator shall verify the TSS description includes correct salt lengths:*
  - *If an HMAC-hash is being used as the MAC, the salt length can be any value up to the maximum bit length permitted for input to the hash function hash.*
  - *If an AES-CMAC is being used as the MAC, the salt length shall be the same length as the AES key (i.e. 128, 192, or 256 bits).*

Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key types for all uses identified in the ST.
KMD
The evaluator shall examine the KMD to ensure that:

- *The KMD describes the complete key derivation chain and the description must be consistent with the description in the TSS. For all key derivations the TOE must use a method as described in the cPP table. There should be no uncertainty about how a key is derived from another in the chain.*
- *The length of the key derivation key is defined by the PRF. The evaluator should check whether the key derivation key length is consistent with the length provided by the selected PRF.*
- *If a key is used as an input to several KDFs, each invocation must use a distinct context string. If the output of a KDF execution is used for multiple cryptographic keys, those keys must be disjoint segments of the output.*

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 evaluator shall perform one or more of the following tests to verify the correctness of the key derivation function, depending on the specific functions that are supported:

Preconditions for testing:

- *Specification of input parameter to the key derivation function to be tested*
- *Specification of further required input parameters*
- *Access to derived keys*

The following table maps the data fields in the tests below to the notations used in SP 800-108 and SP 800-56C

Data Fields	***Notations***	
	SP 800-108	*SP 800-56C*
Pseudorandom function	*PRF*	*PRF*
Counter length	*r*	*r*
Length of output of PRF	*r*	*r*
Length of derived keying material	*L*	*L*
Length of input values	*I_length*	*I_length*
Pseudorandom input values I	*K1 (key derivation key)*	*Z (shared secret)*
Pseudorandom salt values		*S*
Randomness extraction MAC	*n/a*	*MAC*

The below tests are derived from Key Derivation using Pseudorandom Functions (SP 800-108) Validation System (KBKDFVS), Updated 4 January 2016, Section 6.2, from the National Institute of Standards and Technology.

KeyDrv1: Counter Mode Tests:
The evaluator shall determine the following characteristics of the key derivation function:

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Location of the counter relative to fixed input data: before, after, or in the middle.*
  - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values I.*

For each supported combination of I_length, MAC, salt, PRF, counter location, value of r, and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it. For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output.

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

KeyDrv2: Feedback Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Whether or not zero-length IVs are supported.*
- *Whether or not a counter is used, and if so:*
  - *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
  - *Location of the counter relative to fixed input data: before, after, or in the middle.*
    - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values L.*

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I and pseudorandom salt values. If the KDF supports zero-length IVs, five of these test vectors will be accompanied by pseudorandom IVs and the other five will use zerolength IVs. If zero-length IVs are not supported, each test vector will be accompanied by an pseudorandom IV. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KeyDrv3: Double Pipeline Iteration Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Whether or not a counter is used, and if so:*
  - *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
  - *Location of the counter relative to fixed input data: before, after, or in the middle.*
    - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values I.*

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KeyDrv4: Intermediate Keys Method

If the selected algorithm is a hash then the testing of the hash primitive is the only required Evaluation Activity. If the selected algorithm is XOR then no separate primitive testing is necessary.

KeyDrv5: Concatenated Keys Method

The evaluator should confirm that the combined length of the concatenated keys should be at least as long as the keysize of the selected methods. There are no other tests other than for the methods selected for this row performed for KeyDrv1, KeyDrv2, and KeyDrv3.

KeyDrv6: Two Keys Method

The evaluator should confirm that the combined length of the two keys should be at least as long as the keysize of the selected methods. There are no other tests other than for the methods selected for this row from FCD_COP.1/SK.

KeyDrv7: Shared Secret, Salt, Output Length, Fixed Information Method

For each supported selection of PRF, length of shared secret (Z) [selection: 128, 256] bits, length of salt (S) [selection: length of input block of PRF, one-half length of input block of PRF, 0] bits, output length (L) [selection: 128, 256] bits, and length of fixed information (FixedInfo) [selection: length of on input block of PRF, onehalf length of input block of PRF, 0] bits, the evaluator shall generate 10 test vectors that include pseudorandom input values for Z, salt values (for non-zero lengths, otherwise, omit) and fixed information (for non-zero lengths, otherwise, omit).

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KeyDrv8: Shared Secret, Salt, IV, Output Length, Fixed Information Method

For each supported selection of PRF, length of shared secret (Z), length of salt, length of initialization vector (IV), output length (L), and length of fixed information (FixedInfo), the evaluator shall generate 10 test vectors that include pseudorandom input values for Z, salt values (for non-zero lengths, otherwise, omit), IV (for non-zero lengths, otherwise, use a vector

of length equal to length of input block of PRF and fill with zeros), and fixed information (for non-zero lengths, otherwise, omit).

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

FCS_COP.1/Hash Cryptographic Operation (Hashing)

The inclusion of this selection-based component depends upon selection in FCS_COP.1.1/KAT, FCS_RBG_EXT.1.1, FPT_ROT_EXT.2.1, FPT_TUD_EXT.2.1.

FCS_COP.1.1/Hash

The TSF shall perform [*cryptographic hashing*] in accordance with a specified cryptographic algorithm [**selection**: *SHA-1*, *SHA-256*, *SHA-384*, *SHA-512*, *SHA-3-224*, *SHA-3-256*, *SHA-3-384*, *SHA-3-512*] that meet the following: [**selection**: *ISO/IEC 10118-3:2018*, *FIPS 180-4*]

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.

Specifically, this SFR must be included if "hash value of the public key" is selected in FPT_TUD_EXT.2, or if FCS_COP.1/KeyedHash is included in the ST.

The hash selection should be consistent with the overall strength of the algorithm used for signature generation. For example, the ST Author should choose SHA-256 for 2048-bit RSA or ECC with P-256, SHA-384 for 3072-bit RSA, 4096-bit RSA, or ECC with P-384, and SHA-512 for ECC with P-521. The ST author selects the standard based on the algorithms selected.

SHA-1 may be used for the following applications: generating and verifying hash-based message authentication codes (HMACs), key derivation functions (KDFs), and random bit/number generation (In certain cases, SHA-1 may also be used for verifying old digital signatures and time stamps, provided that this is explicitly allowed by the application domain).

Evaluation Activities

FCS_COP.1/Hash
TSS
The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Guidance
The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present.

Tests
SHA-1 and SHA-2 Tests
The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

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

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

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

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

Selected Long Messages Test Bit-oriented Mode
*The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 <= i <= m. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Selected Long Messages Test Byte-oriented Mode
*The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

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

SHA-3 Tests
The tests below are derived from the The Secure Hash Algorithm-3 Validation System (SHA3VS),

Updated: April 7, 2016, from the National Institute of Standards and Technology.

For each SHA-3-XXX implementation, XXX represents d, the digest length in bits. The capacity, c, is equal to 2d bits. The rate is equal to 1600-c bits.

The TSF hashing functions can be implemented with one of two orientations. The first is a bit-oriented mode that hashes messages of arbitrary length. The second is a byte-oriented mode that hashes messages that are an integral number of bytes in length (i.e., the length (in bits) of the message to be hashed is divisible by 8). Separate tests for each orientation are given below.

The evaluator shall perform all of the following tests for each hash algorithm and orientation implemented by the TSF and used to satisfy the requirements of this PP. The evaluator shall compare digest values produced by a known-good SHA-3 implementation against those generated by running the same values through the TSF.

Short Messages Test, Bit-oriented Mode
The evaluators devise an input set consisting of rate+1 short messages. The length of the messages ranges sequentially from 0 to rate bits. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Short Messages Test, Byte-oriented Mode
The evaluators devise an input set consisting of rate/8+1 short messages. The length of the messages ranges sequentially from 0 to rate/8 bytes, with each message being an integral number of bytes. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Selected Long Messages Test, Bit-oriented Mode
The evaluators devise an input set consisting of 100 long messages ranging in size from rate+(rate+1) to rate+(100(rate+1)), incrementing by rate+1. (For example, SHA-3-256 has a rate of 1088 bits. Therefore, 100 messages will be generated with lengths 2177, 3266, ..., 109988 bits.) The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Selected Long Messages Test, Byte-oriented Mode
The evaluators devise an input set consisting of 100 messages ranging in size from (rate+(rate+8)) to (rate+100(rate+8)), incrementing by rate+8. (For example, SHA-3-256 has a rate of 1088 bits. Therefore 100 messages will be generated of lengths 2184, 3280, 4376, ..., 110688 bits.) The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Pseudo-randomly Generated Messages Monte Carlo) Test, Byte-oriented Mode

The evaluators supply a seed of d bits (where d is the length of the message digest produced by the hash function to be tested. This seed is used by a pseudorandom function to generate 100,000 message digests. One hundred of the digests (every 1000th digest) are recorded as checkpoints. The TOE then uses the same procedure to generate the same 100,000 message digests and 100 checkpoint values. The evaluators then compare the results generated ensure that the correct result is produced when the messages are generated by the TSF.

FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash)

The inclusion of this selection-based component depends upon selection in FCS_COP.1.1/KAT, FCS_RBG_EXT.1.1, FCS_STG_EXT.3.1.

FCS_COP.1.1/KeyedHash

The TSF shall perform [*keyed hash message authentication*] in accordance with a specified cryptographic algorithm [**selection**: *HMAC-SHA-1*, *HMAC-SHA-256*, *HMAC-SHA-384*, *HMAC-SHA-512*, *KMAC-128*, *KMAC-256*] and cryptographic key sizes [**assignment**: *key size (in bits)*] that meet the following: [**selection**: *ISO/IEC 9797-2:2011 Section 7 "MAC Algorithm 2"*, *[NIST-KDV] section 4 "KMAC"*].

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.

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

Evaluation Activities

FCS_COP.1/KeyedHash
TSS
The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC and KMAC functions: output MAC length used.
Guidance
There are no guidance evaluation activities for this component.
KMD
There are no KMD evaluation activities for this component.
Tests
The following test requires the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

This test is derived from The Keyed-Hash Message Authentication Code Validation System (HMACVS), updated 6 May 2016.

The evaluator shall provide 15 sets of messages and keys for each selected hash algorithm and hash length/key size/MAC size combination. The evaluator shall have the TSF generate HMAC or KMAC tags for these sets of test data. The evaluator shall verify that the resulting HMAC or KMAC tags match the results from submitting the same inputs to a known-good implementation of the HMAC or KMAC function, having the same characteristics.

FCS_COP.1/KAT Cryptographic Operation (Key Agreement/Transport)

The inclusion of this selection-based component depends upon selection in FCS_COP.1.1/KeyEnc.

FCS_COP.1.1/KAT

The TSF shall perform [*cryptographic key agreement/transport*] using the supported methods for key agreement/transport defined by the following rows of Table 11: [**selection**: *KAS1*, *KAS2*, *KTS-OAEP*, *RSAES-PKCS1-v1_5*, *ECDH-NIST*, *ECDH-BPC*, *DH*, *Curve25519*, *ECIES*].

Table 11: Supported Methods for Key Agreement/Transport Operation

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
KAS1	RSA-single party	[**selection**: *2048*, *3072*, *4096*, *6144*, *8192*]bits	NIST SP 800-56Br2 section 8.2
KAS2	RSA-both party	[**selection**: *2048*, *3072*, *4096*, *6144*, *8192*]bits	NIST SP 800-56Br2 section 8.3
KTS-OAEP	RSA	[**selection**: *2048*, *3072*, *4096*, *6144*, *8192*]bits	NIST SP 800-56Br2 section 9
RSAES-PKCS1-v1_5	RSA	[**selection**: *2048*, *3072*, *4096*, *6144*, *8192*]bits	RFC 8017 Section 7.2
ECDH-NIST	ECDH with NIST curves	[**selection**: *256 (P-256)*, *384 (P-384)*, *512 (P-521)*]	NIST SP 800-56Ar3
ECDH-BPC	ECDH with Brainpool curves	[**selection**: *256 (brainpoolP256r1)*, *384 (brainpoolP384r1*, *512 (brainpoolP512r1)*]	RFC 5639 (Section 3)
DH	Diffie-Hellman	[**selection**: *2048*, *3072*, *4096*, *6144*, *8192*]bits	NIST SP 800-56A rev 3, [**selection**: • *RFC 3526 Section* [***selection***: *3, 4, 5, 6, 7*], • *RFC 7919 Appendices* [***selection***: *A.1, A.2, A.3, A.4, A.5*]]
Curve25519	ECDH	256 bits	RFC 7748
ECIES	ECIES	[**selection**: *256*, *384*, *512*]bits	[**selection**: *ANSI X9.63*, *IEEE 1363a*, *ISO/IEC 18033-2 Part 2*, *SECG SEC1 sec 5.1*]

Application Note: This SFR must be included in the ST if key agreement or transport 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.

Specifically, this SFR must be included if "AE1" is selected in FCS_COP.1/KeyEnc.

The selections in this SFR should be consistent with the algorithms selected in FCS_CKM.2.

Evaluation Activities

FCS_COP.1/KAT
TSS
The evaluator shall ensure that the selected RSA and ECDH key agreement/transport schemes correspond to the key generation schemes selected in FCS_CKM.1/AK, and the key establishment schemes selected in FCS_CKM.2 If the ST selects DH, the TSS shall describe how the implementation meets the relevant sections of RFC 3526 (Section 3-7) and RFC 7919 (Appendices A.1-A.5). If the ST selects ECIES, the TSS shall describe the key sizes and algorithms (e.g. elliptic curve point multiplication, ECDH with either NIST or Brainpool curves, AES in a mode permitted by FCS_COP.1/SKC, a SHA-2 hash algorithm permitted by FCS_COP.1/Hash, and a MAC algorithm permitted by FCS_COP.1/KeyedHash) that are supported for the ECIES implementation.

The evaluator shall ensure that, for each key agreement/transport scheme, the size of the derived keying material is at least the same as the intended strength of the key agreement/transport scheme, and where feasible this should be twice the intended security strength of the key agreement/transport scheme.

Table 2 of NIST SP 800-57 identifies the key strengths for the different algorithms that can be used for the various key agreement/transport schemes.

Guidance
There are no guidance evaluation activities for this component.
KMD
There are no KMD 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 evaluator shall verify the implementation of the key generation routines of the supported schemes using the following tests:

If ECDH-NIST or ECDH-BPC is claimed:

SP800-56A Key Agreement Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

If KAS1, KAS2, KTS-OAEP, or RSAES-PKCS1-v1_5 is claimed:

SP800-56B and PKCS#1 Key Establishment Schemes

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

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

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

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

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

DH:

The evaluator shall verify the correctness of each TSF implementation of each supported Diffie-Hellman group by comparison with a known good implementation.

