Protection Profile for General Purpose Operating Systems

Version: 5.0
2025-03-31
National Information Assurance Partnership

Revision History

Version	Date	Comment
4.0	2015-08-14	Release - significant revision
4.1	2016-03-09	Minor updates - cryptographic modes
4.2	2018-05-22	Multiple Technical Decisions applied
4.2.1	2019-04-22	Formatting changes as a result of PP evaluation
4.3	2022-09-27	• Added compatibility with MDM Agent, Bluetooth, and TLS Modules. • Two factor authentication. • Aligned with CNSA.
5.0	2024-09-06	• Updated to conform to CC:2022. • Incorporated X.509 package. • Incorporated applicable technical decisions.

Contents

1 Introduction

1.1 Overview

The scope of this Protection Profile (PP) is to describe the security functionality of operating systems in terms of [CC] and to define functional and assurance requirements for such products. An operating system is software that manages computer hardware and software resources, and provides common services for application programs. The hardware it manages may be physical or virtual.

1.2 Terms

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

1.2.1 Common Criteria Terms

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

Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in an ST.

1.2.2 Technical Terms

Address Space Layout Randomization (ASLR)	An anti-exploitation feature which loads memory mappings into unpredictable locations. ASLR makes it more difficult for an attacker to redirect control to code that they have introduced into the address space of a process.
Administrator	An administrator is responsible for management activities, including setting policies that are applied by the enterprise on the operating system. This administrator could be acting remotely through a management server, from which the system receives configuration policies. An administrator can enforce settings on the system which 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, as well as its supporting documentation.
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.
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.
Data Execution Prevention (DEP)	An anti-exploitation feature of modern operating systems executing on modern computer hardware, which enforces a non-execute permission on pages of memory. DEP prevents pages of memory from containing both data and instructions, which makes it more difficult for an attacker to introduce and execute code.
Developer	An entity that writes OS software. For the purposes of this document, vendors and developers are the same.
General Purpose Operating System	A class of OSes 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 which are typically used in routers, switches, and embedded devices.
Host-based Firewall	A software-based firewall implementation running on the OS for filtering inbound and outbound network traffic to and from processes running on the OS.
Hybrid Authentication	A hybrid authentication factor is one where a user has to submit a combination of a cryptographic token and a PIN or password and both must pass. If either factor fails, the entire attempt fails.

Operating System (OS)	Software that manages physical and logical resources and provides services for applications. The terms *TOE* and *OS* are interchangeable in this document.
Personal Identification Number (PIN)	An authentication factor that is comprised of a set of numeric or alphabetic characters that may be used in addition to a cryptographic token to provide a hybrid authentication factor. At this time it is not considered as a stand-alone authentication mechanism. A PIN is distinct from a password in that the allowed character set and required length of a PIN is typically smaller than that of a password as it is designed to be input quickly.
Personally Identifiable Information (PII)	Any information about an individual maintained by an agency, including, but not limited to, education, financial transactions, medical history, and criminal or employment history and information which can be used to distinguish or trace an individual's identity, such as their name, social security number, date and place of birth, mother's maiden name, biometric records, etc., including any other personal information which is linked or linkable to an individual.[OMB]
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.
User	A user is subject to configuration policies applied to the operating system by administrators. On some systems under certain configurations, a normal user can temporarily elevate privileges to that of an administrator. At that time, such a user should be considered an administrator.

1.3 Compliant Targets of Evaluation

Compliant TOEs will implement security functionality in the following general areas:

- Accountability: ensuring that information exists to discover unintentional issues with the configuration and operation of the TOE so that the root cause can be determined.
- Integrity: ensuring the integrity of updates to the TOE and enforcing mechanisms that control the deployment and execution of applications running on it.
- Management: providing mechanisms for configuration of the TSF and deployment of applications running on the TOE.
- Protected Storage: ensuring that credentials and file system data are not subject to unauthorized disclosure.
- Protected Communications: ensuring that sensitive data in transit to and from the TOE is adequately protected from unauthorized modification and disclosure.

1.3.1 TOE Boundary

The TOE boundary encompasses the OS kernel and its drivers, shared software libraries, and some application software included with the OS. The applications considered within the TOE are those that provide essential security services, many of which run with elevated privileges. Applications which are covered by more-specific Protection Profiles cannot claim evaluation as part of the OS evaluation, even when it is necessary to evaluate some of their functionality as it relates to their role as part of the OS.

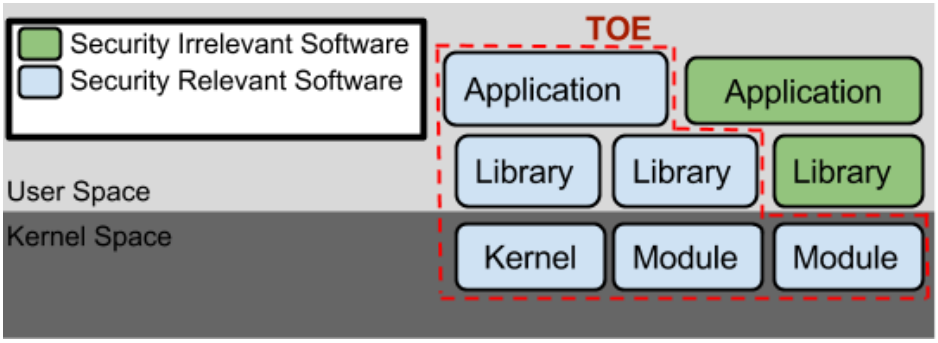


Figure 1: General TOE

1.3.2 TOE Platform

The TOE platform, which consists of the physical or virtual hardware on which the TOE executes, is outside the scope of evaluation. At the same time, the security of the TOE relies upon it. Other hardware components which independently run their own software and are relevant to overall system security are also outside the scope of evaluation.

1.4 Use Cases

Requirements in this Protection Profile are designed to address the security problems in at least the following use cases. These use cases are intentionally very broad, as many specific use cases exist for an operating system. These use cases may also overlap with one another. An operating system's functionality may even be effectively extended by privileged applications installed onto it. However, these are out of scope of this PP.

[USE CASE 1] End User Devices

The OS provides a platform for end user devices such as desktops, laptops, convertibles, and tablets. These devices may optionally be bound to a directory server or management server.

As this Protection Profile does not address threats against data-at-rest, enterprises deploying operating systems in mobile scenarios should ensure that these systems include data-at-rest protection spelled out in other Protection Profiles. Specifically, this includes the Protection Profiles for *Full Drive Encryption - Encryption Engine*, *Full Drive Encryption - Authorization Acquisition*, and *Software File Encryption*. The *Protection Profile for Mobile Device Fundamentals* includes requirements for data-at-rest protection and is appropriate for many mobile devices.

[USE CASE 2] Server Systems

The OS provides a platform for server-side services, either on physical or virtual hardware. Many specific examples exist in which the OS acts as a platform for such services, including file servers, mail servers, and web servers.

[USE CASE 3] Cloud Systems

The OS provides a platform for providing cloud services running on physical or virtual hardware. An OS is typically part of offerings identified as Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS).

This use case typically involves the use of virtualization technology which should be evaluated against the *Protection Profile for Server Virtualization*.

1.5 Package Usage

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

Functional Package for X.509, Version 1.0

No CA Claims Permitted

The TOE will not be a certificate authority because the PP does not provide for the operation of an operating system as a CA. Thus, the ST author shall select the option to request a certificate from an external CA in FIA_XCU_EXT.2.1 and shall not select any options elsewhere in the package that involve claiming the ability to be a CA.

No Platform-Provided Functionality Claims Permitted

Because the TOE boundary is the entire operating system, there is no "platform" that exists outside the logical boundary of the TOE that the TSF could access. Therefore, the ST author shall not claim platform-provided functionality for any of the functions specified in the FP. In other words, any applicable SFRs shall include the claim to implement or provide the functionality, rather than rely on a platform. These may include one or more of the following examples, depending on the TOE, but apply to any requirements where such a selection exists:

- FIA_CMCC_EXT.1
- FIA_CMPC_EXT.1
- FIA_ESTC_EXT.1
- FIA_X509_EXT.1
- FIA_X509_EXT.2
- FIA_X509_EXT.3

Limitations on Signature Algorithms in FIA_X509_EXT.1.1

The TOE must utilize appropriate cryptographic algorithms that conform to CNSA standards. Thus, the TOE shall utilize no other algorithms outside of those specified in RFC 8603 for certificate validation and CRL signature validation (if claimed). If OCSP is claimed, the ST author shall only claim support for algorithms that use SHA-384 and either 3072-bit RSA or greater, or NIST P-384. In all cases, the assignment that allows the ST author to specify additional algorithms shall not be selected.

Required Extension Processing for FIA_X509_EXT.1.2

The ST author shall select the options to process the basicConstraints and extendedKeyUsage extensions. Other extensions may be selected as appropriate without restriction.

CRL or OCSP-based Revocation Required for FIA_X509_EXT.1.3

The TOE must support revocation that only involves CRL or OCSP. Accordingly, the TOE shall select only from options involving CRL or OCSP in FIA_X509_EXT.1.3 (e.g., the selection to treat all certificates older than a given short timeframe is not an acceptable substitute or alternative for supporting CRL or OCSP).

Connections to CRL or OCSP Servers Required for FIA_X509_EXT.1.4

Because the TOE is required to support CRL or OCSP, the TSF shall support an appropriate mechanism for obtaining revocation status information. In the case of CRL, the ST author shall claim that revocation status information is obtained via network connection to a CRL distribution point. In the case of OCSP, the ST author shall claim that revocation status information is obtained via network connection to an OCSP responder, via OCSP stapling, or via OCSP multi-stapling.

Restrictions on Acceptable Key Usage Values for FIA_X509_EXT.1.5

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

Restrictions on Acceptable Functions for FIA_X509_EXT.2.1

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

Restrictions on Acceptable Purposes for Certificate Acquisition for FIA_XCU_EXT.2.1

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

1.6 Product Features Mapped to Implementation-dependent Requirements

The features enumerated below, if implemented by the TOE, require that additional SFRs be claimed in the ST.

1.6.1 Bluetooth Support

A conformant TOE may implement Bluetooth functionality. If so, it is necessary for the TOE to support specific cryptographic algorithms and key sizes to support Bluetooth communications. The TOE will also claim a PP-Configuration that includes the PP-Module for Bluetooth.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM_EXT.7**

1.6.2 Key Encapsulation Support

A conformant TOE is required to implement trusted communications. Depending on the specific protocols used and the supported connection parameters for these protocols, it may be necessary to implement a key encapsulation algorithm as part of key establishment.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM.2**

1.6.3 Key Agreement Support

A conformant TOE is required to implement trusted communications. Depending on the specific protocols used and the supported connection parameters for these protocols, it may be necessary to implement one or more key agreement algorithms as part of key establishment.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM_EXT.7**

1.6.4 WLAN Support

A conformant TOE may implement WLAN client functionality. If so, it is necessary to implement authenticated encryption and key wrapping in support of secure wireless communications. The TOE will also claim a PP-Configuration that includes the PP-Module for

Wireless LAN Clients.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM.2**

2 Conformance Claims

Conformance Statement

An ST must claim exact conformance to this PP.

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

CC Conformance Claims

This PP is conformant to Part 2 (extended) and Part 3 (extended) of Common Criteria CC:2022, Revision 1.

PP Claim

This PP does not claim conformance to any Protection Profile.

The following PPs and PP-Modules are allowed to be specified in a PP-Configuration with this PP:

- PP-Module for Virtual Private Network (VPN) Clients, version 3.0
- PP-Module for Bluetooth, version 2.0
- PP-Module for Mobile Device Management Agent, version 2.0
- PP-Module for Wireless LAN Clients, version 2.0
- cPP-Module for Biometric Enrolment and Verification, version 1.1

Package Claim

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

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

3 Security Problem Definition

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

3.1 Threats

T.NETWORK_ATTACK

An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may engage in communications with applications and services running on or part of the OS with the intent of compromise. Engagement may consist of altering existing legitimate communications.

T.NETWORK_EAVESDROP

An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between applications and services that are running on or part of the OS, resulting in modification or disclosure of sensitive communications.

T.LOCAL_ATTACK

An attacker may compromise applications running on the OS. The compromised application may provide maliciously formatted input to the OS through a variety of channels including unprivileged system calls and messaging via the file system.

T.LIMITED_PHYSICAL_ACCESS

An attacker may attempt to access data on the OS while having a limited amount of time with the physical device, resulting in unauthorized disclosure or modification of the TSF's data or behavior.

3.2 Assumptions

A.PLATFORM

The OS relies upon a trustworthy computing platform for its execution. This underlying platform is out of scope of this PP.

A.PROPER_USER

The user of the OS is not willfully negligent or hostile, and uses the software in compliance with the applied enterprise security policy. At the same time, malicious software could act *as* the user, so requirements which confine malicious subjects are still in scope.

A.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 Security Objectives

4.1 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.

4.2 Security Objectives Rationale

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

Table 1: Security Objectives Rationale

Assumption or OSP	Security Objectives	Rationale
A.PLATFORM	OE.PLATFORM	The operational environment objective OE.PLATFORM is realized through A.PLATFORM.
A.PROPER_USER	OE.PROPER_USER	The operational environment objective OE.PROPER_USER is realized through A.PROPER_USER.
A.PROPER_ ADMIN	OE.PROPER_ ADMIN	The operational environment objective OE.PROPER_ADMIN is realized through A.PROPER_ADMIN.

5 Security Requirements

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

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

5.1 Security Functional Requirements

5.1.1 Auditable Events for Mandatory SFRs

Table 2: Auditable Events for Mandatory SFRs

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	No events specified	N/A
FCS_CKM.1/AKG	No events specified	N/A
FCS_CKM.1/SKG	No events specified	N/A
FCS_CKM.6	No events specified	N/A
FCS_COP.1/AEAD	No events specified	N/A
FCS_COP.1/Hash	No events specified	N/A
FCS_COP.1/KeyedHash	No events specified	N/A
FCS_COP.1/SKC	No events specified	N/A
FCS_COP.1/SigGen	No events specified	N/A
FCS_COP.1/SigVer	No events specified	N/A
FCS_RBG.1	No events specified	N/A
FCS_STO_EXT.1	No events specified	N/A
FDP_ACF_EXT.1	Successful and unsuccessful attempts to access data	No additional information
FIA_AFL.1	No events specified	N/A
FIA_UAU.5	No events specified	N/A
FMT_MOF_EXT.1	Successful or unsuccessful management of the behavior of any TOE functions	No additional information

	Change in permissions to a set of users that have the ability to manage a given function	No additional information
FMT_SMF_EXT.1	No events specified	N/A
FPT_ACF_EXT.1	Unauthorized attempts to perform operations against protected data	No additional information
FPT_ASLR_EXT.1	No events specified	N/A
FPT_FLS.1	No events specified	N/A
FPT_SBOP_EXT.1	No events specified	N/A
FPT_STM.1	No events specified	N/A
FPT_TST.1	No events specified	N/A
FPT_TST_EXT.1	Failure of the integrity checking mechanism	No additional information
FPT_TUD_EXT.1	Failure of the integrity checking mechanism	No additional information
	Successful completion of updates	No additional information
FPT_TUD_EXT.2	Failure of the integrity checking mechanism	No additional information
	Successful completion of updates	No additional information
FTP_ITC_EXT.1	Initiation of trusted channel	No additional information
	Termination of trusted channel	No additional information
	Failure of trusted channel functions	No additional information
FTP_TRP.1	No events specified	N/A

5.1.2 Audit Data Generation (FAU)

FAU_GEN.1 Audit Data Generation

FAU_GEN.1.1

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

a. Start-up and shut-down of the audit functions;
b. All auditable events for the [*not specified*] level of audit;
c. Specifically defined auditable events listed in Table 2
[
d.
- *Authentication events (Success/Failure);*
- *Use of privileged/special rights events (Successful and unsuccessful security, audit, and configuration changes);*
- *Privilege or role escalation events (Success/Failure);*
- *Auditable events as defined in the Functional Package for Secure Shell (SSH), version 2.0;*
- *Auditable events as defined in the Functional Package for Transport Layer Security (TLS), version 2.1;*
- *Auditable events as defined in the Functional Package for X.509, version 1.0;*
- *[**selection**:*
  - *File and object events (Successful and unsuccessful attempts to create, access, delete, modify, modify permissions)*
  - *User and Group management events (Successful and unsuccessful add, delete, modify, disable, enable, and credential change)*
  - *Audit and log data access events (Success/Failure)*
  - *Cryptographic verification of software (Success/Failure)*

- *Attempted application invocation with arguments (Success/Failure e.g. due to software restriction policy)*
- *System reboot, restart, and shutdown events (Success/Failure)*
- *Kernel module loading and unloading events (Success/Failure)*
- *Administrator or root-level access events (Success/Failure)*
- *[**assignment**: other specifically defined auditable events].*

]

].

FAU_GEN.1.2

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

a. Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event;
b. For each audit event type, based on the auditable event definitions of the functional components included in the PP, PP-Module, functional package, or ST, [**assignment**: *other audit relevant information*]

.

Application Note: The term *subject* here is understood to be the user that the process is acting on behalf of. If no auditable event definitions of functional components are provided, then no additional audit-relevant information is required.

Evaluation Activities

FAU_GEN.1
TSS
The evaluator shall examine the TSS to verify that it describes any mechanisms the TSF uses to generate audit data for security-relevant audit events. If multiple separate mechanisms are used, the evaluator shall verify that the TSS identifies the events that are logged via each mechanism.

Guidance
The evaluator shall examine the operational guidance to verify that it includes information on how security-relevant audit events are logged and what the format for the audit records are. The evaluator shall also verify that the operational guidance includes sample audit records for each security-relevant event so that it is clear to the reader how the TSF identifies those events in the audit data.

The evaluator shall check the administrative guide and ensure that it provides a format for audit data. Each audit data format type must be covered, along with a brief description of each field. The evaluator shall ensure that the fields contains the information required.

Tests
The evaluator shall test the OS's ability to correctly generate audit data by having the TOE generate audit data for the events listed in the ST. This should include all instance types of an event specified. When verifying the test results, the evaluator shall ensure the audit data generated during testing match the format specified in the administrative guide, and that the audit data provides the required information.

5.1.3 Cryptographic Support (FCS)

FCS_CKM.1/AKG Cryptographic Key Generation - Asymmetric Key

FCS_CKM.1.1/AKG

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

The following table provides the allowable choices for completion of the selection operations of FCS_CKM.1/AKG.

Table 3: Allowable choices for FCS_CKM.1/AKG

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
RSA	RSA	Modulus of size [**selection**: *3072, 4096, 6144, 8192*] bits	NIST FIPS PUB 186-5 (Section A.1.1)
ECC-ERB	ECC-ERB - Extra Random Bits	Elliptic Curve [**selection**: *P-256, P-384, P-521*]	FIPS PUB 186-5 (Section A.2.1) NIST SP 800-186 (Section 3) [NIST Curves]
ECC-RS	ECC-RS - Rejection Sampling	Elliptic Curve [**selection**: *P-256, P-384, P-521*]	FIPS PUB 186-5 (Section A.2.2) NIST SP 800-186 (Section 3) [NIST Curves]
FFC-ERB	FFC-ERB - Extra Random Bits	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]* • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.3) [key pair generation] [**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
FFC-RS	FFC-RS - Extra Random Bits	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]* • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.3) [key pair generation] [**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
LMS	LMS	Private key size = [**selection**: • *192 bits with[**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Winternitz parameter = [**selection**: *1, 2, 4, 8*] Tree height = [**selection**: *5, 10, 15, 20, 25*]	RFC 8554 [LMS] NIST SP 800-208 [parameters]
ML-KEM	ML-KEM KeyGen	Parameter set = ML-KEM-1024	NIST FIPS 203 (Section 7.1)

ML-DSA	ML-DSA KeyGen	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.1)
XMSS	XMSS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Tree height = [**selection**: *10*, *16*, *20*]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]

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

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

For ECC selections, “P-256” may only be selected if the PP-Module for Bluetooth is included in the ST and P-256 may only be used specifically for Bluetooth functions.

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

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

If the TSF acts only as a receiver in the RSA key establishment scheme, the ST does not need to implement RSA key generation.

Evaluation Activities

FCS_CKM.1/AKG

TSS

The evaluator shall examine the TSS to verify that it describes how the TOE generates a key based on output from a random bit generator as specified in FCS_RBG.1. The evaluator shall review the TSS to verify that it describes how the functionality described by FCS_RBG.1 is invoked. If support for P-256 is claimed, the evaluator shall examine the TSS to verify that it is only used for Bluetooth functions.

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

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

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

Guidance

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

Tests

The following tests are conditional based on the selections made in the SFR. The evaluator shall perform the following tests or witness respective tests executed by the developer. The tests must be executed on a platform that is

as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

RSA Key Generation

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

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

These are:

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

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

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

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

Testing for Random Provable Primes and Conditional Methods

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

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

Testing for Random Probable Primes Method

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

- *n = p*q*
- *p and q are probably prime according to Miller-Rabin tests with error probability $<2^{(-125)}$*
- *$2^{16} < e < 2^{256}$ and e is an odd integer*
- *GCD(p-1,e) = 1*
- *GCD(q-1,e) = 1*
- *$|p-q| > 2^{(nlen/2 - 100)}$*
- *$p \geq squareroot(2)*(2^{(nlen/2 -1)})$*
- *$q \geq squareroot(2)*(2^{(nlen/2 -1)})$*
- *$2^{(nlen/2)} < d < LCM(p-1,q-1)$*
- *e*d = 1 mod LCM(p-1,q-1)*

Elliptic Curve Key Generation

Identifier	*Cryptographic Key Generation Algorithm*	*Cryptographic Algorithm Parameters*	*List of Standards*
ECC-ERB	*ECC – Extra Random Bits*	*Elliptic Curve [**selection:** P-256, P-384, P-521]*	*NIST FIPS PUB 186-5 (Section A.2.1)* *NIST SP 800-186 (Section 3) [NIST Curves]*
ECC-RS	*ECC – Rejection Sampling*	*Elliptic Curve [**selection:** P-256, P-384, P-521]*	*NIST FIPS PUB 186-5 (Section A.2.2)* *NIST SP 800-186 (Section 3) [NIST Curves]*

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

- *Elliptic curve [P-256, P-384, P-521]*
- *Key pair generation method [extra random bits, rejection sampling]*

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

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

Finite Field Cryptography Key Generation

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

FFC-RS	*FFC – Rejection Sampling*	*Static domain parameters approved for [**selection:** IKE groups [**selection:** MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [**selection:** ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]]*	*NIST SP 800-56A Revision 3 (Section 5.6.1.1.4) [key pair generation]* *[**selection:** RFC 3526 [IKE groups], RFC 7919 [TLS groups]]*

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

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

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

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

LMS Key Generation

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

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

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

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

Table 4: Number of LMS Test Cases

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

25	*1*

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

ML-KEM Key Generation

Identifier	***Cryptographic Key Generation Algorithm***	***Cryptographic Algorithm Parameters***	***List of Standards***
ML-KEM	*ML-KEM Key Generation*	*Parameter set = [ML-KEM-1024]*	*NIST FIPS PUB 203 (Section 7.1)*

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

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

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

ML-DSA Key Generation

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

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

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

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

XMSS Key Generation

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

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

- *Hash algorithm [SHA-256/192, SHAKE256/192, SHA-256, SHAKE256]*

- *Tree height [10, 16, 20] (XMSS only)*

Table 5: Number of Test Cases for XMSS^{MT}

Height	*Number of test cases*
10	*5*
16	*4*
20	*3*
40	*2*
60	*1*

XMSS Key Generation Test
For each supported combination of hash algorithm and tree height, the evaluator shall generate one public key for each test case. The number of test cases depends on the tree height as specified in Table 5.

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

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

FCS_CKM.1/SKG Cryptographic Key Generation - Symmetric Key

FCS_CKM.1.1/SKG

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

The following table provides the allowable choices for completion of the selection operations of FCS_CKM.1/SKG.

Table 6: Allowable choices for FCS_CKM.1/SKG

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

Evaluation Activities

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

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

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

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

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

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

FCS_CKM.2 Cryptographic Key Distribution

This is an implementation-based component. Its inclusion in depends on whether the TOE implements one or more of the following features:

- ***Key Encapsulation Support***
- ***WLAN Support***

as described in Appendix A.3: Implementation-based Requirements.

FCS_CKM.2.1

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

Application Note: Key encapsulation is used in support of TLS when ML-KEM is used as the method of key establishment. Key wrapping is used in support of wireless LAN communications.

Evaluation Activities

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

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

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

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

FCS_CKM.6 Timing and Event of Cryptographic Key Destruction

FCS_CKM.6.1

The TSF shall destroy [*all plaintext keying material and critical security parameters (CSPs)*] when [*no longer needed*].

Application Note: For the purposes of this requirement, key material refers to authentication data, passwords, secret/private symmetric keys, private asymmetric keys, data used to derive keys, values derived from passwords, etc.

FCS_CKM.6.2

The TSF shall destroy **plaintext** keying material **and critical security parameters** by implementing key destruction in accordance with the following rules:

- For volatile memory, the destruction shall be executed by a single direct overwrite [**selection**: *consisting of a pseudo-random pattern using the TSF or platform RBG (as specified in FCS_RBG.1), consisting of zeroes*]
- For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo-random pattern using the TSF or platform RBG (as specified in FCS_RBG.1), followed by a read-verify.
- For non-volatile flash memory, that is not wear-leveled, the destruction shall be executed [**selection**: *by a single direct overwrite consisting of zeros followed by a read-verify, by a block erase that erases the reference to memory that stores data as well as the data itself*]
- For non-volatile flash memory, that is wear-leveled, the destruction shall be executed [**selection**: *by a single direct overwrite consisting of zeros, by a block erase*]
- For non-volatile memory other than EEPROM and flash, the destruction shall be executed by a single direct overwrite with a random pattern that is changed before each write.

Application Note: 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 visible. For instance, in a given implementation that overwrites a key stored in non-volatile memory, the application may have access to the file system details and may be able to logically address specific memory locations. In another implementation, that instructs the underlying platform to destroy the representation of a key stored in non-volatile memory, the application may simply have a handle to a resource and can only ask the platform to delete the resource, as may be the case with a platforms secure key store. The latter implementation should only be used for the most restricted access. The level of detail to which the TOE has access will be reflected in the TSS section of the ST.

Several selections allow assignment of a 'value that does not contain any CSP.' This means that the TOE uses some other specified data not drawn from a source that may contain key material or reveal information about key material, and not being any of the particular values listed as other selection options. The point of the phrase 'does not contain any CSP' is to ensure that the overwritten data is carefully selected, and not taken from a general 'pool' that might contain current or residual data that itself requires confidentiality protection.

For the selection , a key can be considered destroyed by destroying the key that protects the key. If a key is wrapped or encrypted it is not necessary to "overwrite" that key, overwriting the key that is used to wrap or encrypt the key used to encrypt/decrypt data, using the appropriate method for the memory type involved, will suffice. For example, if a product uses a KEK to encrypt a Data Encryption Key (DEK), destroying the KEK using one of the methods in FCS_CKM.6 is sufficient, since the DEK would no longer be usable (of course, presumes the DEK is still encrypted and the KEK cannot be recovered or re-derived.).