Curve25519:

The evaluator shall verify a TOE's implementation of the key agreement scheme using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specification. These components include the calculation of the shared secret K and the hash of K.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement role and hash function combination, the tester shall generate 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the shared secret value K, and the hash of K. The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value K and compare the hash generated from this value.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results. To conduct this test, the evaluator generates a set of 30 test vectors consisting of data sets including the evaluator's public keys and the TOE's public/private key pairs.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value K or the hash of K. At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

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

ECIES:

The evaluator shall verify the correctness of each TSF implementation of each supported use of ECIES by comparison with a known good implementation.

FCS_COP.1/KeyEnc Cryptographic Operation (Key Encryption)

The inclusion of this selection-based component depends upon selection in FCS_STG_EXT.1.1.

FCS_COP.1.1/KeyEnc

The TSF shall perform [*key encryption and decryption*] using the methods defined in the following rows of Table 12: [**selection**: *SE1*, *AE1*, *XOR*]

Table 12: Supported Methods for Key Encryption Operation

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
SE1	Symmetric [**selection**: *AES-CCM*, *AES-GCM*, *AES-CBC*, *AES-CTR*, *AES-KWP*, *AES-KW*]	[**selection**: *128*, *192*, *256*] bits	See FCS_COP.1/SKC
AE1	Asymmetric KTS-OAEP	[**selection**: *2048*, *3072*] bits	See FCS_COP.1/KAT
XOR	Exclusive OR operation	[**selection**: *128*, *192*, *256*] bits	See FCS_CKM_EXT.5

Application Note: This SFR must be included in the ST if key encryption 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.

Specifically, this SFR must be included in the ST if "software-based" is selected in FCS_STG_EXT.1.

A TOE will use this requirement to specify how the Key Encryption Key (KEK) wraps a symmetric encryption key. A TOE will always need this requirement in order to capture the last stage of the key chain in which the Key Encryption Key (KEK) wraps the symmetric encryption key.

If XOR is selected, the selection-based SFR FCS_CKM_EXT.5 must be claimed by the TOE.

Evaluation Activities

FCS_COP.1/KeyEnc
TSS
The evaluator shall examine the TSS to ensure that it identifies whether the implementation of this cryptographic operation for key encryption (including key lengths and modes) is an implementation that is tested in FCS_COP.1/SKC.

The evaluator shall check that the TSS includes a description of the key wrap functions and shall check that this uses a key wrap algorithm and key sizes according to the specification selected in the ST out of the table as provided in the cPP table.

Guidance
The evaluator checks the AGD documents to confirm that the instructions for establishing the evaluated configuration use only those key wrap functions selected in the ST. If multiple key access modes are supported, the evaluator shall examine the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

KMD
The evaluator shall examine the KMD to ensure that it describes when the key wrapping occurs, that the KMD description is consistent with the description in the TSS, and that for all keys that are wrapped the TOE uses a method as described in the cPP table. No uncertainty should be left over which is the wrapping key and the key to be wrapped and where the wrapping key potentially comes from i.e. is derived from.

If "AES-GCM" or "AES-CCM" is used the evaluator shall examine the KMD to ensure 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. Moreover in the case of GCM, he must ensure that, at each invocation of GCM, the length of the plaintext is at most (2^32)-2 blocks.

Tests
Refer to FCS_COP.1/SKC for the required testing for each symmetric key wrapping method selected from the table and to FCS_COP.1/KAT for the required testing for each asymmetric key wrapping method selected from the table. Each distinct implementation shall be tested separately.

If the implementation of the key encryption operation is the same implementation tested under FCS_COP.1/SKC or FCS_COP.1/KAT, and it has been tested with the same key lengths and modes, then no further testing is required. If key encryption uses a different implementation, (where "different implementation" includes the use of different key lengths or modes), then the evaluator shall additionally test the key encryption implementation using the corresponding tests specified for FCS_COP.1/SKC or FCS_COP.1/KAT.

FCS_COP.1/PBKDF Cryptographic Operation (Password-Based Key Derivation Functions)

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

FCS_COP.1.1/PBKDF

The TSF shall perform [*password-based key derivation functions*] in accordance with a specified cryptographic algorithm [*HMAC-* [**selection**: *SHA-256*, *SHA-384*, *SHA-512*]], with [**assignment**: *integer number greater than or equal to 1000*] iterations, and output cryptographic key sizes [**selection**: *128*, *192*, *256*]bits that meet the following standard: [*NIST SP 800-132*].

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.

The ST must condition a password into a string of bits prior to using it as input to algorithms that form SKs and KEKs. The ST can perform conditioning using one of the identified hash functions or the process described in NIST SP 800-132; the ST author selects the method used. NIST SP 800-132 requires the use of a pseudo-random function (PRF) consisting of HMAC with an approved hash function.

Appendix A of NIST SP 800-132 recommends setting the iteration count in order to increase the computation needed to derive a key from a password and, therefore, increase the workload of performing a dictionary attack.

The TOE must claim this requirement if it claims FCS_CKM.1/SK and selects an algorithm in the PBK row or claims FCS_CKM_EXT.5 and selects at least one of KeyDrv4, KeyDrv5, or KeyDrv6 AND uses password-based key derivation to create at least one of the inputs.

Evaluation Activities

FCS_COP.1/PBKDF
TSS
The evaluator shall review the TSS to verify that it contains a description of the PBKDF. The evaluator shall also confirm the ST supports the selected hash function itself. The evaluator shall confirm that the TSS contains a description of how the TOE ensures that the output of the PBKDF is at least the same length as that specified in FCS_CKM.1/SK and for the KeyDrv4, KeyDrv5, or KeyDrv6 in FCS_CKM_EXT.5.

If the ST performs additional conditioning, whitening, or manipulation of the password or passphrase before applying the PBKDF, or to the output of the PBKDF, the evaluator shall ensure that the TSS describes the actions and provides assurance that the TSF does not negatively impact the entropy of the PBKDF output.

If any manipulation of the key is performed in forming the submask that will be used to form the KEK, that process shall be described in the TSS.
Guidance
There are no AGD evaluation activities for this component.
KMD
There are no KMD evaluation activities for this component.
Tests
No explicit testing of the formation of the submask from the input password is required.

For the NIST SP 800-132-based conditioning of the passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1/HMAC).

The evaluator shall verify that the iteration count for PBKDFs performed by the TOE comply with NIST SP 800-132 by ensuring that the TSS contains a description of the estimated time required to derive key material from passwords and how the TOE increases the computation time for password-based key derivation (including but not limited to increasing the iteration count).

FCS_COP.1/SigGen Cryptographic Operation (Signature Generation)

The inclusion of this selection-based component depends upon selection in FCS_STG_EXT.3.1.

FCS_COP.1.1/SigGen

The TSF shall perform [*digital signature generation*] using the supported methods for signature generation defined in the following rows of Table 13 [**selection**: *RSASSA-PKCS1*, *DSS 2*, *DSS 3*, *RSASSA-PSS*, *ECDSA*].

Table 13: Supported Methods for Signature Generation Operation

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
RSASSA-PKCS1	RSASSA-PKCS1-v1_5 using [**selection**: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*]	[**selection**: *2048 bit, 3072 bit*]	[**selection**: *RFC 8017, PKCS #1 v2.2 (Section 8.2), FIPS186-4, (Section 5.5)*] (RSASSA-PKCS1-v1_5) [**selection**: *ISO10118-3 (Clause 10, 11), FIPS180-4 (Section 6)*] (SHA)
DSS 2	Digital signature scheme 2 using [**selection**: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*]	[**selection**: *2048 bit, 3072 bit*]	ISO9796-2, (Clause 9) (Digital signature scheme 2) [**selection**: *ISO10118-3 (Clause 10, 11), FIPS180-4 (Section 6)*] (SHA)
DSS 3	Digital signature scheme 3 using [**selection**: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*]	[**selection**: *2048 bit, 3072 bit*]	ISO9796-2, (Clause 10) (Digital signature scheme 3) [**selection**: *ISO10118-3 (Clause 10, 11), FIPS180-4 (Section 6)*] (SHA)
RSASSA-PSS	RSASSA-PSS using [**selection**: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*]	[**selection**: *2048 bit, 3072 bit*]	[RFC8017, PKCS#1v2.2 (Section 8.1)] (RSASSAPSS) [**selection**: *ISO10118-3 (Clause 10, 11), FIPS180-4 (Section 6)*] (SHA)
ECDSA	ECDSA on [**selection**: *brainpoolP256r1, brainpoolP384r1, brainpoolP512r1, NIST P-256, NIST P-384, NIST P-521*] using [**selection**: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*]	[**selection**: *2048 bit, 3072 bit*]	[**selection**: • [***selection***: *ISO14888-3, FIPS186-4 (Section 6)] (EDCSA),* • *RFC5639 (Section 3) (Brainpool Curves),* • *FIPS186-4 (Appendix D.1.2) (NIST Curves)*] [**selection**: *ISO10118-3 (Clause 10, 11), FIPS180-4 (Section 6)*] (SHA)

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.

Evaluation Activities

FCS_COP.1/SigGen
TSS
The evaluator shall examine the TSS to ensure that all signature generation functions use the approved algorithms and key sizes.
Guidance
There are no AGD evaluation activities for this component.
KMD

There are no KMD 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.

Each section below contains tests the evaluators must perform for each selected digital signature scheme. Based on the assignments and selections in the requirement, the evaluators choose the specific activities that correspond to those selections.

The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are not found on the TOE in its evaluated configuration.

If SigGen1: RSASSA-PKCS1-v1_5or SigGen4: RSASSA-PSS is claimed:

The below test is derived from The 186-4 RSA Validation System (RSA2VS). Updated 8 July 2014, Section 6.3, from the National Institute of Standards and Technology.

To test the implementation of RSA signature generation the evaluator uses the system under test to generate signatures for 10 messages for each combination of modulus size and SHA algorithm. The evaluator then uses a known-good implementation and the associated public keys to verify the signatures.

If SigGen2: Digital Signature Scheme 2 (DSS2) or SigGen3: Digital Signature Scheme 3 (DSS3):

To test the implementation of DSS2/3 signature generation the evaluator uses the system under test to generate signatures for 10 messages for each combination of SHA algorithm, hash size and key size. The evaluator them uses a known-good implementation and the associated public keys to verify the signatures.

If SigGen5: ECDSA is claimed:

The below test is derived from The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS). Updated 18 March 2014, Section 6.4, from the National Institute of Standards and Technology.

To test the implementation of ECDSA signature generation the evaluator uses the system under test to generate signatures for 10 messages for each combination of curve, SHA algorithm, hash size, and key size. The evaluator then uses a known-good implementation and the associated public keys to verify the signatures.

FCS_COP.1/SigVer Cryptographic Operation (Signature Verification)

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.1, FPT_TUD_EXT.1.1.

FCS_COP.1.1/SigVer

Refinement: The TSF shall perform The TSF shall perform [*digital signature verification*] using the supported methods for signature verification defined in the following rows of Table 14 [**selection**: *RSASSA-PKCS1*, *DSS 2*, *DSS 3*, *RSASSA-PSS*, *ECDSA*].

Table 14: Supported Methods for Signature Verification Operation

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
RSASSA-PKCS1	RSASSA-PKCS1-v1_5 using [**selection**: *SHA-256*, *SHA-384*, *SHA-512*, *SHA3-256*, *SHA3-384*, *SHA3-512*]	[**selection**: *2048 bit*, *3072 bit*]	[**selection**: *RFC 8017*, *PKCS #1 v2.2 (Section 8.2)*, *FIPS186-4*, *(Section 5.5)*] (RSASSA-PKCS1-v1_5) [**selection**: *ISO10118-3 (Clause 10*, *11)*, *FIPS180-4 (Section 6)*] (SHA)
DSS 2	Digital signature scheme 2 using [**selection**: *SHA-256*, *SHA-384*, *SHA-512*, *SHA3-256*, *SHA3-384*, *SHA3-512*]	[**selection**: *2048 bit*, *3072 bit*]	ISO9796-2, (Clause 9) (Digital signature scheme 2) [**selection**: *ISO10118-3 (Clause 10*, *11)*, *FIPS180-4 (Section 6)*] (SHA)
DSS 3	Digital signature scheme 3 using [**selection**: *SHA-256*, *SHA-384*, *SHA-512*, *SHA3-256*, *SHA3-384*, *SHA3-512*]	[**selection**: *2048 bit*, *3072 bit*]	ISO9796-2, (Clause 10) (Digital signature scheme 3) [**selection**: *ISO10118-3 (Clause 10*, *11)*, *FIPS180-4 (Section 6)*] (SHA)
RSASSA-PSS	RSASSA-PSS using [**selection**: *SHA-256*, *SHA-384*, *SHA-512*, *SHA3-256*, *SHA3-384*, *SHA3-512*]	[**selection**: *2048 bit*, *3072 bit*]	[RFC8017, PKCS#1v2.2 (Section 8.1)] (RSASSAPSS) [**selection**:

			ISO10118-3 (Clause 10, 11), FIPS180-4 (Section 6)] (SHA)
ECDSA	ECDSA on [**selection**: *brainpoolP256r1, brainpoolP384r1, brainpoolP512r1, NIST P-256, NIST P-384, NIST P-521*] using [**selection**: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*]	[**selection**: *2048 bit, 3072 bit*]	[**selection**: • [***selection**: ISO14888-3, FIPS186-4 (Section 6)] (EDCSA),* • *RFC5639 (Section 3) (Brainpool Curves),* • *FIPS186-4 (Appendix D.1.2) (NIST Curves)*] [**selection**: *ISO10118-3 (Clause 10, 11), FIPS180-4 (Section 6)*] (SHA)

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.

The ST author should choose the algorithm implemented to perform vertification of digital signatures. if more than one algorithm is available, this requirement should be iterated to specify the functionality. 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. As the preferred approach for digital signatures, elliptic curves will be required after all the necessary standards and other supporting information are fully established.

If cryptographic signature vertification services are provided to the TOE or to tenant software by a local DSC, then the ST should include an instance of this SFR with instance identifier "(DSC)."

Evaluation Activities

FCS_COP.1/SigVer

TSS

The evaluator shall check the TSS to ensure that it describes the overall flow of the signature verification. This should at least include identification of the format and general location (e.g., "firmware on the hard drive device" rather than “memory location 0x00007A4B") of the data to be used in verifying the digital signature; how the data received from the operational environment are brought onto the device; and any processing that is performed that is not part of the digital signature algorithm (for instance, checking of certificate revocation lists).