Evaluation Activities

FCS_CKM.6

TSS

The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator shall check to ensure the TSS lists each type of key that is stored in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

If the selection "destruction of all key encrypting keys (KEKs) protecting the target key according to FCS_CKM.6.2, where none of the KEKs protecting the target key are derived" is included, the evaluator shall examine the TOE's keychain in the TSS and identify each instance when a key is destroyed by this method. In each instance the evaluator will verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM.6.2. The evaluator shall verify that all of the keys capable of decrypting the target key are not able to be derived to reestablish the keychain after their destruction.

Guidance

There are a variety of concerns that may prevent or delay key 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 any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.

Some examples of what is expected to be in the documentation are provided here.

When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.

The drive should be healthy and contains minimal corrupted data and should be end-of-lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

Tests

- *Test FCS_CKM.6:1: Applied to each key held as in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:*
  1. *Record the value of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Cause the TOE to stop the execution but not exit.*
  5. *Cause the TOE to dump the entire memory of the TOE into a binary file.*
  6. *Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.*

  Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.
- *Test FCS_CKM.6:2: Applied to each key held in non-volatile memory and subject to destruction by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.*
  1. *Identify the purpose of the key and what access should fail when it is deleted. (e.g. the data encryption key being deleted would cause data decryption to fail.)*
  2. *Cause the TOE to clear the key.*
  3. *Have the TOE attempt the functionality that the cleared key would be necessary for.*

  The test succeeds if step 3 fails.
- *Test FCS_CKM.6:3:*

  Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key as the TOE has no visibility into the inner workings and completely relies on the underlying platform.

The following tests are used to determine if the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.

Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):

1. *Record the value of the key in the TOE subject to clearing.*
2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
3. *Cause the TOE to clear the key.*
4. *Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.*

- *Test FCS_CKM.6:4: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:*
  1. *Record the logical storage location of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.*

  The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

FCS_CKM_EXT.7 Cryptographic Key Agreement

This is an implementation-based component. Its inclusion in depends on whether the TOE implements one or more of the following features:

- ***Bluetooth Support***
- ***Key Agreement Support***

as described in Appendix A.3: Implementation-based Requirements.

FCS_CKM_EXT.7.1

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

The following table provides the allowable choices for completion of the selection operations of FCS_CKM_EXT.7.

Table 7: Allowable choices for FCS_CKM_EXT.7

Identifier	Cryptographic algorithm	Cryptographic parameters	List of standards
KAS2	RSA	Modulus size [**selection**: *3072, 4096, 6144, 8192*] bits	NIST SP 800-56B Revision 2 (Section 8.3) [KAS2]
DH	Finite Field Cryptography Diffie-Hellman	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]* • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	NIST SP 800-56A Revision 3 (Section 5.7.1.1) [DH] [**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
ECDH	Elliptic Curve Diffie-Hellman	Elliptic Curve [**selection**: *P-256, P-384, P-521*]	NIST SP 800-56A Revision 3 (Section

	5.7.1.2) [ECDH] NIST SP 800-186 (Section 3.2.1) [NIST Curves]

Application Note: This SFR is claimed if the TSF supports key agreement schemes as a method of key establishment for trusted channels. This will generally apply to all conformant TOEs, except in the rare (but possible) case where only key encapsulation is used.

All of the above algorithms with the selectable parameters are CNSA 1.0 compliant with the exception of P-256 ECDH. This may only be selected if the PP-Module for Bluetooth is included in the ST and P-256 may only be used specifically for Bluetooth functions.

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.

Evaluation Activities

FCS_CKM_EXT.7

TSS

The evaluator shall ensure that the TSS documents that the security strength of the material contributed by the TOE is sufficient for the security strength of the key and the agreement method. If support for P-256 is claimed, the evaluator shall examine the TSS to verify that it is only used for Bluetooth functions.

Guidance

There are no guidance EAs for this component.

Tests

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

KAS2

Identifier	*Cryptographic Algorithm*	*Cryptographic Parameters*	*List of Standards*
KAS2	*RSA*	*Modulus Size [**selection:** 3072, 4096, 6144, 8192] bits*	*NIST SP 800-56B Revision 2 (Section 8.3) [KAS2]*

To test the TOE's implementation of the of the KAS2 RSA Key Agreement scheme, the evaluator shall perform the Algorithm Functional Test and Validation Test using the following input parameters:

- *RSA Private key format [Basic, Prime Factor, Chinese Remainder Theorem]*
- *Modulo value [3072, 4096, 6144, 8192]*
- *Role [initiator, responder]*

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

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

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

Algorithm Functional Test

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

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

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

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

FFC Diffie-Hellman Key Agreement

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

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

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

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

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

Elliptic Curve Diffie-Hellman Key Agreement

Identifier	***Cryptographic Algorithm***	***Cryptographic Parameters***	***List of Standards***
ECDH	*Elliptic Curve Diffie-Hellman*	*Elliptic Curve [**selection:** P-256, P-384, P-521*	*NIST SP 800-56A Revision 3 (Section 5.7.1.2) [ECDH]* *NIST SP 800-186 (Section 3.2.1) [NIST Curves]*

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

- *Elliptic Curve [P-256, P-384, P-521]*

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

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

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

FCS_COP.1.1/AEAD

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

The following table provides the allowable choices for completion of the selection operations of FCS_COP.1/AEAD.

Table 8: Allowable choices for FCS_COP.1/AEAD

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

Application Note: The use of 256-bit keys for AES encryption is required by CNSA 1.0 and 2.0. 128-bit keys may only be used when the PP-Module for Bluetooth is claimed and only for the purpose of Bluetooth communications.

Evaluation Activities

FCS_COP.1/AEAD
TSS

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

If a CCM mode algorithm is selected, then the evaluator shall examine the TOE summary specification to confirm that it describes how the nonce is generated and that the same nonce is never reused to encrypt different plaintext pairs under the same key. If the use of 128-bit keys is selected in conjunction with AES-CCM, the evaluator shall verify that the TSS describes the use of 128-bit AES-CCM as being solely for Bluetooth.

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

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

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

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

AES-CCM

Identifier	***Cryptographic Algorithm***	***Cryptographic Key Sizes***	***List of Standards***
AES-CCM	*AES in CCM mode with non-repeating nonce, minimum size of 64 bits*	*[**selection:**128 bits, 256 bits]*	*[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES]* *[**selection:** ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C] [CCM]*

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

- *Key Size [128,256] bits*
- *Associated data size [0-65536] bits in increments of 8*
- *Payload size [0-256] bits in increments of 8*
- *IV/Nonce size [64-104] bits in increments of 8*
- *Tag size [32-128] bits in increments of 16*

Algorithm Functional Tests

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

Variable Associated Data Test

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

Variable Payload Test

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

Variable Nonce Test

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

Variable Tag Test

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

Decryption Verification Test

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

AES-GCM

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

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

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

Encryption Algorithm Functional Tests

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

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

Note that the IV size is always 96 bits.

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

Decryption Algorithm Functional Tests

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

FCS_COP.1/Hash Cryptographic Operation - Hashing

FCS_COP.1.1/Hash

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

Application Note: In accordance with CNSA 1.0 and 2.0:

- SHA-1 hash is no longer permitted to be used as a hash function,
- SHA3 hashes may be used only for internal hardware functionality such as boot integrity checks, and
- SHA-256 is permitted only for use as a PRF or MAC as part of a key derivation function, or as part of LMS or XMSS.

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

Evaluation Activities

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

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

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

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

SHA-256, SHA-384, SHA-512

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

Algorithm Functional Test

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

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

Monte Carlo Test

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

There are two versions of the Monte Carlo Test for SHA-1 and SHA-2. Either one is acceptable. For the standard Monte Carlo test the message hashed is always three times the length of the initial seed.

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

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

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

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

Large Data Test

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

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

SHA3-384, SHA3-512

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

Algorithm Functional Test

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

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

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

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

Monte Carlo Test

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

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

```
MD[0] = SEED
INITIAL_SEED_LENGTH = LEN(SEED)
For 100 iterations
For i = 1 to 1000
MSG = MD[i-1];
```

```
if LEN(MSG) >= INITIAL_SEED_LENGTH:
MSG = leftmost INITIAL_SEED_LENGTH bits of MSG
else:
MSG = MSG || INITIAL_SEED_LENGTH - LEN(MSG) 0 bits
MD[i] = SHA3(MSG)
MD[0] = MD[1000]
Output MD[0]
```

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

Large Data Test

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

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

FCS_COP.1/KeyedHash Cryptographic Operation - Keyed Hash

FCS_COP.1.1/KeyedHash

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

The following table provides the allowable choices for completion of the selection operations of FCS_COP.1/KeyedHash.

Table 9: Allowable choices for FCS_COP.1/KeyedHash

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

Application Note: The intent of this requirement is to specify the keyed-hash message authentication function used for key establishment purposes for the various cryptographic protocols used by the OS (e.g., trusted channel). The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1/Hash.

In accordance with CNSA 1.0 and 2.0, HMAC-SHA-256 may be used only as a PRF or MAC step in a key derivation function.

Evaluation Activities

FCS_COP.1/KeyedHash

TSS

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

The evaluator shall examine the TSS to verify that if HMAC-SHA-256 is selected, that it is being used only as a PRF or MAC step in a key derivation function.

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

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

HMAC

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

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

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

Algorithm Functional Test

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

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

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

FCS_COP.1/KeyEncap Cryptographic Operation - Key Encapsulation

This is a selection-based component. Its inclusion depends upon selection from FCS_CKM.2.1.

FCS_COP.1.1/KeyEncap

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

The following table provides the allowable choices for completion of the selection operations of FCS_COP.1/KeyEncap.

Table 10: Allowable choices for FCS_COP.1/KeyEncap

Identifier	**Cryptographic algorithm**	**Cryptographic key sizes**	**List of standards**

ML-KEM	ML-KEM	Parameter set = ML-KEM-1024	NIST FIPS 203

Application Note: For this PP, the only anticipated use of key encapsulation is the use of ML-KEM as part of key establishment for trusted communications.

Evaluation Activities

FCS_COP.1/KeyEncap

TSS

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

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

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

ML-KEM Key Encapsulation

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

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

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

Encapsulation Test

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

Encapsulation Key Check (if supported)

The evaluator shall generate 10 encapsulation keys such that:

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

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

Decapsulation Key Check (if supported)

The evaluator shall generate 10 decapsulation keys such that:

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

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

Decapsulation Test

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

FCS_COP.1/KeyWrap Cryptographic Operation - Key Wrapping

This is a selection-based component. Its inclusion depends upon selection from FCS_CKM.2.1.

FCS_COP.1.1/KeyWrap

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

The following table provides the allowable choices for completion of the selection operations of FCS_COP.1/KeyWrap.

Table 11: Allowable choices for FCS_COP.1/KeyWrap

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

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

Application Note: NIST 800-57p1rev5 sec. 5.6.2 specifies that the size of key used to protect the key being transported should be at least the security strength of the key it is protecting.

Evaluation Activities

FCS_COP.1/KeyWrap

TSS

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

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

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

AES-KW

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

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

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

Algorithm Functional Test

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

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

The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.

AES-KWP

Identifier	***Cryptographic Algorithm***	***Cryptographic Key Sizes***	***List of Standards***
AES-KWP	*AES in KWP mode*	*256 bits*	*[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES]* *NIST SP 800-38F (Section 6.3) [KWP mode]*

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

- *Key size [256] bits*
- *Keyword cipher type [cipher, inverse]*
- *Payload sizes [8-4096] bits by 8s*

Algorithm Functional Test

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

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

The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.

FCS_COP.1/SigGen Cryptographic Operation - Signature Generation

FCS_COP.1.1/SigGen

The TSF shall perform [*digital signature generation*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic Algorithm*] and cryptographic key sizes [**selection**: *Cryptographic Key Sizes*] that meet the following: [**selection**: *List of Standards*]

The following table provides the allowable choices for completion of the selection operations in FCS_COP.1/SigGen.

Table 12: Allowable choices for FCS_COP.1/SigGen

Identifier	**Cryptographic Algorithm**	**Cryptographic Key Sizes**	**List of Standards**
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection**: *3072, 4096, 6144, 8192*] bits, hash [**selection**: *SHA-384, SHA-512*]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [**selection**: *3072, 4096, 6144, 8192*] bits, hash	RFC 8017 (Section 8.1) [PKCS#1 v2.2]

		[**selection**: *SHA-384, SHA-512*], Salt Length (*sLen*) such that [**assignment**: *0 ≤ sLen ≤ hLen (Hash Output Length)*] and Mask Generation Function = MGF1	FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]
ECDSA	ECDSA	Elliptic Curve [**selection**: *P-384, P-521*], per-message secret number generation [**selection**: *extra random bits, rejection sampling, deterministic*] and hash function using [**selection**: *SHA-384, SHA-512*]	[**selection**: *ISO/IEC 14888-3:2018 (Subclause 6.6), FIPS PUB 186-5 (Sections 6.3.1, 6.4.1*][ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]
ML-DSA	ML-DSA Signature Generation	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.2)

Application Note: As stated in the selection, 2048-bit RSA signatures may be used only for secure boot.

Evaluation Activities

FCS_COP.1/SigGen

TSS

The evaluator shall examine the TSS and verify that any hash function is the appropriate security strength for the signing algorithm.

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

The evaluator shall examine the TSS to verify that the TOE has appropriate measures in place to ensure that hash-based signature algorithms do not reuse private keys.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

RSA-PKCS Signature Generation

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
RSA-PKCS	*RSASSA-PKCS1-v1_5*	*Modulus of size [**selection:** 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512]*	*RFC 8017 (Section 8.2) [PKCS #1 v2.2]* *NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]*

To test the TOE's ability to perform RSA Digital Signature Generation using PKCS1-v1,5 signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

- *Modulus size [3072, 4096, 6144, 8192] bits*

- *Hash algorithm [SHA-384, SHA-512]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate three test cases using random data. The evaluator shall compare the results against those from a known good implementation.

RSA-PSS Signature Generation

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
RSA-PSS	*RSASSA-PSS*	*Modulus of size [**selection:** 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512], Salt Length (sLen) such that [**assignment:** 0 ≤ sLen ≤ hLen (Hash Output Length)] and Mask Generation Function = MGF1*	*RFC 8017 (Section 8.2) [PKCS #1 v2.2]* *NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]*

To test the TOE's ability to perform RSA Digital Signature Generation using PSS signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

- *Modulus size [3072, 4096, 6144, 8192] bits*
- *Hash algorithm [SHA-384, SHA-512]*
- *Salt length [Fixed based on implementation]*
- *Mask function [MGF1]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate three test cases using random data. The evaluator shall compare the results against those from a known good implementation.

ECDSA Signature Generation

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
ECDSA	*ECDSA*	*Elliptic Curve [**selection:** P-384, P-521], per-message secret number generation [**selection:** extra random bits, rejection sampling, deterministic] and hash function using [**selection:** SHA-384, SHA-512]*	*[**selection:** ISO/IEC 14888-3:2018 (Subclause 6.6), NIST FIPS PUB 186-5 (Sections 6.3.1, 6.4.1] [ECDSA]* *NIST SP-800 186 (Section 4) [NIST Curves]*

To test the TOE's ability to perform ECDSA Digital Signature Generation using extra random bits or rejection sampling for secret number generation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Elliptic Curve [P-384, P-521]*
- *Hash algorithm [SHA-384, SHA-512]*

To test the TOE's ability to perform ECDSA Digital Signature Generation using deterministic secret number generation, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Elliptic Curve [P-384, P-521]*
- *Hash algorithm [SHA-384, SHA-512]*

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate 10 test cases using random data. The evaluator shall compare the results against those from a known good implementation.

ML-DSA Signature Generation

Identifier	*Cryptographic Algorithm Parameters*	*Cryptographic Key Sizes*	*List of Standards*
ML-DSA	*ML-DSA SigGen*	*Parameter set = ML-DSA-87*	*NIST FIPS PUB 204 (Section 5.2)*

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

- *Parameter set [ML-DSA-87]*
- *Seed [32 random bytes] (for non-deterministic signature testing), or*
- *Seed [32 zero bytes] (for deterministic signature testing)*
- *Message to sign [8-65535] bytes*
- *Mu value (if generated externally)*
- *Previously generated private key (sk)*
- *Context (for external interface testing)*

Algorithm Functional Test

For each combination of supported parameter set and capabilities, the evaluator shall require the implementation under test to generate 15 signatures pairs using 15 different randomly generated 32-byte seed values. To determine correctness, the evaluator shall compare the resulting key pairs with those generated using a known good implementation using the same inputs.

Known Answer Test for Rejection Cases

For each supported parameter set, the evaluator shall cause the TOE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in the fourth row for each corresponding test case below.

The test values are defined as follows:

- *Seed is the seed to generate the key pair (pk, sk)*
- *Hash of keys is computed by SHA-256(pk||sk)*
- *Message is the message to be signed*
- *Hash of sig is computed by SHA-256(sig)*

ML-DSA-87 Test Cases for Rejection Cases

```
Test case 87-RC-01
Seed:                   E4F5AFCF697E0EC3C1BDEB66FAA903221E803902F9C3F716E1056A63D77DC250
Hash of Keys:   61618E8DDA6998072C8EB36974E03880D741CAF0BD523356DFC161E7C9E63934
Message:                F4F1C05004D5B946F69EAFE104C4020519086ADDB9582A20FDE887D13DFC36B1
Hash of sig:    B584E38FA442FC3C81A147D4BDBF058D73C822CAF5CA4C06B0110867F60A8001

Test case 87-RC-02
Seed:                   8B828D871254D6C57384A8E7025AA3F7160CAD1D2C754499DF3844426062C3DD
Hash of Keys:   BB64481317D6C0DBAD20C0C7EF11078AD54E5D574F4A07652115A95F77C655FA
Message:                0F9409C5A4930C25B83FC5B77FDB5BB49C75372DE724D9C1A77DB700CF0CF154
Hash of sig:    F86B49BE9DEB2B209BDEB4E922E5939E92D38E562C44BB09AFBD67323C345192

Test case 87-RC-03
Seed:                   E693D282CACB8CE65FD4D108DA7A373F097F0AA9713550BE242AAD5BD3E2E452
Hash of Keys:   B0BEAF56713A69BD4AB2CBEE006FA5001E7B41F3AE541E05F088933AA0CC78DF
Message:                24DABB9D57ADEBD560ED65D9451C5106D437061708F849BA53F3543CDF9AAAE0
Hash of sig:    DBF65CEFF9F96A74AAF6F3AB27B043231BE6AA04FBA2EEC987A24A00BDD6A08E

Test case 87-RC-04
Seed:                   4002163EB8EED01A8E0919BA8C07D291341EDCAE25B02B9779A2CFFE50561AF0
Hash of Keys:   FED1BE685C20ECB322FC40D41DEE7E0E98D0409FBF989CAE71B8AD2D58AD645E
Message:                EE316BB5EBED53325B4A55571C60657B53E353B51B831F4A0BBB28107EBA4BA8
Hash of sig:    3BE9B5545FDCED92547B3409C83B3312CCB5792A8EC3A4DA63BA692C79BEF17C

Test case 87-RC-05
Seed:                   9C7AD524F65854C27E565BCEDF8E86D650F13A40D0448F9AE10C05F10F777120
Hash of Keys:   0EA872CA5A4BEA94F4E8EF7ED31800727899A51059FDEE111E5CB15F0233B534
```

```
Message:                CE09831294AA96CAF684B9E667947B021C57B24C138EC7D4DA270694C82F2E08
Hash of sig:    3B9526CEE6587F2418BFE603ADB0F7DF0D69EBA31C9F9F005C60C993945EBD33

Test case 87-RC-06
Seed:                   2EB7676D4A28700DA7772A7A035EB495CAA6F842352A74824EF5FD891BC38B2A
Hash of Keys:   D5B73703A1DDC5BCB0D14AE39B193A25D6ADA6535827973181ADB0BE70435A5B
Message:                C2B3A0AC483A5517682285C205974B2A506946448A8F7D3E1934C155EFDFE922
Hash of sig:    375D598704B722C8A1FEF1626FD7738A532C06329AA4217357460E3B729660F8

Test case 87-RC-07
Seed:                   E4E80CCE8B26DF1B02B99949851EE2F907FE4F0CC34790352C76D5D91634D073
Hash of Keys:   84B7E61684A12698400B09EA332EA3C4FBCFA47FE37FD6AE725CBC5FA8A99D3F
Message:                89E6AB43C9CB1CC59C3986D53217A558357E62102A26F666F2B64CD1DBB7A536
Hash of sig:    7C4AABD163CAEF8F6EBFDA3E3EEBC0A9604675B0E991ABAFD284F1AE8BA07B2A

Test case 87-RC-08
Seed:                   5787262B803499223D4E5A8C1EE572E89F7A69B359B3F8505355B0BDEAB95E5C
Hash of Keys:   85AE1DE605A7B479C02730BF4B7DD6D0FD8FFE5C980893CA6DAD00BD8BD1CE68
Message:                D3230C4E061964BBFB17702432D5D36FC1EB3D1068F8CCAA84044776E3B5CC55
Hash of sig:    D3ABE460EE2DD9595F413CFE2780A319E4E4DFD6592995298A7AB0B82A5E2815

Test case 87-RC-09
Seed:                   CE099B99330537DD153052243FC32ACAD509A126AB982410258858567D410D79
Hash of Keys:   E04A9F15EDF8F078EB336CE624249EF2A8EDF2CDBF6A8276E9F5E92ED9B0BAE8
Message:                0035931762665F561A1B22176567E3B10FDE2441521F77030733A8E39312EEEE
Hash of sig:    3EEF413CB5EB179896ECA172D0DBFB9B251545DC561D61580BD5BBC8B6D734E1

Test case 87-RC-10
Seed:                   FC8F2929878CBD81E1CCC23913F290380120C043A4A8A251AEEBF09705B8E590
Hash of Keys:   7E2ECCA86F532E8E8092FEBB6E0007F92E7909AD2BCBE2E02AB375DAC9969E5E
Message:                D3C28875D2671C0EF23BFDC8869E8ECF8868D3F0561C3134D254F7479D0CE0E5
Hash of sig:    EB69A908EDCC04320A0B61AD57E21B044465F2037698636B64229CF2DB259789
```

Known Answer Test for Large Number of Rejection Cases (Total Rejection Count)

For each supported parameter set, the evaluator shall cause the TQE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in the fourth row of the corresponding test case below.

ML-DSA-87 Test Cases for Total Rejection Count

```
Test case 87-LN-01
Seed:                   98B6298051D92BF37293C93C97370747BF527B87B71F6C4264182F45155ADE4C
Hash of Keys:   04A135B5C9B7020332C7B16E7108E8FF7FC1EAE1C23C5FA0B5D5CED0FEEE7424
Message:                D7B0341269259083ABF3C8DC47559A19D57669B4486E0224F376DC43E577A3D8
Hash of sig:    58D72D76EC0FB65BFB9893C4479366B79DD7B8B7577E4291D13514FCC76C26DD

Test case 87-LN-02
Seed:                   DFB5BDD90F58571DCA962426C623F13D046BBE814D183886AC90D143EAD725A7
Hash of Keys:   2B6AB8CFCCCC41F759CAF01932E9413F5DC6D949BC827F739866929683FB155E
Message:                21005DB2B583CC826A9684BFFD0EE00AB97E0479FE4A1D266699337540145778
Hash of sig:    C93EA34E00FFFFC3ECEA072D5FB038A83B5539CAF7B831AEDCFA785E50B3CA5E

Test case 87-LN-03
Seed:                   5AD414E0DD0EF2FE685F342871875FDF06F503717A86C3B3466565ADD2096417
Hash of Keys:   BD9C2D52F3FC78DB17E682DA2E78947ECFC0898333838D60C892700B2B0DDA9F
Message:                29139C279816B25F2D6BB52C8247D163544F7BA332C3CF63359B9E23FBC56515
Hash of sig:    DB4BE2DE19FB40437BDB7E9B6578D665DB05B4E88C16907DF4546EBA9BE03AEA

Test case 87-LN-04
Seed:                   484DD2F406A4D15F49A91AD5FC3BDC1D0FF253622EB68F83D6E1C870D0E89E29
Hash of Keys:   A719DC9A77C91C46295555C2353BA0CBEA513DA9A92A5C34D2E949EFF46A12D8
Message:                6AD6E959F0EA60126364FB7C95FA71133F246A9265A11B4965EE78AB0CB5AF0E
Hash of sig:    5050D7A665074EC63D9F3966C1F01A1BFB18F9E83AE0B09F838BC1E2342ED6F4

Test case 87-LN-05
Seed:                   B25C1816F82D59940D5CB829BAC364AAD013C4C16415CE1CF6DCC2F15199B391
Hash of Keys:   ADBB2CD43F222640BD9FF4E61C80E63853E8DC1F759C581B7447C9C166EAA38E
Message:                824E47322895BFFE37B6B4AFC41CF6115C07EEC0C24EB81076C87A1B01AE8617
Hash of sig:    667ADA46073BC69D64DC47BB9A76DD0D78302E7415D87D5E816B05FB95F9E84D

Test case 87-LN-06
Seed:                   B2CE72B3560AF07E06465881F56ADA00262BA708D87B73F39E04E310F3B8A3E9
Hash of Keys:   FD9C4AC53AE803242A62DF933B8E8BAD6CE5207AC4A73683B6D9383B5E70B17A
Message:                A1501CC84C917E0D2D7C27C2AC382220BD8FFFE807DB38E37A9E429EC2781911
Hash of sig:    779553B195E11558EE59EF3942F5F6B446A2144600D1F4F50B300C6C56504760

Test case 87-LN-07
Seed:                   AB01D0E591B7DDCD3C03395AED808FA2763C0A486D44119D621BE0FD0B022B25
```

```
Hash of Keys:   93B6ADE34F78A4ADB36B2F6D2C51DB793E659E1243E80488AE1C03B65125D6D7
Message:                8DE8122D89D15FE84A4C34F6B59B2C4B11F33B6A053154D199B634F557FDF5F6
Hash of sig:    0483045999A79B583F403DB96A736F0F0B24E2DFBC4E5CFA9B50E3D910786F07

Test case 87-LN-08
Seed:                   15D60D3693762F82C9AC1DCB0576936651AC81D863842EDB91109C8EE83AE705
Hash of Keys:   2DF544E2E939AA717741C2437288FAEB308DEB8FF37A2652FAE34BAE8B84D779
Message:                F05946A6113905C34163AEF2246FD69016CE24A7BA40F8E7E42EDAC2D0A44605
Hash of sig:    F8383917AF79C8E540D2356AB05F08B465BF32DFEC444B787CE31BF48CC6C3DD

Test case 87-LN-09
Seed:                   21212285BED53B3411705DAF5F3BDDB6F0618EB571B36EE11A74053407A269F5
Hash of Keys:   737061155A9A03F11F9FEBBB940BED4DD54542C4A6212F89A5EB4EC2BE542782
Message:                FFE38246BF3DEFD9CAD15CC17CEA511C067D582E04227B479E32F9197CF91482
Hash of sig:    C4C12C58032052FB2D21F0C6A7388A63154FB85B74287D2859DE6C1C6F7F277B

Test case 87-LN-10
Seed:                   A2744470587C71BA43EC26DC390CE3531978F315993C653E5D3EFD2849D5D9F1
Hash of Keys:   B1BF37BFFB11531B6ADD697870D7DB2E2462D0A97A63F09C1D0038457C6D795A
Message:                9831A830231A160B9847203341A5F30BF3E87A2A482AEEA6886315C92B5C4E4C
Hash of sig:    46C669D2FEB643A38E54FF87B790CC33F44043A1B6B31DB9474D301328CA2A7F
```

FCS_COP.1/SigVer Cryptographic Operation - Signature Verification

FCS_COP.1.1/SigVer

The TSF shall perform [*digital signature verification*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic Algorithm*] and cryptographic key sizes [**selection**: *Cryptographic Key Sizes*] that meet the following: [**selection**: *List of Standards*]

The following table provides the allowable choices for completion of the selection operations in FCS_COP.1/SigVer.