Guidance

There are no AGD evaluation activities for this component.

KMD

There are no KMD 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.

Each section below contains tests the evaluators must perform for each selected digital signature scheme. Based on the assignments and selections in the requirement, the evaluators choose the specific activities that correspond to those selections.

The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are not found on the TOE in its evaluated configuration.

SigVer1: RSASSA-PKCS1-v1_5 and SigVer4: RSASSA-PSS

These tests are derived from The 186-4 RSA Validation System (RSA2VS), updated 8 Jul 2014, Section 6.4.

The FIPS 186-4 RSA Signature Verification Test tests the ability of the TSF to recognize valid and invalid signatures. The evaluator shall provide a modulus and three associated key pairs (d, e) for each combination of selected SHA algorithm, modulus size and hash size. Each private key d is used to sign six pseudorandom messages each of 1024 bits. For five of the six messages, the public key (e), message, IR format, padding, or signature is altered so that signature verification should fail. The test passes only if all the signatures made using unaltered parameters result in successful signature verification, and all the signatures made using altered parameters result in unsuccessful signature verification.

SigVer5: ECDSA on NIST and Brainpool Curves

These tests are derived from The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS), updated 18 Mar 2014, Section 6.5.

The FIPS 186-4 ECC Signature Verification Test tests the ability of the TSF to recognize valid and invalid signatures. The evaluator shall provide a modulus and associated key pair (x, y) for each combination of selected curve, SHA algorithm, modulus size, and hash size. Each private key (x) is used to sign 15 pseudorandom messages of 1024 bits. For eight of the fifteen messages, the message, IR format, padding, or signature is altered so that signature verification should fail. The test passes only if all the signatures made using unaltered parameters result in successful signature verification, and all the signatures made using altered parameters result in unsuccessful signature verification.

SigVer2: Digital Signature Scheme 2

The following or equivalent steps shall be taken to test the TSF.

For each supported modulus size, underlying hash algorithm, and length of the trailer field (1- or 2-byte), the evaluator shall generate NT sets of recoverable message (M1), non-recoverable message (M2), salt, public key and signature (Σ).

1. *N_T shall be greater than or equal to 20.*
2. *The length of salts shall be selected from its supported length range of salt. The typical length of salt is equal to the output block length of underlying hash algorithm (see 9.2.2 of ISO/IEC 9796-2:2010).*
3. *The length of recoverable messages should be selected by considering modulus size, output block length of underlying hash algorithm, and length of salt (L_S). As described in Annex D of ISO/IEC 9796-2:2010, it is desirable to maximise the length of recoverable message. The following table shows the maximum bit-length of recoverable message that is divisible by 512, for some combinations of modulus size, underlying hash algorithm, and length of salt. None that 2-byte trailer field is assumed in calculating the maximum length of recoverable message*

Maximum length of recoverable message divisible by 512 (bits)	***Modulus size (bits)***	***Underlying hash algorithm (bits)***	***Length of salt L_S (bits)***
1536	*2048*	*SHA-256*	*128*
1024			*256*
1024		*SHA-512*	*128*
1024			*256*
512			*512*
2560	*3072*	*SHA-256*	*128*
2048			*256*
2048		*SHA-512*	*128*
2048			*256*
1536			*512*

4. *The length of non-recoverable messages should be selected by considering the underlying hash algorithm and usages. If the TSF is used for verifying the authenticity of software/firmware updates, the length of non-recoverable messages should be selected greater than or equal to 2048-bit. With this length range, it means that the underlying hash algorithm is also tested for two or more input blocks.*
5. *The evaluator shall select approximately one half of N_T sets and shall alter one of the values (non-recoverable message, public key exponent or signature) in the sets. In altering public key exponent, the evaluator shall alter the public key exponent while keeping the exponent odd. In altering signatures, the following ways should be considered:*
   a. *Altering a signature just by replacing a bit in the bit-string representation of the signature*
   b. *Altering a signature so that the trailer in the message representative cannot be interpreted. This can be achieved by following ways:*
      - *Setting the rightmost four bits of the message representative to the values other than '1100'.*
      - *In the case when 1-byte trailer is used, setting the rightmost byte of the message representative to the values other than '0xbc', while keeping the rightmost four bits to '1100'.*
      - *In the case when 2-byte trailer is used, setting the rightmost byte of the message representative to the values other than '0xcc', while keeping the rightmost four bits to '1100'.*
   c. *In the case when 2-byte trailer is used, altering a signature so that the hash algorithm identifier in the trailer (i.e. the left most byte of the trailer) does not correspond to hash algorithms identified in the SFR. The hash algorithm identifiers are 0x34 for SHA-256 (see Clause 10 of ISO/IEC 10118-3:2018), and 0x35 for SHA-512 (see Clause 11 of ISO/IEC 10118-3:2018).*
   d. *Let L_S be the length of salt, altering a signature so that the intermediate bit string D in the message representative is set to all zeroes except for the rightmost L_S bits of D.*
   e. *(non-conformant signature length) Altering a signature so that the length of signature Σ is changed to modulus size and the most significant bit of signature Σ is set equal to '1'.*
   f. *(non-conformant signature) Altering a signature so that the integer converted from signature Σ is greater than modulus n.*

The evaluator shall supply the NT sets to the TSF and obtain in response a set of NT Verification-Success or Verification-Fail values. When the VerificationSuccess is obtained, the evaluator shall also obtain recovered message (M 1).*

The evaluator shall verify that Verification-Success results correspond to the unaltered sets and Verification-Fail results correspond to the altered sets.

For each recovered message, the evaluator shall compare the recovered message (M1) with the corresponding recoverable message (M 1) in the unaltered sets.*

The test passes only if all the signatures made using unaltered sets result in Verification-Success, each recovered message (M 1) is equal to corresponding M 1 in the unaltered sets, and all the signatures made using altered sets result in Verification-Fail.*

SigVer3: Digital Signature Scheme 3

The evaluator shall perform the test described in SigVer2: Digital Signature Scheme 2 while using a fixed salt for NT sets.

FCS_COP.1/SKC Cryptographic Operation (Symmetric Key Cryptography)

The inclusion of this selection-based component depends upon selection in FCS_COP.1.1/KAT, FCS_COP.1.1/KeyEnc, FCS_RBG_EXT.1.1, FCS_STG_EXT.3.1.

FCS_COP.1.1/SKC

The TSF shall perform [*data encryption/decryption*] using the supported symmetric-key cryptography methods defined in the following rows of Table 15 [**selection**: *AES-CCM*, *AES-GCM*, *AES-CBC*, *AES-CTR*, *XTS-AES*, *AES-KWP*, *AES-KW*].

Table 15: Supported Methods for Symmetric Key Cryptography Operation

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
AES-CCM	AES in CCM mode with unpredictable, nonrepeating nonce, minimum size of 64 bits	[**selection**: *128 bits*, *192 bits*, *256 bits*]	ISO 18033-3 (AES) ISO 19772, Clause 8 (CCM) NIST SP800-38C (CCM)
AES-GCM	AES in GCM mode with non-repeating IVs; IV length must be equal to 96 bits; the deterministic IV construction method (SP800-38D, Section 8.2.1) must be used; the MAC length t must be one of the values [**selection**: *96*, *104*, *112*, *120*, *128*]	[**selection**: *128 bits*, *192 bits*, *256 bits*]	ISO 18033-3 (AES) ISO 19772, Clause 11 (GCM) NIST SP800-38D (GCM)
AES-CBC	AES in CBC mode with non-repeating and unpredictable IVs	[**selection**: *128 bits*, *192 bits*, *256 bits*]	ISO 18033-3 (AES) ISO 10116 (CBC) NIST SP800-38A (CBC)
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	[**selection**: *128 bits*, *192 bits*, *256 bits*]	ISO 18033-3 (AES) ISO 10116 (CTR) NIST SP800-38A (CTR)
XTS-AES	AES in XTS mode with unique [**selection**: *consecutive non-negative integers starting at an arbitrary non-negative integer*, *data unit sequence numbers*] tweak values	[**selection**: *256 bits*, *512 bits*]	ISO 18033-3 (AES) [**selection**: *IEEE 1619*, *NIST SP800-38E*](XTS)
AES-KWP	KWP based on AES	[**selection**: *128 bits*, *192 bits*, *256 bits*]	ISO 18033-3 (AES) NIST SP 800-38F, sec. 6.3 (KWP)
AES-KW	KW based on AES	[**selection**: *128 bits*, *192 bits*, *256 bits*]	ISO 18033-3 (AES) NIST SP 800-38F, sec. 6.2 (KW) ISO/IEC 19772, clause 7 (key wrap)

Application Note: This SFR must be included in the ST if symettric-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 includes any of the following selections:

- "CTR_DRBG (AES)" in FCS_RBG_EXT.1
- "SE1" in FCS_COP.1/KeyEnc
- "ECIES" in FCS_COP.1/KAT
- "AES-*" in FCS_STG_EXT.3

Evaluation Activities

FCS_COP.1/SKC
TSS
The evaluator shall check that the TSS includes a description of encryption functions used for symmetric key encryption. The evaluator should check that this description of the selected

encryption function includes the key sizes and modes of operations as specified in the cPP table 9 "Supported Methods for Symmetric Key Cryptography Operation."

The evaluator shall check that the TSS describes the means by which the TOE satisfies constraints on algorithm parameters included in the selections made for 'cryptographic algorithm' and 'list of standards'.

Guidance
If the product supports multiple modes, the evaluator shall examine the vendor's documentation to determine that the method of choosing a specific mode/key size by the end user is described.
KMD
The evaluator shall examine the KMD to ensure that the points at which symmetric key encryption and decryption occurs are described, and that the complete data path for symmetric key encryption is described. The evaluator checks that this description is consistent with the relevant parts of the TSS.

Assessment of the complete data path for symmetric key encryption includes confirming that the KMD describes the data flow from the device's host interface to the device's non-volatile memory storing the data, and gives information enabling the user data datapath to be distinguished from those situations in which data bypasses the data encryption engine (e.g. read-write operations to an unencrypted Master Boot Record area). The evaluator shall ensure that the documentation of the data path is detailed enough that it thoroughly describes the parts of the TOE that the data passes through (e.g. different memory types, processors and co-processors), its encryption state (i.e. encrypted or unencrypted) in each part, and any places where the data is stored. For example, any caching or buffering of the data should be identified and distinguished from the final destination in non-volatile memory (the latter represents the location from which the host will expect to retrieve the data in future).

If support for AES-CTR is claimed and the counter value source is internal to the TOE, the evaluator shall verify that the KMD describes the internal counter mechanism used to ensure that it provides unique counter block values.

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.

Preconditions for testing:

- *Specification of keys as input parameter to the function to be tested*
- *specification of required input parameters such as modes*
- *Specification of user data (plaintext)*
- *Tapping of encrypted user data (ciphertext) directly in the non-volatile memory*

AES-CBC:
For the AES-CBC tests described below, the plaintext, ciphertext, and IV values shall consist of 128-bit blocks. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known-good implementation.

These tests are intended to be equivalent to those described in NIST's AES Algorithm Validation Suite (AESAVS) (http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf). It is not recommended that evaluators use values obtained from static sources such as the example NIST's AES Known Answer Test Values from the AESAVS document, or use values not generated expressly to exercise the AES-CBC implementation.

AES-CBC Known Answer Tests

KAT-1 (GFSBox): To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different plaintext values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros.

To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different ciphertext values for each selected key size and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using a key value of all zeros and an IV of all zeros.

KAT-2 (KeySBox): To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros.

To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the plaintext that results from AES-CBC decryption of an all-zeros ciphertext using the given key and an IV of all zeros.

KAT-3 (Variable Key): To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of keys for each selected key size (as described below) and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using each key and an IV of all zeros.

Key i in each set shall have the leftmost i bits set to ones and the remaining bits to zeros, for values of i from 1 to the key size. The keys and corresponding ciphertext are listed in AESAVS, Appendix E.

To test the decrypt functionality of AES-CBC, the evaluator shall use the same keys as above to decrypt the ciphertext results from above. Each decryption should result in an all-zeros plaintext.

KAT-4 (Variable Text): To test the encrypt functionality of AES-CBC, for each selected key size, the evaluator shall supply a set of 128-bit plaintext values (as described below) and obtain the ciphertext values that result from AES-CBC encryption of each plaintext value using a key of each size and IV consisting of all zeros.

Plaintext value i shall have the leftmost i bits set to ones and the remaining bits set to zeros, for values of i from 1 to 128. The plaintext values are listed in AESAVS, Appendix D.

To test the decrypt functionality of AES-CBC, for each selected key size, use the plaintext values from above as ciphertext input, and AES-CBC decrypt each ciphertext value using key of each size consisting of all zeros and an IV of all zeros.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting nine i-block messages for each selected key size, for $2 \leq i \leq 10$. For each test, the evaluator shall supply a key, an IV, and a plaintext message of length i blocks, and encrypt the message using AES-CBC. The resulting ciphertext values shall be compared to the results of encrypting the plaintext messages using a known good implementation.

The evaluator shall test the decrypt functionality by decrypting nine i-block messages for each selected key size, for 2 ≤ i ≤ 10. For each test, the evaluator shall supply a key, an IV, and a ciphertext message of length i blocks, and decrypt the message using AES-CBC. The resulting plaintext values shall be compared to the results of decrypting the ciphertext messages using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality for each selected key size using 100 3-tuples of pseudo-random values for plaintext, IVs, and keys.

The evaluator shall supply a single 3-tuple of pseudo-random values for each selected key size. This 3-tuple of plaintext, IV, and key is provided as input to the below algorithm to generate the remaining 99 3-tuples, and to run each 3-tuple through 1000 iterations of AES-CBC encryption.

```
# Input: PT, IV, Key
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]
        If (KeySize == 128) Key[i+1] = Key[i] xor CT[j]
        If (KeySize == 192) Key[i+1] = Key[i] xor (last 64 bits of CT[j-1] || CT[j])
        If (KeySize == 256) Key[i+1] = Key[i] xor ((CT[j-1] | CT[j])
        IV[i+1] = CT[j]
        PT[0] = CT[j-1]
}
```

The ciphertext computed in the 1000th iteration (CT[999]) is the result for each of the 100 3-tuples for each selected key size. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

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

AES-CCM:

These tests are intended to be equivalent to those described in the NIST document, "The CCM Validation System (CCMVS)," updated 9 Jan 2012, found at http://csrc.nist.gov/groups/STM/cavp/documents/mac/CCMVS.pdf.