Table 13: Allowable choices for FCS_COP.1/SigVer

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection**: *2048 (for secure boot only), 3072, 4096, 6144, 8192*] bits and hash [**selection**: *SHA-384, SHA-512*]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [**selection**: *2048 (for secure boot only), 3072, 4096, 6144, 8192*] bits and hash [**selection**: *SHA-384, SHA-512*]	RFC 8017 (Section 8.1) [PKCS#1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]
ECDSA	ECDSA	Elliptic Curve [**selection**: *P-384, P-521*] using hash [**selection**: *SHA-384, SHA-512*]	[**selection**: *ISO/IEC 14888-3:2018 (Subclause 6.6), FIPS PUB 186-5 (Section 6.4.2)*][ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]
LMS	LMS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*	RFC 8554 [LMS] NIST SP 800-208 [parameters]

		] Winternitz parameter = [**selection**: *1*, *2*, *4*, *8*] Tree height = [**selection**: *5*, *10*, *15*, *20*, *25*]	
XMSS	XMSS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Tree height = [**selection**: *10*, *16*, *20*]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]
ML-DSA	ML-DSA Signature Verification	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.3)

Evaluation Activities

FCS_COP.1/SigVer

TSS

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

If "2048 (for secure boot only)" is selected for RSA, the evaluator shall check that the TSS documents that 2048-bit RSA is used only for secure boot and that a larger key size is used for any other function.

Guidance

If "2048 (for secure boot only)" is selected for RSA, the evaluator shall check that the guidance includes an configuration needed to ensure that 2048-bit RSA is used only for secure boot and that a larger key size is used for any other function.

Tests

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

RSA-PKCS Signature Verification

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
RSA-PKCS	*RSASSA-PKCS1-v1_5*	*Modulus of size [**selection:** 2048 (for secure boot only), 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512]*	*RFC 8017 (Section 8.2) [PKCS #1 v2.2]* *NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]*

To test the TOE's ability to perform RSA Digital Signature Verification using PKCS1-v1,5 signature type, the evaluator shall perform Generated Data Test using the following input parameters:

- *Modulus size [2048, 3072, 4096, 6144, 8192] bits*
- *Hash algorithm [SHA-384, SHA-512]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate six test cases using a random message and its signature such that the test cases are modified as follows:

- *One test case is left unmodified*
- *For one test case the Message is modified*
- *For one test case the Signature is modified*
- *For one test case the exponent (e) is modified*
- *For one test case the IR is moved*
- *For one test case the Trailer is moved*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

RSA-PSS Signature Verification

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
RSA-PSS	*RSASSA-PSS*	*Modulus of size [**selection:** 2048 (for secure boot only), 3072, 4096, 6144, 8192] bits, hash [**selection:** SHA-384, SHA-512]*	*RFC 8017 (Section 8.2) [PKCS #1 v2.2]* *NIST FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]*

To test the TOE's ability to perform RSA Digital Signature Verification using PSS signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

- *Modulus size [2048, 3072, 4096, 6144, 8192] bits*
- *Hash algorithm [SHA-384, SHA-512]*
- *Salt length [0-hash length]*
- *Mask function [MGF1]*

Generated Data Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate six test cases using random data such that the test cases are modified as follows:

- *One test case is left unmodified*
- *For one test case the Message is modified*
- *For one test case the Signature is modified*
- *For one test case the exponent (e) is modified*
- *For one test case the IR is moved*
- *For one test case the Trailer is moved*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

ECDSA Signature Verification

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
ECDSA	*ECDSA*	*Elliptic Curve [**selection:** P-384, P-521] and hash function using [**selection:** SHA-384, SHA-512]*	*[**selection:** ISO/IEC 14888-3:2018 (Subclause 6.6), NIST FIPS PUB 186-5 (Sections 6.3.1, 6.4.1) [ECDSA]* *NIST SP-800 186 (Section 4) [NIST Curves]*

To test the TOE's ability to perform ECDSA Digital Signature Verification, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Elliptic Curve [P-384, P-521]*
- *Hash algorithm [SHA-384, SHA-512]*

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall cause the TOE to generate test cases consisting of messages and signatures such that the 21 test cases are modified as follows:

- *Three test cases are left unmodified*
- *For three test cases the Message is modified*
- *For three test cases the key is modified*
- *For three test cases the r value is modified*
- *For three test cases the s value is modified*
- *For three test cases the value r is zeroed*
- *For three test cases the value s is zeroed*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

LMS Signature Verification

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
LMS	*LMS*	*Private key size = [**selection:** 192 bits with [**selection:** SHA256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]], Winternitz parameter = [**selection:** 1, 2, 4, 8], and tree height = [**selection:** 5, 10, 15, 20, 25]*	*RFC 8554 [LMS]* *NIST SP 800-208 [parameters]*

To test the TOE's ability to verify cryptographic digital signature using LMS, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

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

Algorithm Functional Test

For each supported combination of the above parameters, the evaluator shall generate four test cases consisting of signed messages and keys, such that

- *One test case is unmodified (i.e. correct)*
- *For one test case modify the message, i.e. the message is different*
- *For one test case modify the signature, i.e. signature is different*
- *For one test case modify the signature header so that it is a valid header for a different LMS parameter set.*

The TOE must correctly verify the unmodified test case and fail to verify the modified test cases.

XMSS Signature Verification

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
XMSS	*XMSS*	*Private key size = [**selection:** 192 bits with [**selection:** SHA256/192, SHAKE256/192], 256 bits with [**selection:** SHA-256, SHAKE256]], and tree height = [**selection:** 10, 16, 20]*	*RFC 8391 [XMSS]* *NIST SP 800-208 [parameters]*

To test the TOE's ability to verify digital signatures using XMSS or XMSS MT, the evaluator shall perform the XMSS digital signature verification test using the following input parameters:

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

XMSS Digital Signature Verification Test

For each supported combination of the above parameters, the evaluator shall generate four test cases consisting of signed messages and keys, such that

- *One test case is unmodified (i.e. correct)*
- *For one test case modify the message, i.e. the message is different*
- *For one test case modify the signature, i.e. signature is different*
- *For one test case modify the signature header so that it is a valid header for a different XMSS parameter set*

The evaluator shall verify the correctness of the implementation by verifying that the TOE correctly verifies the unmodified test case and fails to verify the modified test cases.

ML-DSA Signature Verification

Identifier	*Cryptographic Algorithm Parameters*	*Cryptographic Key Sizes*	*List of Standards*
ML-DSA	*ML-DSA SigVer*	*Parameter set = ML-DSA-87*	*NIST FIPS PUB 204 (Section 5.2)*

To test the TOE's ability to validate digital signatures using ML-DSA, the evaluator shall perform the Algorithm Functional Test using the following input parameters:

- *Parameter set [ML-DSA-87]*
- *Previously generated signed Message [8-65535] bytes*
- *Mu value (if generated externally)*
- *Context (for external interface testing)*
- *Previously generated public key (pk)*
- *Previously generated Signature*

Algorithm Functional Test

For each combination of supported parameter set and capabilities, the evaluator shall require the implementation under test to validate 15 signatures. Each group of 15 test cases is modified as follows:

- *Three test cases are left unmodified*
- *For three test cases the Signed message is modified*
- *For three test cases the component of the signature that commits the signer to the message is modified*
- *For three test cases the component of the signature that allows the verifier to construct the vector z is modified*
- *For three test cases the component of the signature that allows the verifier to construct the hint array is modified*

The TOE must correctly verify the unmodified signatures and fail to verify the modified signatures.

FCS_COP.1/SKC Cryptographic Operation - Encryption/Decryption

FCS_COP.1.1/SKC

The TSF shall perform [*symmetric-key encryption and decryption*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic Algorithm*] and cryptographic key sizes [**selection**: *Cryptographic Key Sizes*] that meet the following: [**selection**: *List of Standards*]

The following table provides the allowable choices for completion of the selection operations in FCS_COP.1/SKC.

Table 14: Allowable choices for FCS_COP.1/SKC

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CBC	AES in CBC mode with non-repeating and unpredictable IVs	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A*] [CBC]
XTS-AES	AES in XTS mode with unique tweak values that are consecutive non-negative integers starting at an arbitrary non-negative integer	512 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *IEEE Std. 1619-2018, NIST SP 800-38E*] [XTS]
AES-CTR	AES in Counter Mode with a non-repeating initial counter and with no repeated use of counter values across multiple messages with the same secret key	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A*] [CBC]

Evaluation Activities

FCS_COP.1/SKC

TSS

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

If XTS-AES is claimed then the evaluator shall examine the TSS to verify that the TOE creates full-length keys by methods that ensure that the two key halves are different and independent.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

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

AES-CBC

Identifier	*Cryptographic Algorithm*	*Cryptographic Key Sizes*	*List of Standards*
AES-CBC	*AES in CBC mode with non-repeating and unpredictable IVs*	*256 bits*	*[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES]*

			*[**selection:** ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]*

To test the TOE's ability to encrypt and decrypt data using AES in CBC mode, the evaluator shall perform Algorithm Functional Tests and Monte Carlo Tests using the following input parameters:

- *Key size [256] bits*
- *Direction [encryption, decryption]*

Algorithm Functional Tests

Algorithm Functional Tests are designed to verify the correct operation of the logical components of the algorithm implementation under normal operation using different block sizes. For AES-CBC, there are two types of AFTs:

Known-Answer Tests

For each combination of direction and claimed key size, the TOE must be tested using the GFSBox, KeySbox, VarTxt, and VarKey test cases listed in Appendixes B through E of The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), NIST, 15 November 2002.

Multi-Block Message Tests

For each combination of direction and claimed key size, the TOE must be tested against 10 test cases consisting of a random IV, random key, and random plaintext or ciphertext. The plaintext or ciphertext starts with a length of 16 bytes and increases by 16 bytes for each test case until reaching 160 bytes.

Monte Carlo Tests

Monte Carlo tests are intended to test the implementation under strenuous conditions. The TOE must process the test cases according to the following algorithm once for each combination of direction and key size:

```
Key[0] = Key
IV[0] = IV
PT[0] = PT
for i = 0 to 99 {
Output Key[i], IV[i], PT[0]
for j = 0 to 999 {
if (j == 0) {
CT[j] = AES-CBC-Encrypt(Key[i], IV[i], PT[j])
PT[j+1] = IV[i]
} else {
CT[j] = AES-CBC-Encrypt(Key[i], PT[j])
PT[j+1] = CT[j-1]
}
}
Output CT[j]
AES_KEY_SHUFFLE(Key, CT)
IV[i+1] = CT[j]
PT[0] = CT[j-1]
}
```

where

`AES_KEY_SHUFFLE`

is defined as:

```
If ( keylen = 128 )
Key[i+1] = Key[i] xor MSB(CT[j], 128)
If ( keylen = 192 )
Key[i+1] = Key[i] xor (LSB(CT[j-1], 64) || MSB(CT[j], 128))
If ( keylen = 256 )
Key[i+1] = Key[i] xor (MSB(CT[j-1], 128) || MSB(CT[j], 128))
```

The above pseudocode is for encryption. For decryption, swap all instances of CT and PT.

The initial IV, key, and plaintext or ciphertext should be random.

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

XTS-AES

Identifier	*Cryptographic Algorithm*	*Cryptographic Key Sizes*	*List of Standards*
XTS-AES	*AES in XTS mode with unique tweak values that are consecutive non-negative integers starting at an arbitrary non-negative integer*	*512 bits*	*[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES]* *[**selection:** IEEE Std. 1619-2018, NIST SP 800-38E] [XTS]*

To test the TOE's ability to encrypt and decrypt data using AES in XTS mode, the evaluator shall perform the Single Data Unit Test and the Multiple Data Unit Test using the following input parameters:

- *Direction [encryption, decryption]*
- *Key size [512] bits*
- *Tweak value format [128-bit hex string, data unit sequence number]*

Single Data Unit Test

For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. The payload size and data unit size must be equal.

Multiple Data Unit Test

For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. Likewise, the data unit size is determined randomly for each test case from supported values within the range [128-65535] bits. The payload size and data unit size must not be equal.

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

AES-CTR

Identifier	*Cryptographic Algorithm*	*Cryptographic Key Sizes*	*List of Standards*
AES-CTR	*AES in Counter Mode with a non-repeating initial counter and with no repeated use of counter values across multiple messages with the same secret key.*	*256 bits*	*[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES]* *[**selection:** ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A] [CTR]*

To test the TOE's ability to encrypt and decrypt data using AES in CTR mode, the evaluator shall perform the Algorithm Functional Test and the Counter Test using the following input parameters:

- *Direction [encryption, decryption]*
- *Key size [256] bits*

Algorithm Functional Tests

Algorithm Functional Tests are designed to verify the correct operation of the logical components of the algorithm implementation under normal operation using different block sizes. For AES-CTR, there are three types of AFTs:

Known-Answer Tests

For each combination of direction and claimed key size, the TOE must be tested using the GFSBox, KeySbox, VarTxt, and VarKey test cases listed in Appendixes B through E of The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), NIST, 15 November 2002.

Single Block Message Tests

For each combination of direction and claimed key, the evaluator shall generate 10 test cases with a data size of 128 bits.

Partial Block Message Tests

Monte Carlo tests are intended to test the implementation under strenuous conditions. The TOE must process the test cases according to the following algorithm once for each combination of direction and key size:

For each combination of direction and claimed key, the evaluator shall generate five test cases such that the data size is not a multiple of 128 bits.

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

Counter Test

The evaluator shall generate a single message of 1000 blocks (128000 bits) and either encrypt or decrypt it. Back-compute the IVs used and verify that they are unique and increasing (encryption) or decreasing (decryption).

FCS_COP.1/XOF Cryptographic Operation - Extendable-Output Function

This is a selection-based component. Its inclusion depends upon selection from FCS_CKM.1.1/AKG, FCS_COP.1.1/SigVer.

FCS_COP.1.1/XOF

The TSF shall perform [*extendable-output function*] in accordance with a specified cryptographic algorithm [**selection**: *Cryptographic Algorithm*] and parameters [**selection**: *Parameters*] that meet the following: [**selection**: *List of Standards*]

The following table provides the allowable choices for completion of the selection operations of FCS_COP.1/XOF.

Table 15: Allowable choices for FCS_COP.1/XOF

Cryptographic Algorithm	Parameters	List of Standards
SHAKE	Functions = [SHAKE128, SHAKE256]	NIST FIPS PUB 202 Section 6.2 [SHAKE]

Application Note: In accordance with CNSA 2.0, SHAKE is permitted to be used only as a component of LMS or XMSS. Therefore this component is claimed only if LMS or XMSS is claimed in FCS_COP.1/SigVer.

Since LMS and XMSS use both SHAKE128 and SHAKE256 internally, claiming and testing of both functions is mandatory.

Evaluation Activities

FCS_COP.1/XOF
TSS
There are no additional TSS evaluation activities for this component.

Guidance

There are no additional Guidance evaluation activities for this component.

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

SHAKE

Cryptographic Algorithm	*Parameters*	*List of Standards*
SHAKE	*Function = [SHAKE128, SHAKE256]*	*NIST FIPS PUB 202 Section 6.2 [SHAKE]*

To test the TOE's implementation of the SHAKE Extendable Output Function the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Variable Output Test using the following input parameters:

- *Function [SHAKE128, SHAKE256]*
- *Output length [16-65536] bits*

Algorithm Functional Test

For each supported function, generate test cases consisting of random data for every message length from 0 bits (if supported) to rate-1 bits, where rate equals

- *1344 for SHAKE128, and*
- *1088 for SHAKE256.*

Additionally, generate tests cases of random data for messages of every multiple of (rate+1) bits starting at length rate, and continuing until 65535 is exceeded. For SHAKE128, this should result in a total of 1391 test cases.

Monte Carlo Test

The Monte Carlos test takes in a single 128-bit message (SEED) and desired output length in bits, and runs 100 iterations of the chained computation. MaxOutBytes and MinOutBytes are the largest and smallest supported input and output sizes in bytes, respectively.

```
Range = maxOutBytes - minOutBytes + 1
OutputLen = maxOutBytes
For j = 0 to 99
MD[0] = SEED
For i = 1 to 1000
MSG[i] = 128 leftmost bits of MD[i-1]
if (MSG[i] < 128 bits)
Append 0 bits on rightmost side of MSG[i] til MSG[i] is 128 bits
MD[i] = SHAKE(MSG[i], OutputLen * 8)

RightmostOutputBits = 16 rightmost bits of MD[i] as an integer
OutputLen = minOutBytes + (RightmostOutputBits % Range)
Output MD[1000], OutputLen
SEED = MD[1000]
```

Variable Output Test

This test measures the ability of the TOE to generate output digests of varying sizes.

The evaluator shall generate 512 test cases such that the input for each test case consists of 128- bits of random data, and the output length includes the minimum supported value, the maximum supported value, and 510 random values between the minimum and maximum digest sizes supported by the implementation.

FCS_RBG.1 Random Bit Generation (RBG)

FCS_RBG.1.1

The TSF shall perform deterministic random bit generation services using [**selection**:

- *Hash_DRBG (any)*
- *HMAC_DRBG (any)*
- *CTR_DRBG (AES)*

] in accordance with [*NIST SP 800-90A*] after initialization with a seed.

Application Note: NIST SP 800-90A contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used and include the specific underlying cryptographic primitives used in the requirement or in the TSS.

FCS_RBG.1.2

The TSF shall use a [**selection**: *TSF noise source [**assignment**: name of noise source]*, ***multiple TSF noise sources [assignment: names of noise sources]***, *TSF interface for seeding*] for initialized seeding.

Application Note: For the selection in this requirement, the ST author selects "TSF noise source" if a single noise source is used as input to the DRBG. The ST author selects "multiple TSF noise sources" if a seed is formed from a combination of two or more noise sources within the TOE boundary. If the TSF implements two or more separate DRBGs that are seeded in separate manners, this SFR should be iterated for each DRBG. If multiple distinct noise sources exist such that each DRBG only uses one of them, then each iteration would select "TSF noise source"; "multiple TSF noise sources" is only selected if a single DRBG uses multiple noise sources for its seed. The ST author selects "TSF interface for seeding" if noise source data is generated outside the TOE boundary.

If "TSF noise source" is selected, FCS_RBG.3 must be claimed.

If "multiple TSF noise sources" is selected, FCS_RBG.4 and FCS_RBG.5 must be claimed.

If "TSF interface for seeding" is selected, FCS_RBG.2 must be claimed.

FCS_RBG.1.3

The TSF shall update the RBG state by [**selection**: *reseeding, uninstantiating and reinstantiating*] using a [**selection**: *TSF noise source [**assignment**: name of noise source]*, *TSF interface for seeding*] in the following situations: [**selection**:

- *never*
- *on demand*
- *on the condition: [**assignment**: condition]*
- *after [**assignment**: time]*

] in accordance with [**assignment**: *list of standards*].

Evaluation Activities

FCS_RBG.1.1
TSS
The evaluator shall verify that the TSS identifies the DRBGs used by the TOE.

Guidance
If the DRBG functionality is configurable, the evaluator shall verify that the operational guidance includes instructions on how to configure this behavior.

Tests
The evaluator shall perform the following tests:

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

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

bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

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 list contains 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 less than or equal to 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_RBG.1.2
Documentation will be produced and the evaluator shall perform the activities in accordance with Appendix D - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

FCS_RBG.1.3
TSS
The evaluator shall verify that the TSS identifies how the DRBG state is updated, and the situations under which this may occur.

Guidance
If the ST claims that the DRBG state can be updated on demand, the evaluator shall verify that the operational guidance has instructions for how to perform this operation.

Tests
There are no test activities for this element.

FCS_RBG.2 Random Bit Generation (External Seeding)

This is a selection-based component. Its inclusion depends upon selection from FCS_RBG.1.2.

FCS_RBG.2.1

The TSF shall be able to accept a minimum input of [**assignment**: *minimum input length greater than zero*] from a TSF interface for the purpose of seeding.

Application Note: This requirement is claimed when a DRBG is seeded with entropy from one or more noise source that is outside the TOE boundary. Typically the entropy produced by an environmental noise source is conditioned such that the input length has full entropy and is therefore usable as the seed. However, if this is not the case, it should be noted what the minimum entropy rate of the noise source is so that the TSF can collect a sufficiently large sample of noise data to be conditioned into a seed value.

Evaluation Activities

FCS_RBG.2
The evaluator shall examine the entropy documentation required by FCS_RBG.1.2 to verify that it identifies, for each DRBG function implemented by the TOE, the TSF external interface used to seed the TOE's DRBG. The evaluator shall verify that this includes the amount of sampled data and the min-entropy rate of the sampled data such that it can be determined that sufficient entropy can be made available for the highest strength keys that the TSF can

generate (e.g., 256 bits). If the seed data cannot be assumed to have full entropy (e.g., the min-entropy of the sampled bits is less than 1), the evaluator shall ensure that the entropy documentation describes the method by which the TOE estimates the amount of entropy that has been accumulated to ensure that sufficient data is collected and any conditioning that the TSF applies to the output data to create a seed of sufficient size with full entropy.

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

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

Tests
There are no test activities for this component.

FCS_RBG.3 Random Bit Generation (Internal Seeding - Single Source)

This is a selection-based component. Its inclusion depends upon selection from FCS_RBG.1.2.

FCS_RBG.3.1

The TSF shall be able to seed the RBG using a [**selection, choose one of**: *TSF software-based noise source*, *TSF hardware-based noise source [**assignment**: name of noise source]*] with a minimum of [**assignment**: *number of bits*] bits of min-entropy.

Application Note: This requirement is claimed when a DRBG is seeded with entropy from a single noise source that is within the TOE boundary. Min-entropy should be expressed as a ratio of entropy bits to sampled bits so that the total amount of data needed to ensure full entropy is known, as well as the conditioning function by which that data is reduced in size to the seed.

Evaluation Activities

FCS_RBG.3
The evaluator shall examine the entropy documentation required by FCS_RBG.1.2 to verify that it identifies, for each DRBG function implemented by the TOE, the TSF noise source used to seed the TOE's DRBG. The evaluator shall verify that this includes the amount of sampled data and the min-entropy rate of the sampled data such that it can be determined that sufficient entropy can be made available for the highest strength keys that the TSF can generate (e.g., 256 bits). If the seed data cannot be assumed to have full entropy (e.g., the min-entropy of the sampled bits is less than 1), the evaluator shall ensure that the entropy documentation describes the method by which the TOE estimates the amount of entropy that has been accumulated to ensure that sufficient data is collected and any conditioning that the TSF applies to the output data to create a seed of sufficient size with full entropy.

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

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

Tests
There are no test activities for this component.

FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)

This is a selection-based component. Its inclusion depends upon selection from FCS_RBG.1.2.

FCS_RBG.4.1

The TSF shall be able to seed the RBG using [**selection**: *[**assignment**: number] TSF software-based noise source(s), [**assignment**: number] TSF hardware-based noise source(s)*].

Application Note: This requirement is claimed when a DRBG is seeded with entropy from multiple noise sources that are within the TOE boundary. FCS_RBG.5 defines the mechanism by

which these sources are combined to ensure sufficient minimum entropy.

Evaluation Activities

FCS_RBG.4
The evaluator shall examine the entropy documentation required by FCS_RBG.1.2 to verify that it identifies, for each DRBG function implemented by the TOE, each TSF noise source used to seed the TOE's DRBG. The evaluator shall verify that this includes the amount of sampled data and the min-entropy rate of the sampled data from each data source.

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

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

Tests
There are no test activities for this component.

FCS_RBG.5 Random Bit Generation (Combining Noise Sources)

This is a selection-based component. Its inclusion depends upon selection from FCS_RBG.1.2.

FCS_RBG.5.1

The TSF shall [**assignment**: *combining operation*] [**selection**: *output from TSF noise source(s), input from TSF interface(s) for seeding*] to create the entropy input into the derivation function as defined in [**assignment**: *list of standards*], resulting in a minimum of [**assignment**: *number of bits*] bits of min-entropy.

Evaluation Activities

FCS_RBG.5
Using the entropy sources specified in FCS_RBG.4, the evaluator shall examine the entropy documentation required by FCS_RBG.1.2 to verify that it describes the method by which the various entropy sources are combined into a single seed. This should include an estimation of the rate at which each noise source outputs data and whether this is dependent on any system-specific factors so that each source's relative contribution to the overall entropy is understood. The evaluator shall verify that the resulting combination of sampled data and the min-entropy rate of the sampled data is described in sufficient detail to determine that sufficient entropy can be made available for the highest strength keys that the TSF can generate (e.g., 256 bits). If the seed data cannot be assumed to have full entropy (e.g., the min-entropy of the sampled bits is less than 1), the evaluator shall ensure that the entropy documentation describes the method by which the TOE estimates the amount of entropy that has been accumulated to ensure that sufficient data is collected and any conditioning that the TSF applies to the output data to create a seed of sufficient size with full entropy.

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

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

Tests
There are no test activities for this component.

FCS_RBG.6 Random Bit Generation Service

This is an optional component. However, applied modules or packages might redefine it as mandatory.

FCS_RBG.6.1

The TSF shall provide a [**selection**: *hardware, software, [**assignment**: other interface type]*] interface to make the RBG output, as specified in FCS_RBG.1 Random bit generation (RBG), available as a service to entities outside of the TOE.

Application Note: This SFR is defined for the case where an operating system includes a mechanism for e.g. an application or a virtualization system running on the operating system to obtain random bits from the TOE, whether those bits are generated by the TSF or the TSF provides an interface for the third party to access a hardware interface on the platform.

Evaluation Activities

FCS_RBG.6
TSS
The evaluator shall verify that the TSS identifies the interface that the TSF makes available for calling applications to obtain DRBG output.

Guidance
The evaluator shall verify that the guidance documentation includes an API specification for the random bit generation service such that it is clear how a calling application is able to obtain DRBG output from the TSF.

Tests
The evaluator shall invoke the API specified by the guidance documentation to determine that the TSF provides DRBG output upon proper invocation of the API.