It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

- ***Keys:** All supported and selected key sizes (e.g., 128, 192, or 256 bits).*
- ***Associated Data:** Two or three values for associated data length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.*
- ***Payload:** Two values for payload length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported payload lengths.*
- ***Nonces:** All supported nonce lengths (e.g., 8, 9, 10, 11, 12, 13) in bytes.*
- ***Tag:** All supported tag lengths (e.g., 4, 6, 8, 10, 12, 14, 16) in bytes.*

The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

Variable Associated Data Test: For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Payload Text: For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Nonce Test: For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Tag Test: For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Decryption-Verification Process Test: To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

***AES-GCM:** These tests are intended to be equivalent to those described in the NIST document, "The Galois/Counter Mode (GCM) and GMAC Validation System (GCMVS) with the Addition of XPN Validation Testing," rev. 15 Jun 2016, section 6.2, found at http://csrc.nist.gov/groups/STM/cavp/documents/mac/gcmvs.pdf.*

It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/gcmtestvectors.zip, or use values not generated expressly to exercise the AES-GCM implementation.

The evaluator shall test the authenticated encryption functionality of AES-GCM by supplying 15 sets of Key, Plaintext, AAD, IV, and Tag data for every combination of the following parameters as selected in the ST and supported by the implementation under test:

- ***Key size in bits:** Each selected and supported key size (e.g., 128, 192, or 256 bits).*
- ***Plaintext length in bits:** Up to four values for plaintext length: Two values that are non-zero integer multiples of 128, if supported. And two values that are non-multiples of 128, if supported.*
- ***AAD length in bits:** Up to five values for AAD length: Zero-length, if supported. Two*

values that are non-zero integer multiples of 128, if supported. And two values that are integer non-multiples of 128, if supported.

- ***IV length in bits:*** *Up to three values for IV length: 96 bits. Minimum and maximum supported lengths, if different.*
- ***MAC length in bits:*** *Each supported length (e.g., 128, 120, 112, 104, 96).*

To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known-good implementation.

The evaluator shall test the authenticated decrypt functionality of AES-GCM by supplying 15 Ciphertext-Tag pairs for every combination of the above parameters, replacing Plaintext length with Ciphertext length. 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. To determine correctness, the evaluator shall compare the resulting pass/fail status and Plaintext values to the results obtained by submitting the same inputs to a known-good implementation.

AES-CTR:

For the AES-CTR tests described below, the plaintext and ciphertext values shall consist of 128-bit blocks. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known-good implementation.

These tests are intended to be equivalent to those described in NIST's AES Algorithm Validation Suite (AESAVS) (http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf). It is not recommended that evaluators use values obtained from static sources such as the example NIST's AES Known Answer Test Values from the AESAVS document, or use values not generated expressly to exercise the AES-CTR implementation.

AES-CTR Known Answer Tests

KAT-1 (GFSBox): To test the encrypt functionality of AES-CTR, the evaluator shall supply a set of five different plaintext values for each selected key size and obtain the ciphertext value that results from AES-CTR encryption of the given plaintext using a key value of all zeros.

To test the decrypt functionality of AES-CTR, the evaluator shall supply a set of five different ciphertext values for each selected key size and obtain the plaintext value that results from AES-CTR decryption of the given ciphertext using a key value of all zeros.

KAT-2 (KeySBox): To test the encrypt functionality of AES-CTR, the evaluator shall supply a set of five different key values for each selected key size and obtain the ciphertext value that results from AES-CTR encryption of an all-zeros plaintext using the given key value.

To test the decrypt functionality of AES-CTR, the evaluator shall supply a set of five different key values for each selected key size and obtain the plaintext that results from AES-CTR decryption of an all-zeros ciphertext using the given key.

KAT-3 (Variable Key): To test the encrypt functionality of AES-CTR, the evaluator shall supply a set of keys for each selected key size (as described below) and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using each key.

Key i in each set shall have the leftmost i bits set to ones and the remaining bits to zeros, for values of i from 1 to the key size. The keys and corresponding ciphertext are listed in AESAVS, Appendix E.

To test the decrypt functionality of AES-CTR, the evaluator shall use the same keys as above to decrypt the ciphertext results from above. Each decryption should result in an all-zeros plaintext.

KAT-4 (Variable Text): To test the encrypt functionality of AES-CTR, for each selected key size, the evaluator shall supply a set of 128-bit plaintext values (as described below) and obtain the ciphertext values that result from AES-CTR encryption of each plaintext value using a key of each size.

Plaintext value i shall have the leftmost i bits set to ones and the remaining bits set to zeros, for values of i from 1 to 128. The plaintext values are listed in AESAVS, Appendix D.

To test the decrypt functionality of AES-CTR, for each selected key size, use the plaintext values from above as ciphertext input, and AES-CTR decrypt each ciphertext value using key of each size consisting of all zeros.

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting nine i-block messages for each selected key size, for $2 \leq i \leq 10$. For each test, the evaluator shall supply a key and a plaintext message of length i blocks, and encrypt the message using AES-CTR. The resulting ciphertext values shall be compared to the results of encrypting the plaintext messages using a known good implementation.

The evaluator shall test the decrypt functionality by decrypting nine i-block messages for each selected key size, for $2 \leq i \leq 10$. For each test, the evaluator shall supply a key and a ciphertext message of length i blocks, and decrypt the message using AES-CTR. The resulting plaintext values shall be compared to the results of decrypting the ciphertext messages using a known good implementation.

AES-CTR Monte Carlo Tests

The evaluator shall test the encrypt functionality for each selected key size using 100 2-tuples of pseudo-random values for plaintext and keys.

The evaluator shall supply a single 2-tuple of pseudo-random values for each selected key size. This 2-tuple of plaintext and key is provided as input to the below algorithm to generate the remaining 99 2-tuples, and to run each 2-tuple through 1000 iterations of AES-CTR encryption.

```
# Input: PT, Key
Key[0] = Key
PT[0] = PT
for i = 0 to 99 {
        Output Key[i], PT[0]
        for j = 0 to 999 {
                CT[j] = AES-CTR-Encrypt(Key[i], PT[j])
                PT[j+1] = CT[j]
        }
        Output CT[j]
        If (KeySize == 128) Key[i+1] = Key[i] xor CT[j]
        If (KeySize == 192) Key[i+1] = Key[i] xor (last 64 bits of CT[j-1] || CT[j])
        If (KeySize == 256) Key[i+1] = Key[i] xor ((CT[j-1] | CT[j])
        PT[0] = CT[j]
}
```

The ciphertext computed in the 1000th iteration (CT[999]) is the result for each of the 100 2-tuples for each selected key size. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as above, exchanging CT

and PT, and replacing AES-CTR-Encrypt with AES-CTR-Decrypt. 198 Note additional design considerations for this mode are addressed in the KMD requirements.

XTS-AES: *These tests are intended to be equivalent to those described in the NIST document, "The XTS-AES Validation System (XTSVS)," updated 5 Sept 2013, found at http://csrc.nist.gov/groups/STM/cavp/documents/aes/XTSVS.pdf*

It is not recommended that evaluators use values obtained from static sources such as the XTS-AES test vectors at http://csrc.nist.gov/groups/STM/cavp/documents/aes/XTSTestVectors.zip or use values not generated expressly to exercise the XTS-AES implementation.

The evaluator shall generate test values as follows:

For each supported key size (256 bit (for AES-128) and 512 bit (for AES-256) keys), the evaluator shall provide up to five data lengths:

- *Two data lengths divisible by the 128-bit block size, If data unit lengths of complete block sizes are supported.*
- *Two data lengths not divisible by the 128-bit block size, if data unit lengths of partial block sizes are supported.*
- *The largest data length supported by the implementation, or 2^16 (65536), whichever is larger.*

The evaluator shall specify whether the implementation supports tweak values of 128-bit hexadecimal strings or a data unit sequence numbers, or both.

For each combination of key size and data length, the evaluator shall provide 100 sets of input data and obtain the ciphertext that results from XTS-AES encryption. If both kinds of tweak values are supported then each type of tweak value shall be used in half of every 100 sets of input data, for all combinations of key size and data length. The evaluator shall verify that the resulting ciphertext matches the results from submitting the same inputs to a known-good implementation of XTS-AES.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS- AES decrypt.

The evaluator shall check that the full length keys are created by methods that ensure that the two halves are different and independent.

AES-KWP:

The tests below are derived from "The Key Wrap Validation System (KWVS), Updated: June 20, 2014" from the National Institute of Standards and Technology.

The evaluator shall test the authenticated-encryption functionality of AES-KWP (KWP-AE) using the same test as for AES-KW authenticated-encryption with the following change in the five plaintext lengths:

- *Four lengths that are multiples of 8 bits*
- *The largest supported length less than or equal to 4096 bits.*

The evaluator shall test the authenticated-decryption (KWP-AD) functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticatedencryption with AES-KWP authenticated-decryption. For the Authenticated Decryption test, 20 out of the 100 trials per plaintext length have ciphertext values that fail authentication.

Additionally, the evaluator shall perform the following negative tests:

Test 1: (invalid plaintext length):

Determine the valid plaintext lengths of the implementation from the TOE specification. Verify that the implementation of KWP-AE in the TOE rejects plaintexts of invalid length by testing plaintext of the following lengths: 1) plaintext with length greater than 64 semi-blocks, 2) plaintext with bit-length not divisible by 8, and 3) plaintext with length 0.

Test 2: (invalid ciphertext length): *Determine the valid ciphertext lengths of the implementation from the TOE specification. Verify that the implementation of KWP-AD in the TOE rejects ciphertexts of invalid length by testing ciphertext of the following lengths: 1) ciphertext with length greater than 65 semi-blocks, 2) ciphertext with bit-length not divisible by 64, 3) ciphertext with length 0, and 4) ciphertext with length of one semi-block.*

Test 3: (invalid ICV2): *Test that the implementation detects invalid ICV2 values by encrypting any plaintext value four times using a different value for ICV2 each time as follows: Start with a base ICV2 of 0xA65959A6. For each of the four tests change a different byte of ICV2 to a different value, so that each of the four bytes is changed once. Verify that the implementation of KWP-AD in the TOE outputs FAIL for each test.*

Test 4: (invalid padding length): *Generate one ciphertext using algorithm KWP-AE with substring [len(P)/8]32 of S replaced by each of the following 32-bit values, where len(P) is the length of P in bits and [ ]32 denotes the representation of an integer in 32 bits:*

- *[0]32*
- *[len(P)/8-8]32*
- *[len(P)/8+8]32*
- *[513]32.*

Verify that the implementation of KWP-AD in the TOE outputs FAIL on those inputs.

Test 5: (invalid padding bits):

If the implementation supports plaintext of length not a multiple of 64-bits, then

```
                                        for each PAD length [1..7]
                                                for each byte in PAD set a zero PAD value;
                                                        replace current byte by a non-zero value and use the resulting plaintext as
                                                                input to algorithm KWP-AE to generate ciphertexts;
                                                        verify that the implementation of KWP-AD in the
TOE outputs FAIL on
                                                                this input.
```

AES-KW:

The tests below are derived from "The Key Wrap Validation System (KWVS), Updated: June 20, 2014" from the National Institute of Standards and Technology.

The evaluator shall test the authenticated-encryption functionality of AES-KW for each combination of the following input parameters:

- *Supported key lengths selected in the ST (e.g. 128 bits, 256 bits)*
- *Five plaintext lengths:*
  - *Two lengths that are non-zero multiples of 128 bits (two semi-block lengths)*
  - *Two lengths that are odd multiples of the semi-block length (64 bits)*
  - *The largest supported plaintext length less than or equal to 4096 bits.*

For each set of the above parameters the evaluator shall generate a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To

determine correctness, the evaluator shall compare the results with those obtained from the AES-KW authenticated-encryption function of a known good implementation.

The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption (KW-AE) with AES-KW authenticated-decryption (KW-AD). For the authenticated-decryption test, 20 out of the 100 trials per plaintext length must have ciphertext values that are not authentic; that is, they fail authentication.

Additionally, the evaluator shall perform the following negative tests:

Test 1 (invalid plaintext length):

Determine the valid plaintext lengths of the implementation from the TOE specification. Verify that the implementation of KW-AES in the TOE rejects plaintexts of invalid length by testing plaintext of the following lengths: 1) plaintext length greater than 64 semi-blocks, 2) plaintext bit-length not divisible by 64, 3) plaintext with length 0, and 4) plaintext with one semi-block.

Test 2 (invalid ciphertext length):

Determine the valid ciphertext lengths of the implementation from the TOE specification. Verify that the implementation of KW-AD in the TOE rejects ciphertexts of invalid length by testing ciphertext of the following lengths: 1) ciphertext with length greater than 65 semi-blocks, 2) ciphertext with bit-length not divisible by 64, 3) ciphertext with length 0, 4) ciphertext with length of one semiblock, and 5) ciphertext with length of two semi-blocks.

Test 3 (invalid ICV1):

Test that the implementation detects invalid ICV1 values by encrypting any plaintext value eight times using a different value for ICV1 each time as follows: Start with a base ICV1 of 0xA6A6A6A6A6A6A6A6. For each of the eight tests change a different byte to a different value, so that each of the eight bytes is changed once. Verify that the implementation of KW-AD in the TOE outputs FAIL for each test.

FCS_HTTPS_EXT.1 HTTPS Protocol

The inclusion of this selection-based component depends upon selection in FIA_X509_EXT.2.1, 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 or if "HTTPS" is selected in FIA_X509_EXT.2.

If this SFR is included in the ST, then the *Functional Package for Transport Layer Security* must also be included.

The ST author must provide enough detail to determine how the implementation is complying with the standard(s) identified; this can be done either by adding elements to this component, or by additional detail in the TSS.

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.
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 FIA_X509_EXT.2.1, 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 is included in the ST if the ST Author selected "IPsec" in FTP_ITC_EXT.1.1.

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

While there must be a means to order the rules, a general approach to ordering is not mandated, as long as the 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 shall select tunnel mode. If the TOE uses IPsec to establish an end-to-end connection to another IPsec VPN Client, the ST author shall 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 shall 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-128, AES-GCM-256 (as specified in RFC 4106), [**selection**: *AES-CBC-128 (specified in RFC 3602), AES-CBC-256 (specified in RFC 3602), no other algorithms*]] together with a Secure Hash Algorithm (SHA)-based HMAC.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol:

[**selection**:

- *IKEv1, using Main Mode for Phase 1 exchanges, as defined in RFC 2407, RFC 2408, RFC 2409, RFC 4109, [**selection**: no other RFCs for extended sequence numbers, RFC 4304 for extended sequence numbers], [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions], and [**selection**: support for XAUTH, no support for XAUTH]* ,
- *IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8784, RFC 8247, and [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions].*

]

Application Note: If the TOE implements SHA-2 hash algorithms for IKEv1 or IKEv2, the ST author shall select RFC 4868.

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the [**selection**: *IKEv1*, *IKEv2*] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [**selection**: *AES-GCM-128 as specified in RFC 5282*, *AES-GCM-256 as specified in RFC 5282*, *no other algorithm*].

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [**selection**:

- *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]* ,
- *IKEv1 SA lifetimes can be configured by [**selection**: an Administrator, a VPN Gateway] based on [**selection**: number of packets/number of bytes, length of time]* ,
- *IKEv1 SA lifetimes are fixed based on [**selection**: number of packets/number of bytes, length of time]. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.*

]

Application Note: The ST author is afforded a selection based on the version of IKE in their implementation. There is a further selection within this 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.

The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5. The IKEv1 requirement can be accomplished either by providing Authorized Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE), or by “hard coding” the limits in the implementation. For IKEv2, there are no hardcoded limits, but in this case 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 [19 (256-bit Random ECP), 20 (384-bit Random ECP), and [**selection**: *24 (2048-bit MODP with 256-bit POS), 15 (3072-bit MODP), 14 (2048-bit MODP), no other DH groups*]].

Application Note: The selection is used to specify additional DH groups supported. This applies to IKEv1 and 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_EXT.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**: *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-Module; one or more of the public key schemes is chosen by the ST author to reflect what is implemented. The ST author also ensures that appropriate FCS requirements reflecting the algorithms used (and key generation capabilities, if provided) are listed to support those methods. Note that the TSS will elaborate on the way in which these algorithms are to be used (for example, 2409 specifies three authentication methods using public keys; each one supported will be described in the TSS).

If "pre-shared keys" is selected, the selection-based requirement FIA_PSK_EXT.1 must be claimed.

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**: *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.

The configuration of the peer reference identifier is addressed by FMT_SMF.1.1/VPN.

FCS_IPSEC_EXT.1.14

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

Application Note: If this functionality is configurable, the TSF may be configured by a VPN Gateway or by an Administrator of the TOE itself.

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

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 Operational Guidance 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 operational guidance 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

Note that the evaluator shall perform all tests using the Virtualization System and a representative sample of platforms listed in the ST (for TOEs that claim to support multiple platforms).

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 (e.g. is the IPsec implementation an integrated part of the VS or is it a standalone executable that is bundled into the VS).

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 operational guidance 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 operational guidance is consistent with the description in the TSS, and that the level of detail in the operational guidance 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 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 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 am IPsec VPN can be established to operate in tunnel mode or transport mode (as selected).

Guidance
The evaluator shall confirm that the operational guidance 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 operational guidance 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 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 2:*** *(conditional): If transport mode is selectted, 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
If both transport mode and tunnel mode are implemented, the evaluator shall review the operational guidance to determine how the use of a given mode is specified.

Guidance
The evaluator shall check that the operational guidance provides instructions on how to construct or acquire the SPD and uses the guidance to configure the TOE for the following test.

Tests
The evaluator shall perform the following test:

- ***Test 1:*** *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 algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the "ST" author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1/KeyHash Cryptographic Operations (Keyed Hash Algorithms).

Guidance
The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Tests

- ***Test 1:*** *The evaluator shall configure the TOE/platform as indicated in the operational guidance configuring the TOE/platform to use each of the supported algorithms, 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 IKEv1 and/or IKEv2 are implemented. If IKEv1 is implemented, the evaluator shall verify that the TSS indicates whether or not XAUTH is supported, and that aggressive mode is not used for IKEv1 Phase 1 exchanges (i.e. only main mode is used). It may be that these are configurable options.

Guidance
The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the test below. If XAUTH is implemented, the evaluator shall verify that the operational guidance provides instructions on how it is enabled or disabled.

If the TOE supports IKEv1, the evaluator shall verify that the operational guidance either asserts that only main mode is used for Phase 1 exchanges, or provides instructions for disabling aggressive mode.

Tests
Tests are performed in conjunction with the other IPsec evaluation activities with the exception of the activities below:

- ***Test 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. If the TOE supports IKEv1 with or without XAUTH, the evaluator shall verify that this test can be successfully repeated with XAUTH enabled and disabled in the manner specified by the operational guidance. If the TOE only supports IKEv1 with XAUTH, the evaluator shall verify that connections not using XAUTH are unsuccessful. If the TOE only supports IKEv1 without XAUTH, the evaluator shall verify that connections using XAUTH are unsuccessful.*
- ***Test 2:*** *(conditional) If the TOE supports IKEv1, the evaluator shall perform any applicable operational guidance steps to disable the use of aggressive mode and then attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator shall show that the TOE will reject a VPN gateway from initiating an IKEv1 Phase 1 connection in aggressive mode. The evaluator should then show that main mode exchanges are supported.*

FCS_IPSEC_EXT.1.6

TSS
The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms AES-CBC-128, AES-CBC-256 are specified, and if others are chosen in the selection of the requirement, those are included in the TSS discussion.

Guidance
The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct 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:

- ***Test 1:*** *The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or 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 TSS EAs for this requirement.

Guidance
The evaluator shall check the operational 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 operational guidance 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. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. 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 for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

- ***Test 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 2:*** *(Conditional): The evaluator shall construct a test where a Phase 1 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 3:*** *[conditional] The evaluator shall perform a test similar to Test 2 for Phase 2 SAs, except that the lifetime will be 8 hours or less instead of 24 hours or less.*
- ***Test 4:*** *[conditional]: If a fixed limit for IKEv1 SAs is supported, the evaluator shall establish an SA and observe that the connection is closed after the fixed traffic and/or time value is reached.*

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 AGD EAs for this requirement.
Tests
The evaluator shall perform the following test:

- ***Test 1:*** *For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols 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 AGD EAs for this requirement.

Tests
There are no test EAs for this requirement.

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 operational guidance describes how pre-shared keys are to be generated and established.

The evaluator ensures the operational guidance 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 operational guidance 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 operational guidance 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 FIA_X509_EXT.2 and FIA_X509_EXT.3 (for IPsec connections and depending on the Base-PP), 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 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 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 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, and 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 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 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 IKE authentication succeeds.*
- ***Test 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 IKE authentication fails.*

  The following tests are conditional:
- ***Test 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 IKE authentication fails.*
- ***Test 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 IKE 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 IKE authentication fails. To demonstrate a comparison of DN values, the evaluator shall change any one of the four DN values and verify that the IKE authentication fails.*
- ***Test 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 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_IPSEC_EXT.1.14
TSS
The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or

IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.
Guidance
There are no AGD EAs for this requirement.
Tests
The evaluator follows the guidance to configure the TOE to perform the following tests:

- ***Test 1:*** *This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.*
- ***Test 2:*** *[conditional] This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.*
- ***Test 3:*** *This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.*
- ***Test 4:*** *: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.*

FCS_RBG_EXT.1 Random Bit Generation

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

FCS_RBG_EXT.1.1

The TSF shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using [**selection**: *Hash_DRBG (any)*, *HMAC_DRBG (any)*, *CTR_DRBG (AES)*].

FCS_RBG_EXT.1.2

The deterministic RBG shall be seeded by at least one entropy source in accordance with NIST SP 800-90B that accumulates entropy from [**selection**: *[**assignment**: number of software-based sources] software-based noise source*, *[**assignment**: number of hardware-based sources] hardware-based noise source*] with a minimum of [**selection**: *128*, *192*, *256*] bits of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011, of the keys and CSPs that it will generate.

Application Note: 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.

Specifically, this SFR must be included in the ST if "RSK" is selected in FCS_CKM.1/SK.

This SFR is also needed if the following SFRs are included in the ST: FCS_IPSEC_EXT.1, FCS_SLT_EXT.1, FCS_CKM.1/AK, and FCS_COP.1/SigGen.

For the selection in FCS_RBG_EXT.1.2, the ST author selects the appropriate number of bits of entropy that corresponds to the greatest security strength of the algorithms included in the ST. Security strength is defined in Tables 2 and 3 of NIST SP 800-57A. For example, if the implementation includes 2048-bit RSA (security strength of 112 bits), AES 128 (security strength 128 bits), and HMAC-SHA-256 (security strength 256 bits), then the ST author would select 256 bits.

FCS_RBG_EXT.1.3

The TSF shall be capable of providing output of the RBG to tenant software on the TOE that request random bits.

Application Note: ISO/IEC 18031:2011 contains three different methods of generating random numbers. Each of these in turn depends on underlying cryptographic primitives (hash functions/ciphers). This cPP allows SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 for Hash_DRBG or HMAC_DRBG and only AES-based implementations for CTR_DRBG.

This requirement must be included in the ST if the TOE generates keys, signatures, or Salts for its own use or for tenant software (FCS_CKM.1/AK, FCS_CKM.1/SK, FCS_COP.1/SigGen, FCS_SKT_EXT.1) or if it provides RNG services for tenant software (FCS_RBG_EXT.1).

Evaluation Activities

FCS_RBG_EXT.1
TSS
The evaluator shall examine the TSS to determine that it specifies the DRBG type, identifies the entropy sources seeding the DRBG, and state the assumed or calculated min-entropy supplied either separately by each source or the min-entropy contained in the combined seed value.

In addition to the materials below, documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [DSCcPP].

Guidance
There are no AGD evaluation activities for this component.
KMD
There are no KMD 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 evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are

additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

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

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

- ***Entropy input:** the length of the entropy input value must equal the seed length.*
- ***Nonce:** If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.*
- ***Personalization string:** The length of the personalization string must be ≤ seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.*
- ***Additional input:** the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.*

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 encryption methods as specified in FCS_COP.1/KeyEnc.

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 AGD evaluation activities for this component.
KMD
There are no KMD evaluation activities for this component
Tests
There are no test evaluation 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**:

- *Symmetric encryption in [**selection**: AES_CCM, AES_GCM, AES_KW, AES_KWP] mode in accordance with FCS_COP.1/SKC,*
- *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*

].

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. 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 is not required to be part of the TSS – it can be submitted as a separate document and marked as developer proprietary.

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 AGD evaluation activities for this component.
KMD
There are no KMD evaluation activities for this component.
Tests
There are no test evaluation activities for this component.

B.4 Identification and Authentication (FIA)

FIA_AFL_EXT.1 Authentication Failure Handling

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4, FPT_TUD_EXT.4.2.

FIA_AFL_EXT.1.1

The TSF shall consider password and [*no other*] as critical authentication mechanisms.

Application Note: If no additional authentication mechanisms are selected in FIA_UAU.5.1, then 'no other' must be selected. If an additional authentication mechanism is selected in FIA_UAU.5.1, then it must only be selected in FIA_AFL_EXT.1.1 if surpassing the authentication failure threshold for biometric data causes a countermeasure to be triggered regardless of the failure status of the other authentication mechanisms.

If the TOE implements multiple Authentication Factor interfaces (for example, a DAR decryption interface, a lockscreen interface, an auxiliary boot mode interface), this component 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 integer within [**assignment**: *range of acceptable values for each authentication mechanism*] of [**selection**: *unique*, *non-unique*] unsuccessful authentication attempts occur related to last successful authentication for each authentication mechanism.

Application Note: The positive integer(s) is configured according to Table 19 in FMT_SMF_EXT.1.1.

An unique authentication attempt is defined as any attempt to verify a password 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 if 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 component 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 in regards 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 component 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.

Application Note: 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 device. 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 will be limited to other available authentication mechanisms, unless the given mechanism is designated as a critical authentication mechanism.

Application Note: 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 single critical authentication mechanism has been surpassed, the TSF shall perform a wipe of all protected data.

Application Note: Wipe is performed in accordance with FCS_CKM_EXT.4. 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 component 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 guidance describes how the administrator configures the maximum number of unique unsuccessful authentication attempts.
Tests
The evaluator shall configure the device with all authentication mechanisms selected in FIA_UAU.5.1. The evaluator shall perform the following tests for each available authentication interface:

- ***Test 1:*** *The evaluator shall configure the TOE, according to the AGD guidance, with a maximum number of unsuccessful authentication attempts. The evaluator shall enter the locked state and enter incorrect passwords until the wipe occurs. The evaluator shall verify that the number of password entries corresponds to the configured maximum and that the wipe is implemented.*
- ***Test 2:*** *[conditional] If the TOE supports multiple authentication mechanisms the previous test shall be repeated using a combination of authentication mechanisms confirming that the critical authentication mechanisms will cause the device to wipe and that when the maximum number of unsuccessful authentication attempts for a non-critical authentication mechanism is exceeded, the device limits authentication attempts to other available authentication mechanisms. If multiple authentication mechanisms utilize a shared counter, then the evaluator shall verify that the maximum number of unsuccessful authentication attempts can be reached by using each individual authentication mechanism and a combination of all authentication mechanisms that share the counter.*

FIA_PMG_EXT.1 Password Management

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4, FPT_TUD_EXT.4.2.

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 up to [**assignment**: *an integer greater than or equal to 14*] characters shall be supported.

Application Note: While some corporate policies require passwords of 14 characters or better, the use of a REK 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 TSS evaluation activities for this component.
Guidance
The evaluator shall examine the operational guidance 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

- ***Test 1:*** *The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify*

the subset of those characters chosen for testing.

FIA_UAU.5 Multiple Authentication Mechanisms

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4, FPT_TUD_EXT.4.2.

FIA_UAU.5.1

The TSF shall provide password and [**selection**: *X.509 certificate-based authentication, SSH certificate-based authentication, biometric authentication, hybrid authentication, no other authentication mechanism*] to support user authentication.

Application Note: The TSF must support a Password Authentication Factor and may optionally implement a biometric authentication factor. A hybrid authentication factor is where a user has to submit a combination of PIN/password and biometric sample where both have to pass and if either fails the user is not made aware of which factor failed.

If "hybrid" is selected, a biometric modality does not need to be selected, but should be selected if the biometric authentication can be used independent of the hybrid authentication, i.e. without having to enter a PIN/password.

The Password Authentication Factor is configured according to FIA_PMG_EXT.1.

FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the [**assignment**: *rules describing how each authentication mechanism selected in FIA_UAU.5.1 provides 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 BAFs 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 guidance for each authentication mechanism is addressed in the AGD guidance.
Tests
- ***Test 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 2:*** *For each authentication mechanism rule, the evaluator shall ensure that the authentication mechanism(s) behave accordingly.*

FIA_UAU.7 Protected Authentication Feedback

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4, FPT_TUD_EXT.4.2.