FCS_STO_EXT.1 Storage of Sensitive Data

FCS_STO_EXT.1.1

The TSF shall implement functionality to encrypt sensitive data stored in non-volatile storage and provide interfaces to applications to invoke this functionality.

Application Note: Sensitive data will be identified in the TSS by the ST author, and minimally includes credentials and keys. The interface for invoking the functionality could take a variety of forms: it could consist of an API, or simply well-documented conventions for accessing credentials stored as files.

Evaluation Activities

FCS_STO_EXT.1
TSS
The evaluator shall check the TSS to ensure that it lists all persistent sensitive data for which the OS provides a storage capability. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it can be used, and how it is stored. The evaluator shall confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1/SKC.

Guidance
The evaluator shall consult the developer documentation to verify that instructions exists on applications should securely store credentials.

Tests
There are no test activities for this component.

5.1.4 User Data Protection (FDP)

FDP_ACF_EXT.1 Access Controls for Protecting User Data

FDP_ACF_EXT.1.1

The TSF shall implement access controls which can prohibit unprivileged users from accessing files and directories owned by other users.

Application Note: Effective protection by access controls may also depend upon system configuration. This requirement is designed to ensure that, for example, files and directories owned by one user in a multi user system can be protected from access by another user in that system.

Evaluation Activities

FDP_ACF_EXT.1
TSS
The evaluator shall confirm that the TSS comprehensively describes the access control policy enforced by the OS. The description must include the rules by which accesses to particular files and directories are determined for particular users. The evaluator shall inspect the TSS to ensure that it describes the access control rules in such detail that given any possible scenario between a user and a file governed by the OS the access control decision is unambiguous.

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

Tests
The evaluator shall create two new standard user accounts on the TOE and conduct the tests below. If the TSF enforces access controls on user home directories that prevents the evaluator from exercising the permissions directly (i.e., because the home directory itself or its contents are restricted by permissions), the evaluator shall temporarily relax the permissions on the parent directory/directories in order to perform the tests.

- *Test FDP_ACF_EXT.1:1: The evaluator shall authenticate to the system as the first user and create a file within that user's home directory. The evaluator shall then log off the system and log in as the second user. The evaluator shall then attempt to read the file created in the first user's home directory. The evaluator shall ensure that the read attempt is denied.*
- *Test FDP_ACF_EXT.1:2: The evaluator shall authenticate to the system as the first user and create a file within that user's home directory. The evaluator shall then log off the system and log in as the second user. The evaluator shall then attempt to modify the file created in the first user's home directory. The evaluator shall ensure that the modification is denied.*
- *Test FDP_ACF_EXT.1:3: The evaluator shall authenticate to the system as the first user and create a file within that user's user directory. The evaluator shall then log off the system and log in as the second user. The evaluator shall then attempt to delete the file created in the first user's home directory. The evaluator shall ensure that the deletion is denied.*
- *Test FDP_ACF_EXT.1:4: The evaluator shall authenticate to the system as the first user. The evaluator shall attempt to create a file in the second user's home directory. The evaluator shall ensure that the creation of the file is denied.*
- *Test FDP_ACF_EXT.1:5: The evaluator shall authenticate to the system as the first user and attempt to modify the file created in the first user's home directory. The evaluator shall ensure that the modification of the file is accepted.*
- *Test FDP_ACF_EXT.1:6: The evaluator shall authenticate to the system as the first user and attempt to delete the file created in the first user's directory. The evaluator shall ensure that the deletion of the file is accepted.*

FDP_IFC_EXT.1 Information Flow Control

This is a selection-based component. Its inclusion depends upon selection from FTP_ITC_EXT.1.1.

FDP_IFC_EXT.1.1

The OS shall [**selection**:

- *provide an interface which allows a VPN client to protect all IP traffic using IPsec*
- *provide a VPN client that can protect all IP traffic using IPsec*

] with the exception of IP traffic required to establish the VPN connection and [**selection**: *signed updates directly from the OS vendor, no other traffic*] .

Application Note: Typically, the traffic required to establish the VPN connection is referred to as "Control Plane" traffic, whereas the IP traffic protected by the IPsec VPN is referred to as "Data Plane" traffic. All Data Plane traffic must flow through the VPN connection and the VPN must not split-tunnel.

If no native IPsec client is validated or third-party VPN clients may also implement the required Information Flow Control, the first option must be selected. In these cases, the TOE provides an API to third-party VPN clients that allows them to configure the TOE's network stack to perform the required Information Flow Control.

If the TSF implements a native VPN client, then the ST author must select provide a VPN client that can protect all IP traffic using IPsec and includes the PP-Module for VPN Client as part of the ST.

In the future, this requirement may also make a distinction between the current requirement (which requires that when the IPsec trusted channel is enabled, all traffic from the TSF is routed through that channel) and having an option to force the establishment of an IPsec trusted channel to allow any communication by the TSF.

Evaluation Activities

FDP_IFC_EXT.1

TSS

The evaluator shall verify that the TSS section of the ST describes the routing of IP traffic when a VPN client is enabled. The evaluator shall ensure that the description indicates which traffic does not go through the VPN and which traffic does, and that a configuration exists for each in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec).

Guidance

The evaluator shall examine the operational guidance to verify whether the TOE provides its own VPN client or if VPN client capability must be implemented by a third-party application.

Tests

The evaluator shall perform the following test:

- *Test FDP_IFC_EXT.1:1:*
  - ***Step 1:** The evaluator shall enable a network connection. The evaluator shall sniff packets while performing running applications that use the network such as web browsers and email clients. The evaluator shall verify that the sniffer captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.*
  - ***Step 2:** The evaluator shall configure an IPsec VPN client that supports the routing specified in this requirement. The evaluator shall turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step. The evaluator shall verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.*
  - ***Step 3:** The evaluator shall examine the traffic from both step one and step two to verify that all non-excepted Data Plane traffic in Step 2 is encapsulated by IPsec. The evaluator shall examine the Security Parameter Index (SPI) value present in the encapsulated packets captured in Step 2 from the TOE to the Gateway and will verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from the TOE to the Gateway.*
  - ***Step 4:** The evaluator shall perform a ping on the TOE host on the local network and verify that no packets sent are captured with the sniffer. The evaluator shall attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local network, and verify that the TOE discards them.*

5.1.5 Identification and Authentication (FIA)

FIA_AFL.1 Authentication Failure Handling

FIA_AFL.1.1

The TSF shall detect when [**selection**:

- *[**assignment**: positive integer number]*
- *an administrator configurable positive integer within [**assignment**: range of acceptable values]*

] unsuccessful authentication attempts occur related to **events with [selection:**

- ***authentication based on user name and password***
- ***authentication based on user name and a PIN that releases an asymmetric key stored in OE-protected storage***
- ***authentication based on X.509 certificates***

].

Application Note: Selections in FIA_AFL.1 and FIA_UAU.5 must match.

FIA_AFL.1.2

When the defined number of unsuccessful authentication attempts **for an account** has been [*met*], The TSF shall: **[selection: *Account Lockout, Account Disablement, Mandatory Credential Reset, [assignment: list of actions]*]** .

Application Note: The action to be taken will be populated in the assignment of the ST and defined in the administrator guidance.

Evaluation Activities

FIA_AFL.1
TSS
The evaluator shall examine the TSS to verify that it defines the mechanism by which excessive authentication failures are checked (i.e, whether it is a fixed or configurable value) and the actions that are taken against that account when the failure threshold has been reached.

Guidance
The evaluator shall examine the operational guidance to verify that it identifies either the fixed number of successive authentication failures that triggers a response to occur, or if this is a configurable value, the mechanism by which the value is configured. The evaluator shall also examine the operational guidance to verify that it describes the behavior that occurs when the failure threshold is met. If this behavior is configurable, the evaluator shall verify that the operational guidance provides instructions for how to set each response as claimed in FIA_AFL.1.2.

Tests
If configurable, the evaluator shall set an administrator-configurable threshold for failed authentication attempts, or note the ST-specified assignment. The evaluator will then (per selection) repeatedly attempt to authenticate with an incorrect password, PIN, or certificate until the number of attempts reaches the threshold and observe that the behavior claimed in FIA_AFL.1.2 occurs. If the behavior in FIA_AFL.1.2 is configurable, the evaluator shall repeat this test as necessary for each configured behavior and verify that the configured behavior occurs in all cases. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.

- *Test FIA_AFL.1:1: [conditional, to be performed if "authentication based on user name and password" is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator shall attempt to authenticate repeatedly to the system with a known bad password. Once the defined number of failed authentication attempts has been reached the evaluator shall ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator shall ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*
- *Test FIA_AFL.1:2: [conditional, to be performed if "authentication based on user name and a PIN that releases an asymmetric key stored in OE-protected storage" is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator shall attempt to authenticate repeatedly to the system with a known bad PIN. Once the defined number of failed authentication attempts has been reached the evaluator shall ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator shall ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*

- *Test FIA_AFL.1:3: [conditional, to be performed if "authentication based on X.509 certificates" is selected in FIA_AFL.1 and FIA_UAU.5]:The evaluator shall attempt to authenticate repeatedly to the system using a known bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator shall ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator shall ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*

FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UAU.5.1

The **OS** shall provide **[selection:**

- ***authentication based on username and password***
- ***authentication based on username and a PIN that releases an asymmetric key stored in OE-protected storage***
- ***combination of authentication based on user name, password, and time-based one-time password***
- ***authentication based on X.509 certificates***
- ***for use in SSH only, SSH public key-based authentication as specified by the Functional Package for Secure Shell (SSH), version 2.0***

] to support user authentication.

Application Note: The SSH public key-based authentication selection can only be included, and must be included, if FTP_ITC_EXT.1.1 selects SSH
If "authentication based on X.509 certificates" is claimed, the TOE must claim conformance to Functional Package for X.509, version 1.0.

.

FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the [**assignment**: *rules describing how the multiple authentication mechanisms provide authentication*].

Evaluation Activities

FIA_UAU.5
TSS
The evaluator shall ensure that the TSS describes the rules as to how each authentication mechanism specified in FIA_UAU.5.1 is implemented and used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or authentication factor is used), 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). Rules regarding how the authentication factors interact in terms of unsuccessful authentication are covered in FIA_AFL.1.

Guidance
The evaluator shall verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.

Tests

The following content should be included if:

- *authentication based on username and password is selected from FIA_UAU.5.1*
  - *Test FIA_UAU.5:1: The evaluator shall attempt to authenticate to the OS using the known user name and password. The evaluator shall ensure that the authentication attempt is successful.*
  - *Test FIA_UAU.5:2: The evaluator shall attempt to authenticate to the OS using the known user name but an incorrect password. The evaluator will ensure that the authentication attempt is unsuccessful.*

The following content should be included if:

- *authentication based on username and a PIN that releases an asymmetric key stored in OE-protected storage* *is selected from FIA_UAU.5.1*

The evaluator shall examine the TSS for guidance on supported protected storage and will then configure the TOE or OE to establish a PIN which enables release of the asymmetric key from the protected storage (such as a TPM, a hardware token, or isolated execution environment) with which the OS can interface. The evaluator shall then conduct the following tests:

- *Test FIA_UAU.5:3: The evaluator shall attempt to authenticate to the OS using the known user name and PIN. The evaluator shall ensure that the authentication attempt is successful.*
- *Test FIA_UAU.5:4: The evaluator shall attempt to authenticate to the OS using the known user name but an incorrect PIN. The evaluator shall ensure that the authentication attempt is unsuccessful.*

The following content should be included if:

- *combination of authentication based on user name, password, and time-based one-time password* *is selected from FIA_UAU.5.1*

The evaluator shall configure the OS to authentication to authenticate to the OS using a username, password, and one-time password mechanism. The evaluator shall then perform the following tests.

- *Test FIA_UAU.5:5: The evaluator shall attempt to authenticate using a valid username, valid password, and valid one-time password. The evaluator shall ensure that the authentication attempt is successful.*
- *Test FIA_UAU.5:6: The evaluator shall attempt to authenticate using a valid username, invalid password, and valid one-time password. The evaluator shall ensure that the authentication attempt fails.*
- *Test FIA_UAU.5:7: The evaluator shall attempt to authenticate using a valid username, valid password, and invalid one-time password. The evaluator shall ensure that the authentication attempt fails.*
- *Test FIA_UAU.5:8: The evaluator shall attempt to authenticate using a valid username, invalid password, and invalid one-time password. The evaluator shall ensure that the authentication attempt fails.*

Authentication mechanisms related to authentication based on X.509 certificates are tested under FIA_X509_EXT.1 as defined in the Functional Package for X.509, version 1.0 and SSH public key-based authentication are tested in the Functional Package for Secure Shell (SSH), version 2.0.

For each authentication mechanism rule, the evaluator shall ensure that the authentication mechanism(s) behave as documented in the TSS.

5.1.6 Security Management (FMT)

FMT_MOF_EXT.1 Management of Functions Behavior

FMT_MOF_EXT.1.1

The TSF shall restrict the ability to perform the function indicated in the "Administrator" column in FMT_SMF_EXT.1.1 to the administrator.

Application Note: The functions with an "M" in the "Administrator" column must be restricted to (or overridden by) the administrator in the TOE. The functions with an "O" in the "Administrator" column may be restricted to (or overridden by) the administrator when implemented in the TOE at the discretion of the ST author. For such functions, the ST author indicates this by replacing an "O" with an "M" in the ST.

Evaluation Activities

FMT_MOF_EXT.1

TSS

The evaluator shall verify that the TSS describes those management functions that are restricted to Administrators, including how the user is prevented from performing those functions, or not able to use any interfaces that allow access to that function.