FIA_UAU.7.1

The TSF shall provide only obscured feedback to the device's display to the user while the authentication is in progress.

Application Note: This applies to all authentication methods specified in FIA_UAU.5.1. The TSF may briefly (1 second or less) display each character or provide an option to allow the user to unmask the password; however, the password 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 and/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.

Evaluation Activities

FIA_UAU.7
TSS
The evaluator shall ensure that the TSS describes the means of obscuring the authentication entry, 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 the password is obscured by default.
Tests

- ***Test 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 2:*** *[conditional] For each 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.*

FIA_UIA_EXT.1 Administrator Identification and Authentication

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4, FPT_TUD_EXT.4.2.

FIA_UIA_EXT.1.1

The TSF shall require Administrators to be successfully identified and 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: 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 19) in FMT_SMF_EXT.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 operational guidance 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 operational guidance 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 operational guidance provides sufficient instruction on limiting the allowed services.

FIA_X509_EXT.1 X.509 Certificate Validation

The inclusion of this selection-based component depends upon selection in FTP_ITC_EXT.1.1.

FIA_X509_EXT.1.1

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

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a trusted certificate
- The TOE shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field
- The TSF shall validate revocation status of the certificate using [**selection**: *OCSP as specified in RFC 6960, a CRL as specified in RFC 5759, an OCSP TLS Status Request Extension (OCSP stapling) as specified in RFC 6066, OCSP TLS Multi-Certificate Status Request Extension (i.e., OCSP Multi-Stapling) as specified in RFC 6961*].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing Purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.

Application Note: This SFR must be included in the ST if the selection for FPT_TUD_EXT.1.3 is "digital signature mechanism," if "certificate-based authentication of the remote peer" is selected in FTP_ITC_EXT.1.1, or if "authentication based on X.509 certificates" is selected in FIA_UAU.5.1.

FIA_X509_EXT.1.1 lists the rules for validating certificates. The ST author shall select whether revocation status is verified using OCSP or CRLs. FIA_X509_EXT.2 requires that certificates are used for IPsec; this use requires that the extendedKeyUsage rules are verified. Certificates may optionally be used for SSH, TLS, and HTTPs and, if implemented, must be validated to contain the corresponding extendedKeyUsage.

OCSP stapling and OCSP multi-stapling support only TLS server certificate validation. If other certificate types are validated, either OCSP or CRL must be claimed. If OCSP is not supported the EKU provision for checking the OCSP

Signing purpose is met by default.

Regardless of the selection of TSF or TOE platform, the validation must result in a trusted root CA certificate in a root store managed by the platform.

OCSP responses are signed using either the certificate's issuer's CA certificate or an OCSP certificate issued to an OCSP responder delegated by that issuer to sign OCSP responses. A compliant TOE is able to validate OCSP responses in either case, but the OCSP signing extended key usage purpose is only required to be checked in OCSP certificates.

FIA_X509_EXT.1.2

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

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

Evaluation Activities

FIA_X509_EXT.1
TSS
The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.
Tests
The tests described must be performed in conjunction with the other Certificate Services evaluation activities, including the uses listed in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

- ***Test 1:*** *The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:*
  - *by establishing a certificate path in which one of the issuing certificates is not a CA certificate,*
  - *by omitting the basicConstraints field in one of the issuing certificates,*
  - *by setting the basicConstraints field in an issuing certificate to have CA=False,*
  - *by omitting the CA signing bit of the key usage field in an issuing certificate, and*
  - *by setting the path length field of a valid CA field to a value strictly less than the certificate path.*

  The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.
- ***Test 2:*** *The evaluator shall demonstrate that validating an expired certificate results in the function failing.*
- ***Test 3:*** *The evaluator shall test that the TOE can properly handle revoked certificates – conditional on whether CRL, OCSP, OCSP stapling, or OCSP multi-stapling is selected; if multiple methods are selected, and then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions 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 validation function succeeds. The evaluator shall then attempt the test with a certificate that will be revoked (for each method chosen in the selection) and verify that the validation function fails.*
- ***Test 4:*** *If any OCSP option is selected, the evaluator shall present a delegated OCSP certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set and verify that validation of the CRL fails.*
- ***Test 5:*** *(Conditional on support for EC certificates as indicated in FCS_COP.1/SIG). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain..*
- ***Test 6:*** *(Conditional on support for EC certificates as indicated in FCS_COP.1/SIG). The evaluator shall replace the intermediate certificate in the certificate chain for Test 5 with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 5, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.*

FIA_X509_EXT.2 X.509 Certificate Authentication

The inclusion of this selection-based component depends upon selection in FTP_ITC_EXT.1.1.

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [**selection**: *IPsec*, *TLS*, *HTTPS*, *SSH*, *code signing for system software updates*, *[**assignment**: other uses]*]

Application Note: This SFR must be included in the ST if the selection for FPT_TUD_EXT.1.3 is "digital signature mechanism," if "certificate-based authentication of the remote peer" is selected in FTP_ITC_EXT.1, or if "authentication based on X.509 certificates" is selected in FIA_UAU.5.1.

This SFR must also be included in the ST if X.509 certificate-based authentication is used for "other uses" as listed in the assignment in FIA_X509_EXT.2.1.

FIA_X509_EXT.2.2

When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [**selection**: *allow the administrator to choose whether*

to accept the certificate in these cases, accept the certificate, not accept the certificate].

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

Evaluation Activities

FIA_X509_EXT.2
TSS
The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.
Tests
The evaluator shall perform Test 1 for each function listed in FIA_X509_EXT.2.1 that requires the use of certificates:

- ***Test 1:*** *The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.*
- ***Test 2:*** *The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.*

B.5 Security Management (FMT)

FMT_MOF_EXT.1 Management of Security Functions Behavior

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4.

FMT_MOF_EXT.1.1

The TSF shall be capable of supporting [**selection**: *local*, *remote*] administration.

FMT_MOF_EXT.1.2

The TSF shall restrict the ability to perform the functions in column 3 of Table 19 to an authenticated administrator.

Application Note: There are three roles defined in this PP for general-purpose computing platforms: 1) Administrator, 2) User, and 3) tenant software. Only Administrators can perform management functions on the platform, and only Administrators authenticate themselves to the platform.

This SFR should be included in the ST if the platform implements the Administrator role, as consistent with several selections thoughout the PP.

The ST author may not select the same function in both FMT_MOF_EXT.1.1 and FMT_MOF_EXT.1.2.

The ST author should select those security management functions that the administrator may exercise.

For functions that are mandatory, any sub-functions not in a selection are also mandatory and any assignments must contain at least one assigned value. For non-selectable sub-functions in an optional function, all sub-functions outside the selection must be implemented in order for the function to be listed.

If the ST author selects "remote" then FTP_ITC_EXT.1 must be included in the ST.

Evaluation Activities

FMT_MOF_EXT.1
TSS
The evaluator shall verify that the TSS describes those management functions that may be performed by the Administrator, to include how the user is prevented from accessing, performing, or relaxing the function (if applicable). 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_EXT.1.

Guidance
There are no guidance evaluation activities for this component.

Tests

- ***Test 1:*** *The evaluator shall use the test environment to deploy policies to Mobile Devices.*
- ***Test 2:*** *The evaluator shall create policies which collectively include all management functions which are controlled by the (enterprise) administrator and cannot be*

overridden/relaxed by the user as defined in FMT_MOF_EXT.1.2. The evaluator shall apply these policies to devices, attempt to override/relax each setting both as the user (if a setting is available) and as an application (if an API is available), and ensure that the TSF does not permit it. Note that the user may still apply a more restrictive policy than that of the administrator.

- ***Test 3:*** *Additional testing of functions provided to the administrator are performed in conjunction with the testing activities for FMT_SMF_EXT.1.1.*

FMT_SMF_EXT.1 Specification of Management Functions

The inclusion of this selection-based component depends upon selection in FPT_ROT_EXT.2.2, FPT_ROT_EXT.3.2, FPT_TUD_EXT.2.5, FPT_TUD_EXT.3.4.

FMT_SMF_EXT.1.1

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

Table 16: Management Functions

Status Markers:
M - Mandatory
O - Optional/Objective

M = Mandatory (TOE must provide that function to that role)
O = Optional (TOE may or may not provide that function to that role)
S = Selection-Based (TOE must provide that function to that role if the TOE claims a particular selection-based SFR)

Number	Function	Admin	Notes
1	Ability to configure and manage the audit functionality and audit data	X	See FAU_GEN.1.
2	Ability to configure name/address of audit/logging server to which to send audit/logging records	X	See FAU_STG_EXT.1.
3	Ability to review audit records.	X	See FAU_SAR.1.
4	Ability to configure the cryptographic functionality.	X	See FTP_ITC_EXT.1.
5	Ability to initiate a trusted channel for remote administration.	X	See FTP_TRP.1.
6	Ability to set parameters for allowable number of authentication failures.	X	See FIA_AFL_EXT.1.
7	Ability to configure password length and complexity.	X	See FIA_PMG_EXT.1.
8	Ability to configure authentication throttling policy.	X	See FIA_TRT_EXT.1.
9	Ability to manage authentication methods and change default authorization factors	X	See FIA_PMG_EXT.1
10	Ability to configure of certificate revocation checking methods.	X	See FIA_X509_EXT.1.
11	Ability to configure TSF behavior when certificate revocation status cannot be determined.	X	See FIA_X509_EXT.2.
12	Ability to configure action to take on integrity failure.	X	See FPT_ROT_EXT.2, FPT_ROT_EXT.3, and FPT_RVR_EXT.1.
13	Ability to initiate the update process.	X	See FPT_TUD_EXT.1.
14	Ability to action to take on update failure.	X	See FPT_TUD_EXT.2 and FPT_TUD_EXT.3.
15	Ability to manage import and export keys/secrets to and from protected storage.	X	See FCS_STG_EXT.1.

]

Application Note: Administration is considered “local” if the Administrator is physically present at the machine on which the VS is installed.

Administration is considered “remote” if communications between the Administrator and the Management Subsystem travel on a network.

The first column lists the management function number. The second column lists the management functions identified in the PP.

In the following columns:

- ‘M’ means Mandatory
- ‘O’ means Optional/Objective
- ‘S’ means Selection-based

Evaluation Activities

FMT_SMF_EXT.1
TSS
The evaluator shall examine the TSS and Operational Guidance to ensure that it describes which security management functions require Administrator privilege and the actions associated with each management function. The evaluator shall verify that for each management function and role specified in Table 19 the defined role is able to perform all mandatory functions as well as all optional or selection-based functions claimed in the ST.

Guidance
The evaluator shall examine the Operational Guidance to ensure that it describes how the Administrator and User are able to perform each management function that the ST claims the TOE supports.

The evaluator shall verify for each claimed management function that the Operational Guidance is sufficiently detailed to allow the function to be performed and that the function can be performed.

Tests
The evaluator shall test each management function for each role listed in Table 19 in the ST to demonstrate that the function can be performed by the roles that are authorized to do so and the result of the function is demonstrated. The evaluator shall also verify for each claimed management function in Table 19 that if the TOE claims not to provide a particular role with access to the function, then it is not possible to access the TOE as that role and perform that function.

B.6 Class: Protection of the TSF (FPT)

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_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,
- 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 included 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 guidance to ensure that is describes how to configure the conditions under which the recovery mechanism is initiated (if configurable).
Tests
The evaluators shall perform the following tests:

- ***Test 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 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_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: 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 make the selection above 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 Administrator/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,
- Notify an [**selection**: *administrator*, *user*] by [**selection**: *generating an audit event*, [***assignment**: notification method*]]

, and [**selection**:

- *Continue execution*,
- *Halt*,
- *Stop all execution and shut down*,
- *Initiate recovery as specified in FPT_RVR_EXT.1*

] [**selection**:

- *automatically*,
- *in accordance with administrator-configurable policy*,
- *by express determination of an [**selection**: administrator, user]*

].

Application Note: If "administrator" is selected anywhere 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.4, and FAU_STG_EXT.1 must be included in the ST.

If "*in accordance with administrator-configurable policy*" is selected then FMT_MOF_EXT.1 and FMT_SMF_EXT.1 must be included in the ST.

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 shall hash the public key provided with the update image, and ensure 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 shall make the appropriate selections from the audit events table (Table 6).

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 is an update image fails authentication.

Guidance
The evaluator shall examine the operational guidance to ensure that it describes the process for updating the platform firmware.

The evaluator shall examine the operational guidance 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

- ***Test 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 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 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 4:*** *The evaluator performs the same test, this time using a valid update image that is not signed. The update must fail.*
- ***Test 5:*** *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 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.

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:

- Notify Notify an [**selection**: *administrator*, *user*] by [**selection**: *generating an audit event*, ***[assignment****: notification method]*]

and [**selection**:

- *Halt*,
- *Stop all execution and shut down*,
- *Initiate a recovery process as specified in FPT_RVR_EXT.1*

] [**selection**:

- *automatically*,
- *in accordance with administrator-configurable policy*,
- *by express determination of an [**selection**: administrator, user]*

].

Application Note: If "*administrator*" is selected anywhere 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.4, and FAU_STG_EXT.1 must be included in the ST.

If "*in accordance with administrator-configurable policy*" is selected then FMT_MOF_EXT.1 and FMT_SMF_EXT.1 must be included in the ST.

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 implement rollback prevention, this check must be made before the newly installed firmware is executed.

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 operational guidance to ensure that it describes the process for updating the platform firmware.

The evaluator shall examine the operational guidance 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
- ***Test 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 2:*** *The evaluator performs the same test, this time using a inauthentic update image. The update code must fail to execute.*
- ***Test 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 4:*** *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

The [**selection**: *Administrator*, *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: The requirement included in the ST if "the secure local update mechanism described in FPT_TUD_EXT.4" is selected in FPT_RVR_EXT.1 or "implement a secure local platform firmware update mechanism described in FPT_TUD_EXT.4" is selected in FPT_TUD_EXT.1.

If "*Administrator*" 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). 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 an Administrator/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--that is no 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 Administrator/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 operational guidance to ensure that it describes instructions for using the local update mechanism, and how to validate that the update was succesful.

Tests

- ***Test 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 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 Trusted Path/Channels FTP)

FTP_ITC_EXT.1 Trusted Channel Communication

The inclusion of this selection-based component depends upon selection in FAU_STG_EXT.1.1, FMT_MOF_EXT.1.1.

FTP_ITC_EXT.1.1

The TSF shall use [**selection**:

- *TLS as conforming to the Functional Package for Transport Layer Security,*
- *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*

] and [**selection**:

- *X.509 certificate-based authentication of the remote peer,*
- *non-certificate-based authentication of the remote peer,*
- *no authentication of the remote peer*

] protocols to provide a communication channel between itself and [**selection**:

- *audit servers (as required by FAU_STG_EXT.1),*
- *remote administrators (as required by FTP_TRP.1.1 if selected in FMT_MOF_EXT.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_EXT.1 or "remote" is selected in FMT_MOF_EXT.1.

If the ST author selects either TLS or HTTPS, the TSF shall 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. The TLS Package requires that the X509 requirements be included by the PP, so selection of TLS or HTTPS causes FIA_X509_EXT.* to be selected.

If the ST author selects SSH, the TSF shall be validated against the Extended Package for Secure Shell.

If the ST author selects "certificate-based authentication of the remote peer," then FIA_X509_EXT.1 and FIA_X509_EXT.2 must be included in the ST. "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 entities and/or remote users used for the channel as well as the protocol that is used for each.
Guidance
The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to access points, VPN Gateways, and other trusted IT products.

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_TRP.1 Trusted Path

The inclusion of this selection-based component depends upon selection in FMT_MOF_EXT.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 "remote" is selected in FMT_MOF_EXT.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 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 operational guidance contains instructions for establishing the remote administrative sessions for each supported method.
Tests
The evaluator shall also perform the following tests:

- ***Test 1:*** *The evaluators shall ensure that communications using each specified (in the operational guidance) remote administration method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 2:*** *For each method of remote administration supported, the evaluator shall follow the operational guidance 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 3:*** *The evaluator shall ensure, for each method of remote administration, the channel data is not sent in plaintext.*
- ***Test 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-Module.

C.1 Extended Components Table

All extended components specified in the PP-Module are listed in this table:

Table 17: Extended Component Definitions

Functional Class	Functional Components
Security Audit (FAU)	FAU_STG_EXT Off-Loading of Audit Data
Cryptographic Support (FCS)	FCS_CKM_EXT Cryptographic Key Management FCS_ENT_EXT Entropy for Virtual Machines FCS_HTTPS_EXT HTTPS Protocol FCS_IPSEC_EXT IPsec Protocol FCS_RBG_EXT Cryptographic Operation (Random Bit Generation) FCS_STG_EXT Cryptographic Key Storage
User Data Protection (FDP)	FDP_TEE_EXT Trusted Execution Environment
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 FIA_X509_EXT X.509 Certificate
Security Management (FMT)	FMT_CFG_EXT Secure by Default FMT_MOF_EXT Management of Security Function Behavior FMT_SMF_EXT Specification of Security Management Functions
Class: Protection of the TSF (FPT)	FMT_JTA_EXT Debug Port Access FMT_PPF_EXT Protection of Platform Firmware FMT_ROT_EXT Platform Integrity FMT_RVR_EXT Platform Firmware Recovery FMT_TUD_EXT Platform Firmware Update
Trusted Path/Channels FTP)	FTP_ITC_EXT Trusted Channel Communications

C.2 Extended Component Definitions

C.2.1 Class FPT - Class: Protection of the TSF

This PP-Module defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.1.1 FMT_JTA_EXT Debug Port Access

Family Behavior

This family defines requirements for access to debug ports during normal operation.

Component Leveling

C.2.1.2 FMT_ROT_EXT Platform Integrity

Family Behavior

This family defines requirements for platform formware and hardware integrity.

Component Leveling

C.2.1.3 FMT_PPF_EXT Protection of Platform Firmware

Family Behavior

This family defines requirements for protecting platform firmware from unauthorized update.

Component Leveling

C.2.1.4 FMT_RVR_EXT Platform Firmware Recovery

Family Behavior

This family defines requirements for recovering from a firmware integrity failure.

Component Leveling

C.2.1.5 FMT_TUD_EXT Platform Firmware Update

Family Behavior

This family defines requirements for updating platform firmware.

Component Leveling

C.2.2 Class FCS - Cryptographic Support

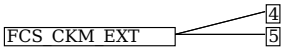
This PP-Module defines the following extended components as part of the FCS class originally defined by CC

C.2.2.1 FCS_CKM_EXT Cryptographic Key Management

Family Behavior

This family defines requirements for management of cryptographic keys.

Component Leveling



FCS_CKM_EXT.4, Cryptographic Key and Key Material Destruction Timing, requires the TSF to destroy keys when no longer used.

FCS_CKM_EXT.5, Cryptographic Key Derivation, requires the TSF to perform key derivation using a defined method.

Management: FCS_CKM_EXT.4

No specific management functions are identified.

Audit: FCS_CKM_EXT.4

There are no auditable events foreseen.

FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction Timing

Hierarchical to: No other components.

Dependencies to: FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1

The TSF shall destroy all keys and keying material when no longer needed.

Management: FCS_CKM_EXT.5

No specific management functions are identified.

Audit: FCS_CKM_EXT.5

There are no auditable events foreseen.

FCS_CKM_EXT.5 Cryptographic Key Derivation

Hierarchical to: No other components.

Dependencies to:

- FCS_CKM.1 Cryptographic Key Generation
- FCS_COP.1 Cryptographic Operation

FCS_CKM_EXT.5.1

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

The rows of Table 18 provide the allowable values for completion of the assignments for FCS_CKM_EXT.5.1.

Table 18: Key Derivation Functions

Identifier	Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KeyDrv1	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Counter Mode using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 (Section 5.1) (KDF in Counter Mode) [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]
KeyDrv2	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Feedback Mode using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 (Section 5.2) (KDF in Feedback Mode) [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]
KeyDrv3	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	KDF in Double Pipeline Iteration Mode using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 (Section 5.3) (KDF in n Double Pipeline Iteration Mode) [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]

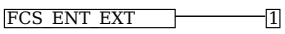
KeyDrv4	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Intermediary keys	[**selection**: *exclusive OR (XOR), SHA-256, SHA-512*]	[**selection**: *128, 192, 256*]bits	[**selection**: *ISO-HASH, FIPS-SHA*]
KeyDrv5	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Concatenated keys	KDF in [**selection**: *Counter Mode, Feedback Mode, Double Pipeline Iteration Mode*] using [**selection**: *CMAC-AES-128, CMAC-AES-192, CMAC-AES-256, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512*]as the PRF	[**selection**: *128, 192, 256*]bits	NIST SP 800-108 [selection: (Section 5.1) (KDF in Counter Mode); (Section 5.2) (KDF in Feedback Mode); (Section 5.3) (KDF in Double-Pipeline Iteration Mode)] [**selection**: *ISO-CMAC, NIST-CMAC, ISO-CIPH, ISO-HMAC, FIPS-HMAC, ISO-HASH, FIPS-SHA*]
KeyDrv6	[**selection**: *symmetric key, initialization vector, authentication token, authorization value, HMAC key, KMAC key*]	Two keys	[**selection**: *AES-CCM, AES-GCM, AES-CBC, AES-KWP, AES-KW, CAM-CBC, CAM-CCM, CAM-GCM*] from FCS_COP.1/SKC Symmetric Key table	[**selection**: *128, 192, 256*]bits	[selection: see List of Standards in FCS_COP.1/SKC Symmetric Key table]
KeyDrv7	[**selection**: *symmetric key, secret IV, seed*]	Shared secret, salt, output length, fixed information	[**selection**: *hash function from FCS_COP.1/Hash, keyed hash from FCS_COP.1/HMAC*]	[**selection**: *128, 192, 256*]bits	(NIST-KDRV) sec 4 [**assignment**: *See List of Standards in FCS_COP.1/Hash and FCS_COP.1/HMAC*]
KeyDrv8	[**selection**: *symmetric key, secret IV, seed*]	Shared secret, salt, IV, output length, fixed information	[**assignment**: *keyed hash from FCS_COP.1/HMAC*]	[**selection**: *128, 192, 256*]bits	(NIST-KDRV) sec 5 [**assignment**: *see List of Standards in FCS_COP.1/Hash and FCS_COP.1/HMAC*]

C.2.2.2 FCS_ENT_EXT Entropy for Virtual Machines

Family Behavior

This family defines requirements for availability of entropy data generated or collected by the TSF.

Component Leveling



FCS_ENT_EXT.1, Entropy for Tenant Software, requires the TSF to provide entropy data to tenant software in a specified manner.

Management: FCS_ENT_EXT.1

No specific management functions are identified.

Audit: FCS_ENT_EXT.1

There are no auditable events foreseen.

FCS_ENT_EXT.1 Entropy for Tenant Software

Hierarchical to: No other components.

Dependencies to: FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

FCS_ENT_EXT.1.1

The TSF shall provide one or more mechanisms to make entropy that meets FCS_RBG_EXT.1 available to tenant software.

C.2.2.3 FCS_HTTPS_EXT HTTPS Protocol

Family Behavior

This family defines requirements for protecting remote management sessions between the TOE and a Security Administrator. This family describes how HTTPS will be implemented.

Component Leveling

FCS_HTTPS_EXT.1, HTTPS Protocol, defines requirements for the implementation of the HTTPS protocol.

Management: FCS_HTTPS_EXT.1

No specific management functions are identified.

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.2.2.4 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.

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_EXT.1 Cryptographic Operation (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 [**selection**: *transport mode*, *tunnel 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-128, AES-GCM-256 (as specified in RFC 4106), [**selection**: *AES-CBC-128 (specified in RFC 3602)*, *AES-CBC-256 (specified in RFC 3602)*, *no other algorithms*]] together with a Secure Hash Algorithm (SHA)-based HMAC.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol:

[**selection**:

- *IKEv1, using Main Mode for Phase 1 exchanges, as defined in RFC 2407, RFC 2408, RFC 2409, RFC 4109, [**selection**: no other RFCs for extended sequence numbers, RFC 4304 for extended sequence numbers], [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions], and [**selection**: support for XAUTH, no support for XAUTH] ,*
- *IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8784, RFC 8247, and [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions].*

]

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the [**selection**: *IKEv1*, *IKEv2*] protocol uses the

cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [**selection**: *AES-GCM-128 as specified in RFC 5282*, *AES-GCM-256 as specified in RFC 5282*, *no other algorithm*].

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [**selection**:

- *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] ,*
- *IKEv1 SA lifetimes can be configured by [**selection**: an Administrator, a VPN Gateway] based on [**selection**: number of packets/number of bytes, length of time] ,*
- *IKEv1 SA lifetimes are fixed based on [**selection**: number of packets/number of bytes, length of time]. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.*

]

FCS_IPSEC_EXT.1.8

The TSF shall ensure that all IKE protocols implement DH groups [19 (256-bit Random ECP), 20 (384-bit Random ECP), and [**selection**: *24 (2048-bit MODP with 256-bit POS)*, *15 (3072-bit MODP)*, *14 (2048-bit MODP)*, *no other 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_EXT.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**: *Pre-shared Keys*, *no other method*].

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**: *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.

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.

FCS_IPSEC_EXT.1.14

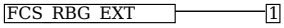
The [**selection**: *TSF*, *VPN Gateway*] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [**selection**: *IKEv1 Phase 1*, *IKEv2 IKE_SA*] connection is greater than or equal to the strength of the symmetric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

C.2.2.5 FCS_RBG_EXT Cryptographic Operation (Random Bit Generation)

Family Behavior

This family defines requirements for random bit/number generation.

Component Leveling



FCS_RBG_EXT.1, Random Bit Generation, requires random bit generation to be performed in accordance with selected standards and seeded by an entropy source.

Management: FCS_RBG_EXT.1

No specific management functions are identified.

Audit: FCS_RBG_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure of the randomization process.

FCS_RBG_EXT.1 Random Bit Generation

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_RBG_EXT.1.1

The TSF shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using [**selection**: *Hash_DRBG (any)*, *HMAC_DRBG (any)*, *CTR_DRBG (AES)*].

FCS_RBG_EXT.1.2

The deterministic RBG shall be seeded by at least one entropy source in accordance with NIST SP 800-90B that accumulates entropy from [**selection**: *[**assignment**: number of software-based sources] software-based noise source*, *[**assignment**: number of hardware-based sources] hardware-based noise source*] with a minimum of [**selection**: *128*, *192*, *256*] bits of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011, of the keys and CSPs that it will generate.

FCS_RBG_EXT.1.3

The TSF shall be capable of providing output of the RBG to tenant software on the TOE that request random bits.

C.2.2.6 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

Management functions:

- Ability to manage import and export keys/secrets to and from protected storage.

Audit: FCS_STG_EXT.1

There are no auditable events foreseen.

FCS_STG_EXT.1 Protected Storage

Hierarchical to: No other components.

Dependencies to: FCS_CKM.4 Cryptographic Key Destruction

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*].

FCS_STG_EXT.1.2

The TSF shall support the capability of [**selection**: *importing keys/secrets into the TOE*, *causing the TOE to generate keys/secrets*] upon request of [**selection**: *a client application*, *an administrator*].

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*].

Management: FCS_STG_EXT.2

No specific management functions are identified.

Audit: FCS_STG_EXT.2

There are no auditable events foreseen.

FCS_STG_EXT.2 Key Storage Encryption

Hierarchical to: No other components.

Dependencies to:

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 encryption methods as specified in FCS_COP.1/KeyEnc.

Management: FCS_STG_EXT.3

No specific management functions are identified.

Audit: FCS_STG_EXT.3

There are no auditable events foreseen.

FCS_STG_EXT.3 Key Integrity Protection

Hierarchical to: No other components.

Dependencies to: FCS_COP.1: Cryptographic Operations

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**:

- *Symmetric encryption in [**selection**: AES_CCM, AES_GCM, AES_KW, AES_KWP] mode in accordance with FCS_COP.1/SKC*,
- *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*

].

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.2.3 Class FIA - Identification and Authentication

This PP-Module defines the following extended components as part of the FIA class originally defined by CC Part 2:

C.2.3.1 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, Authentication Failure Handling, requires the TSF to monitor authorization attempts, including counting and limiting the number of attempts at failed or passed authorizations.

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: FIA_UAU.5 Multiple Authentication Mechanisms
FIA_SMF_EXT.1 Management Functions Specification
FCS_CKM_EXT.4 Key Destruction

FIA_AFL_EXT.1.1

The TSF shall consider password and [*no other*] as critical authentication mechanisms.

FIA_AFL_EXT.1.2

The TSF shall detect when a configurable positive integer within [**assignment**: *range of acceptable values for each authentication mechanism*] of [**selection**: *unique*, *non-unique*] unsuccessful authentication attempts occur related to 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.

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 will 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 single critical authentication mechanism has been surpassed, the TSF shall perform a wipe of all protected data.

FIA_AFL_EXT.1.6

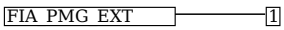
The TSF shall increment the number of unsuccessful authentication attempts prior to notifying the user that the authentication was unsuccessful.

C.2.3.2 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 requirements, minimum lengths, maximum lifetime, and similarity constraints.

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.

Audit: FIA_PMG_EXT.1

No specific audit requirements are forseen.

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 [**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 up to [**assignment**: *an integer greater than or equal to 14*] characters shall be supported.

C.2.3.3 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 a 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

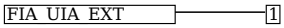
The TSF shall limit automated user authentication attempts by [**selection**: *preventing authentication via an external port, enforcing a delay between incorrect authentication attempts*] for all authentication mechanisms selected in FIA_UAU.5.1. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

C.2.3.4 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 Identification and Authentication, requires the TSF to ensure that all subjects attempting to perform TSF-mediated actions are identified and authenticated prior to authorizing these actions to be performed.

Management: FIA_UIA_EXT.1

No specific management functions are identified.

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. Administrator authentication attempts.
b. All use of the identification and authentication mechanism.
c. Administrator session start time and end time.

FIA_UIA_EXT.1 Administrator Identification and Authentication

Hierarchical to: No other components.

Dependencies to: FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UIA_EXT.1.1

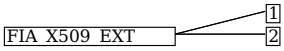
The TSF shall require Administrators to be successfully identified and authenticated using one of the methods in FIA_UAU.5 before allowing any TSF-mediated management function to be performed by that Administrator.

C.2.3.5 FIA_X509_EXT X.509 Certificate

Family Behavior

This family defines requirements for the validation and use of X.509 certificates.

Component Leveling



FIA_X509_EXT.1, X.509 Certificate Validation, defines how the TSF must validate X.509 certificates that are presented to it.

FIA_X509_EXT.2, X.509 Certificate Authentication, requires the TSF to identify the functions for which it uses X.509 certificates for authentication

Management: FIA_X509_EXT.1

The following actions could be considered for the management functions in FMT:

a. Configuration of certificate revocation checking method.

Audit: FIA_X509_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure to validate a certificate.

FIA_X509_EXT.1 X.509 Certificate Validation

Hierarchical to: No other components.

Dependencies to: FPT_STM.1 Reliable Time Stamps

FCS_COP.1 Cryptographic Operations

FCS_RBG_EXT.1 Random Bit Generation

FIA_X509_EXT.1.1

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

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a trusted certificate
- The TOE shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field
- The TSF shall validate revocation status of the certificate using [**selection**: *OCSP as specified in RFC 6960, a CRL as specified in RFC 5759, an OCSP TLS Status Request Extension (OCSP stapling) as specified in RFC 6066, OCSP TLS Multi-Certificate Status Request Extension (i.e., OCSP Multi-Stapling) as specified in RFC 6961*].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing Purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.

FIA_X509_EXT.1.2

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

Management: FIA_X509_EXT.2

The following actions could be considered for the management functions in FMT:

a. Configuration of TSF behavior when certificate revocation status cannot be determined.

Audit: FIA_X509_EXT.2

There are no auditable events foreseen.

FIA_X509_EXT.2 X.509 Certificate Authentication

Hierarchical to: No other components.

Dependencies to: FIA_X509_EXT.1 X.509 Certificate Validation

FTP_ITC_EXT.1 Trusted Channel Communications

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [**assignment**: *secure transport protocols*], and [**assignment**: *other uses*].

FIA_X509_EXT.2.2

When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [**assignment**: *action to take*].

C.2.4 Class FAU - Security Audit

This PP-Module defines the following extended components as part of the FAU class originally defined by CC Part 2:

C.2.4.1 FAU_STG_EXT Off-Loading of Audit Data

Family Behavior

This family defines requirements for the TSF to securely transmit extract audit data from the TOE.

Component Leveling

FAU_STG_EXT.1, Off-Loading of Audit Data, specifies how audit data may be tramsmitted or removed from the TOE.

Management: FAU_STG_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure and manage the audit system and audit data, including the ability to configure name/address of audit/logging server to which to send audit/logging records.

Audit: FAU_STG_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. On failure of logging function, capture record of failure and record upon restart of logging function.

FAU_STG_EXT.1 Off-Loading of Audit Data

Hierarchical to: No other components.

Dependencies to: FAU_GEN.1 Audit Data Generation

FTP_ITC_EXT.1 Trusted Channel Communications

FAU_STG_EXT.1.1

The TSF shall be able to tranfer generated audit data to an external IT entity using [**selection**:

- *a trusted channel as specified in FTP_ITC_EXT.1,*
- *removeable media requiring physical access to the platform*

].

C.2.5 Class FMT - Security Management

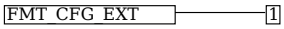
This PP-Module defines the following extended components as part of the FMT class originally defined by CC Part 2:

C.2.5.1 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, Secure by Default Configuration, requires that default Admnistrator credentials be changed immediately after first use.

Management: FMT_CFG_EXT.1

No management functions.

Audit: FMT_CFG_EXT.1

No auditable events are forseen.

FMT_CFG_EXT.1 Secure by Default Configuration

Hierarchical to: No other components.

Dependencies to: No dependencies.

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.2.5.2 FMT_MOF_EXT Management of Security Function Behavior

Family Behavior

This family defines the management functions that can be performed by user roles.

Component Leveling

FMT_MOF_EXT.1, Management of Security Functions Behavior, requires that management functions be assigned to user roles.

Management: FMT_MOF_EXT.1

No additional management functions are envisioned.

Audit: FMT_MOF_EXT.1

No auditable events are forseen.

FMT_MOF_EXT.1 Management of Security Functions Behavior

Hierarchical to: No other components.

Dependencies to: FMT_SMF_EXT.1 Specification of Management Function

FMT_MOF_EXT.1.1

The TSF shall be capable of supporting [**selection**: *local*, *remote*] administration.

FMT_MOF_EXT.1.2

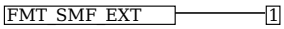
The TSF shall restrict the ability to perform the functions in column 3 of Table 19 to an authenticated administrator.

C.2.5.3 FMT_SMF_EXT Specification of Security Management Functions

Family Behavior

This family defines all security management functions for the TOE.

Component Leveling



FMT_SMF_EXT.1, Specification of Management Functions, requires that certain security functionality be configurable.

Management: FMT_SMF_EXT.1

No additional management functions are envisioned

Audit: FMT_SMF_EXT.1

No auditable events are forseen.

FMT_SMF_EXT.1 Specification of Management Functions

Hierarchical to: No other components.

Dependencies to: FMT_MOF_EXT: Management of Security Functions Behavior

FMT_SMF_EXT.1.1

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

Table 19: Management Functions

Status Markers:
M - Mandatory
O - Optional/Objective

M = Mandatory (TOE must provide that function to that role)
O = Optional (TOE may or may not provide that function to that role)
S = Selection-Based (TOE must provide that function to that role if the TOE claims a particular selection-based SFR)

Number	Function	Admin	Notes
1	Ability to configure and manage the audit functionality and audit data	X	See FAU_GEN.1.
2	Ability to configure name/address of audit/logging server to which to send audit/logging records	X	See FAU_STG_EXT.1.
3	Ability to review audit records.	X	See FAU_SAR.1.
4	Ability to configure the cryptographic functionality.	X	See FTP_ITC_EXT.1.
5	Ability to initiate a trusted channel for remote administration.	X	See FTP_TRP.1.
6	Ability to set parameters for allowable number of authentication failures.	X	See FIA_AFL_EXT.1.
7	Ability to configure password length and complexity.	X	See FIA_PMG_EXT.1.
8	Ability to configure authentication throttling policy.	X	See FIA_TRT_EXT.1.
9	Ability to manage authentication methods and change default authorization factors	X	See FIA_PMG_EXT.1
10	Ability to configure of certificate revocation checking methods.	X	See FIA_X509_EXT.1.
11	Ability to configure TSF behavior when certificate revocation status cannot be determined.	X	See FIA_X509_EXT.2.
12	Ability to configure action to take on integrity failure.	X	See FPT_ROT_EXT.2, FPT_ROT_EXT.3, and FPT_RVR_EXT.1.
13	Ability to initiate the update process.	X	See FPT_TUD_EXT.1.
14	Ability to action to take on update failure.	X	See FPT_TUD_EXT.2 and FPT_TUD_EXT.3.
15	Ability to manage import and export keys/secrets to and from protected storage.	X	See FCS_STG_EXT.1.

]

C.2.6 Class -

This PP-Module defines the following extended components as part of the class originally defined by CC Part 2:

C.2.6.1 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: FAU_STG_EXT.1 Off-Loading of Audit Data

FTP_ITC_EXT.1.1

The TSF shall use [**selection**:

- *TLS as conforming to the Functional Package for Transport Layer Security*,
- *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*

] and [**selection**:

- *X.509 certificate-based authentication of the remote peer,*
- *non-certificate-based authentication of the remote peer,*
- *no authentication of the remote peer*

] protocols to provide a communication channel between itself and [**selection**:

- *audit servers (as required by FAU_STG_EXT.1),*
- *remote administrators (as required by FTP_TRP.1.1 if selected in FMT_MOF_EXT.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.

C.2.7 Class FDP - User Data Protection

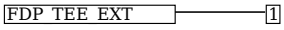
This PP-Module defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.7.1 FDP_TEE_EXT Trusted Execution Environment

Family Behavior

This family defines requirements for Trusted Execution Enviroments 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

No specific management functions are identified.

Audit: FDP_TEE_EXT.1

There are no auditable 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: [*Advanced Trusted Environment: OMTP TR1 v1.1*] and make this TEE available to tenant software.

Appendix D - Entropy Documentation and Assessment

D.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.

D.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 you 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.

D.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.

D.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 E - Equivalency Guidelines

E.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. If a product is found to be compliant with a PP on one platform, then all equivalent products on equivalent platforms are also considered to be compliant with the PP.

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 Protection Profile for General-Purpose Computing Platforms (GPCP). 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 for Equivalence, 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.

E.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.

E.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 20: Factors for Determining Product Equivalence

Factor	Same/Different	Guidance
Product Type	Different	Products in different product classes are not equivalent. Servers, EUDs, and IoT devices 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.

E.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 21: 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.

E.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 F - Use Case Templates

F.1 Server-Class Platform, Basic

The configuration specified below is for the Server-Class Platform, Basic use case describe as [USE CASE 1] Server-Class Platform, Basic.
Include FAU_GEN.1 in the ST
Include FAU_SAR.1 in the ST
Include FAU_STG.1 in the ST
Include FAU_STG.4 in the ST
Include FAU_STG_EXT.1 in the ST

F.2 Server-Class Platform, Enhanced

The configuration specified below is for the Server-Class Platform, Enhanced use case describe as [USE CASE 2] Server-Class Platform, Enhanced.
Include FAU_GEN.1 in the ST
Include FAU_SAR.1 in the ST
Include FAU_STG.1 in the ST
Include FAU_STG.4 in the ST
Include FAU_STG_EXT.1 in the ST
Include FPT_PHP.2 in the ST
Include FPT_PHP.3 in the ST
Include FPT_JTA_EXT.2 in the ST

F.3 Portable Clients (laptops, tablets), Basic

The configuration specified below is for the Portable Clients (laptops, tablets), Basic use case describe as [USE CASE 3] Portable Clients (laptops, tablets), Basic.

F.4 Portable Clients (laptops, tablets), Enhanced

The configuration specified below is for the Portable Clients (laptops, tablets), Enhanced use case describe as [USE CASE 4] Portable Clients (laptops, tablets), Enhanced.
Include FPT_PHP.1 in the ST
Include FPT_JTA_EXT.2 in the ST

F.5 CSfC EUD

The configuration specified below is for the CSfC EUD use case describe as [USE CASE 5] CSfC EUD.
Include FPT_PHP.1 in the ST
Include FPT_PHP.2 in the ST
Include FPT_JTA_EXT.2 in the ST
Include FAU_GEN.1 in the ST
Include FAU_SAR.1 in the ST
Include FAU_STG.1 in the ST
Include FAU_STG.4 in the ST
Include FAU_STG_EXT.1 in the ST

F.6 Tactical EUD

The configuration specified below is for the Tactical EUD use case describe as [USE CASE 6] Tactical EUD.
Include FPT_PHP.3 in the ST
Include FPT_JTA_EXT.2 in the ST
Include FIA_AFL_EXT.1 in the ST

F.7 Enterprise Desktop clients

The configuration specified below is for the Enterprise Desktop clients use case describe as [USE CASE 7] Enterprise Desktop clients.
Include FAU_GEN.1 in the ST
Include FAU_SAR.1 in the ST
Include FAU_STG.1 in the ST
Include FAU_STG.4 in the ST
Include FAU_STG_EXT.1 in the ST

F.8 IoT Devices

The configuration specified below is for the IoT Devices use case describe as [USE CASE 8] IoT Devices.
Include FPT_PHP.1 in the ST
Include FPT_JTA_EXT.2 in the ST

Appendix G - Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
API	Application Programming Interface
BMC	Baseboard Management Controller
Base-PP	Base Protection Profile
CC	Common Criteria
CEM	Common Evaluation Methodology
CMAC	Cipher-based Message Authentication Code
CN	Common Names
CRL	Certificate Revocation List
CSP	Critical Security Parameters
DAR	Data At Rest
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EST	Enrollment over Secure Transport
EUD	End-User Device
FIPS	Federal Information Processing Standards
GPCP	General-Purpose Computing Platform
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
ITSEF	Information Technology Security Evaluation Facility
KMAC	KECCAK Message Authentication Code
MC	Management Controller
NIST	National Institute of Standards and Technology
OE	Operational Environment
OID	Object Identifier
OS	Operating System
PKI	Public Key Infrastructure
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
RNGVS	Random Number Generator Validation System
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
ST	Security Target
SWID	Software Identification
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface

TSS	TOE Summary Specification
USB	Universal Serial Bus
VM	Virtual Machine
VS	Virtualization System
XOR	Exclusive Or
app	Application
cPP	Collaborative Protection Profile

Appendix H - Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017. • Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017. • Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.
[CEM]	Common Evaluation Methodology for Information Technology Security - Evaluation Methodology, CCMB-2012-09-004, Version 3.1, Revision 4, September 2012.
[SP-800-147]	NIST SP800-147 BIOS Protection Guidelines, NIST SP800-147, April 29, 2011.
[SP-800-147B]	NIST SP800-147B BIOS Protection Guidelines for Servers, NIST SP800-147B, August 2014.
[SP-800-193]	NIST SP800-193 Platform Firmware Resiliency Guidelines, NIST SP800-193, May 2018.