Guidance

The evaluator shall examine the operational guidance to verify that for the functions claimed in FMT_SMF_EXT.1 as being restricted to the administrator (i.e., the user cannot perform them), the guidance identifies how that restriction

is enforced (i.e., what role or permission permits the ability to perform the function).

Tests
The evaluator shall also perform the following test.

- *Test FMT_MOF_EXT.1:1: For each function that is indicated as restricted to the administrator, the evaluation will perform the function as an administrator, as specified in the Operational Guidance, and determine that it has the expected effect as outlined by the Operational Guidance and the SFR. The evaluator shall then perform the function (or otherwise attempt to access the function) as a non-administrator and observe that they are unable to invoke that functionality.*

FMT_SMF_EXT.1 Specification of Management Functions

FMT_SMF_EXT.1.1

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

#	Management Function	Administrator	User
1	Enable/disable [**selection**: *screen lock*, *session timeout*]	M	O
2	Configure [**selection**: *screen lock*, *session*] inactivity timeout	M	O
3	import keys/secrets into the secure key storage	O	O
4	Configure local audit storage capacity	O	O
5	Configure minimum password length	O	O
6	Configure minimum number of special characters in password	O	O
7	Configure minimum number of numeric characters in password	O	O
8	Configure minimum number of uppercase characters in password	O	O
9	Configure minimum number of lowercase characters in password	O	O
10	Configure lockout policy for unsuccessful authentication attempts through [**selection**: *timeouts between attempts*, *limiting number of attempts during a time period*]	O	O
11	Configure host-based firewall	O	O
12	Configure name/address of directory server with which to bind	O	O
13	Configure name/address of remote management server from which to receive management settings	O	O
14	Configure name/address of audit/logging server to which to send audit/logging records	O	O
15	Configure audit rules	O	O
16	Configure name/address of network time server	O	O
17	Enable/disable automatic software update	O	O
18	Configure Wi-Fi interface	O	O
19	Enable/disable Bluetooth interface	O	O
20	Enable/disable [**assignment**: *list of other external interfaces*]	O	O

21	[**assignment**: *list of other management functions to be provided by the TSF*]	O	O

.

Application Note: The ST should indicate which of the optional management functions are implemented in the TOE. This can be done by copying the above table into the ST and adjusting the "Administrator" and "User" columns to "M" according to which capabilities are present or not present, and for which privilege level. The Application Note for FMT_MOF_EXT.1 explains how to indicate Administrator or User capability.

The terms "Administrator" and "User" are defined in the glossary. The intent of this requirement is to ensure that the ST is populated with the relevant management functions that are provided by the OS.

Sophisticated account management policies, such as intricate password complexity requirements and handling of temporary accounts, are a function of directory servers. The OS can enroll in such account management and enable the overall information system to achieve such policies by binding to a directory server.

For management functions 12, 13, and 14, it is expected that the interfaces to these remote entities are trusted channels and the ST author is expected to claim them in FTP_ITC_EXT.1 accordingly.

Evaluation Activities

FMT_SMF_EXT.1
TSS
The evaluator shall examine the TSS to verify that identifies the management functions claimed by the TOE and whether they can be performed by an administrator, a user, or both.

Guidance
The evaluator shall examine the operational guidance verify that every management function captured in the ST is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

Tests
The evaluator shall test the OS's ability to provide the management functions by configuring the operating system and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

5.1.7 Protection of the TSF (FPT)

FPT_ACF_EXT.1 Access Controls

FPT_ACF_EXT.1.1

The TSF shall implement access controls which prohibit unprivileged users from modifying:

- Kernel and its drivers/modules
- Security audit logs
- Shared libraries
- System executables
- System configuration files
- [**assignment**: *other objects*]

.

FPT_ACF_EXT.1.2

The TSF shall implement access controls which prohibit unprivileged users from reading:

- Security audit logs
- System-wide credential repositories
- [**assignment**: *list of other objects*]

.

Application Note: "Credential repositories" refer, in this case, to structures containing cryptographic keys or passwords.

Evaluation Activities

FPT_ACF_EXT.1
TSS
The evaluator shall confirm that the TSS specifies the locations of kernel drivers/modules, security audit logs, shared libraries, system executables, and system configuration files. Every file does not need to be individually identified, but the system's conventions for storing and protecting such files must be specified.

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

Tests
The evaluator shall create an unprivileged user account. Using this account, the evaluator shall ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- *Test FPT_ACF_EXT.1:1: The evaluator shall attempt to modify all kernel drivers and modules.*
- *Test FPT_ACF_EXT.1:2: The evaluator shall attempt to modify all security audit logs generated by the logging subsystem.*
- *Test FPT_ACF_EXT.1:3: The evaluator shall attempt to modify all shared libraries that are used throughout the system.*
- *Test FPT_ACF_EXT.1:4: The evaluator shall attempt to modify all system executables.*
- *Test FPT_ACF_EXT.1:5: The evaluator shall attempt to modify all system configuration files.*
- *Test FPT_ACF_EXT.1:6: The evaluator shall attempt to modify any additional components selected.*

The evaluator shall create an unprivileged user account. Using this account, the evaluator shall ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- *Test FPT_ACF_EXT.1:7: The evaluator shall attempt to read security audit logs generated by the auditing subsystem*
- *Test FPT_ACF_EXT.1:8: The evaluator shall attempt to read system-wide credential repositories*
- *Test FPT_ACF_EXT.1:9: The evaluator shall attempt to read any other object specified in the assignment*

FPT_ASLR_EXT.1 Address Space Layout Randomization

FPT_ASLR_EXT.1.1

The TSF shall always randomize process address space memory locations with [**selection**: *8, [**assignment**: number greater than 8]*] bits of entropy except for [**assignment**: *list of explicit exceptions*].

Evaluation Activities

FPT_ASLR_EXT.1
TSS
The evaluator shall examine the TSS to verify that it identifies the number of entropy bits used for ASLR and what exceptions there are to this, if any.

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

Tests
The evaluator shall select 3 executables included with the TSF. If the TSF includes a web browser it must be selected. If the TSF includes a mail client it must be selected. For each of these apps, the evaluator shall launch the same executables on two separate instances of the OS on identical hardware and compare all memory mapping locations. The evaluator shall ensure that no memory mappings are placed in the same location. If the rare chance occurs that two mappings are the same for a single executable and not the same for the other two, the evaluator shall repeat the

test with that executable to verify that in the second test the mappings are different. This test can also be completed on the same hardware and rebooting between application launches.

FPT_BLT_EXT.1 Limitation of Bluetooth Profile Support

This is an objective component.

FPT_BLT_EXT.1.1

The TSF shall disable support for [**assignment**: *list of Bluetooth profiles*] Bluetooth profiles when they are not currently being used by an application on the TOE and shall require explicit user action to enable them.

Application Note: Some Bluetooth services incur more serious consequences if unauthorized remote devices gain access to them. Such services should be protected by measures like disabling support for the associated Bluetooth profile unless it is actively being used by an application on the OS (in order to prevent discovery by a Service Discovery Protocol search), and then requiring explicit user action to enable those profiles in order to use the services. It may be further appropriate to require additional user action before granting a remote device access to that service.

For example, it may be appropriate to disable the OBEX Push Profile until a user pushes a button in an application indicating readiness to transfer an object. After completion of the object transfer, support for the OBEX profile should be suspended until the next time the user requests its use.

Evaluation Activities

FPT_BLT_EXT.1
TSS
The evaluator shall ensure that the TSS lists all Bluetooth profiles that are disabled while not in use by an application and which need explicit user action in order to become enabled.

Guidance
There are no guidance evaluation activities for this component.

Tests
The evaluator shall perform the following tests:

- *Test FPT_BLT_EXT.1:1: The evaluator shall perform this test with a test device that does not have a trust relationship with the TOE. While the service is not in active use by an application on the TOE, the evaluator shall attempt to discover a service associated with a "protected" Bluetooth profile (as specified by the requirement) on the TOE via a Service Discovery Protocol search. The evaluator shall verify that the service does not appear in the Service Discovery Protocol search results. Next, the evaluator shall attempt to gain remote access to the service from a device that does not currently have a trusted device relationship with the TOE. The evaluator shall verify that this attempt fails due to the unavailability of the service and profile.*
- *Test FPT_BLT_EXT.1:2: The evaluator shall repeat Test 1 with a device that currently has a trusted device relationship with the TOE and verify that the same behavior is exhibited.*

FPT_FLS.1 Failure with Preservation of Secure State

FPT_FLS.1.1

The TSF shall preserve a secure state when the following types of failures occur: [*DRBG self-test failure*].

Application Note: The intent of this requirement is to ensure that cryptographic services requiring random bit generation cannot be performed if a failure of a self-test defined in FPT_TST.1 occurs.

Evaluation Activities

FPT_FLS.1
TSS
The evaluator shall verify that the TSF describes how the TOE enters an error state in the event of a DRBG self-test failure.

Guidance
The evaluator shall verify that the guidance documentation describes the error state that results from a DRBG self-test failure and the actions that a user or administrator should take in response to attempt to resolve the error state.

Tests
There are no test activities for this component.

FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

FPT_SBOP_EXT.1.1

The TSF shall [**selection**: *employ stack-based buffer overflow protections, not store parameters or variables in the same data structures as control flow values*].

Application Note: Many OSes store control flow values (i.e. return addresses) in stack data structures that also contain parameters and variables. For these OSes, it is expected that most of the OS, to include the kernel, libraries, and application software from the OS vendor be compiled with stack-based buffer overflow protection enabled. OSes that store parameters and variables separately from control flow values do not need additional stack protections.

Evaluation Activities

FPT_SBOP_EXT.1
TSS
The evaluator shall examine the TSS to verify that it describes how stack-based buffer overflow protection is implemented, or how the TOE is designed in such a way that these protections are not explicitly needed.

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

Tests
For stack-based OSes, the evaluator shall determine that the TSS contains a description of stack-based buffer overflow protections used by the OS. These are referred to by a variety of terms. These include, but are not limited to, tagging, stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner. The evaluator shall also preform the following test:

- *Test FPT_SBOP_EXT.1:1: [Conditional: if stack-based overflow detection can be determined by inventorying]: The evaluator shall inventory the kernel, libraries, and application binaries to determine those that do not implement stack-based buffer overflow protections. This list should match up with the list provided in the TSS.*

For OSes that store parameters/variables separately from control flow values, the evaluator shall verify that the TSS describes what data structures control values, parameters, and variables are stored. The evaluator shall also ensure that the TSS includes a description of the safeguards that ensure parameters and variables do not intermix with control flow values.

FPT_SRP_EXT.1 Software Restriction Policies

This is an objective component.

FPT_SRP_EXT.1.1

The TSF shall restrict execution to only programs which match an administrator-specified [**selection**:

- *file path*
- *file digital signature*
- *version*
- *hash*
- *[**assignment**: other characteristics]*

] .

Application Note: The assignment permits implementations which provide a low level of granularity such as a volume. The restriction is only against direct execution of executable programs. It does not forbid interpreters which may take data as an input, even if this data can subsequently result in arbitrary computation.

Evaluation Activities

FPT_SRP_EXT.1

TSS

The evaluator shall ensure that the description of the supported characteristics in the TSS is consistent with the SFR. The evaluator shall also ensure that any characteristics specified by the ST-author are described in sufficient detail to understand how to test those characteristics.

Guidance

The evaluator shall ensure that the characteristics are described in sufficient detail for administrators to configure policies using them, and that the list of characteristics in the guidance is consistent with the information in the TSS.

Tests

There are two tests for each selection above.

The following content should be included if:

- *file path is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to only allow code execution from the core OS directories. The evaluator shall then attempt to execute code from a directory that is in the allowed list. The evaluator shall ensure that the code they attempted to execute has been executed.

The following content should be included if:

- *file path is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to only allow code execution from the core OS directories. The evaluator shall then attempt to execute code from a directory that is not in the allowed list. The evaluator shall ensure that the code they attempted to execute has not been executed.

The following content should be included if:

- *file digital signature is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to only allow code that has been signed by the OS vendor to execute. The evaluator shall then attempt to execute code signed by the OS vendor. The evaluator shall ensure that the code they attempted to execute has been executed.

The following content should be included if:

- *file digital signature is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to only allow code that has been signed by the OS vendor to execute. The evaluator shall then attempt to execute code signed by another digital authority. The evaluator shall ensure that the code they attempted to execute has not been executed.

The following content should be included if:

- *version is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to allow execution of a specific application based on version. The evaluator shall then attempt to execute the same version of the application. The evaluator shall ensure that the code they attempted to execute has been executed.

The following content should be included if:

- *version is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to allow execution of a specific application based on version. The evaluator shall then attempt to execute an older version of the application. The evaluator shall ensure that the code they attempted to execute has not been executed.

The following content should be included if:

- *hash is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to allow execution based on the hash of the application executable. The evaluator shall then attempt to execute the application with the matching hash. The evaluator shall ensure that the code they attempted to execute has been executed.

The following content should be included if:

- *hash is selected from FPT_SRP_EXT.1.1*

The evaluator shall configure the OS to allow execution based on the hash of the application executable. The evaluator shall modify the application in such a way that the application hash is changed. The evaluator will then attempt to execute the application with the matching hash. The evaluator shall ensure that the code they attempted to execute has not been executed.

The following content should be included if:

- *other is selected from FPT_SRP_EXT.1.1*

The evaluator shall attempt to run an application that should be allowed based on the defined software restriction policy and ensure that it runs.

The following content should be included if:

- *other is selected from FPT_SRP_EXT.1.1*

The evaluator shall then attempt to run an application that should not be allowed the defined software restriction policy and ensure that it does not run.

FPT_TST.1 TSF Self-Testing

FPT_TST.1.1

The TSF shall run a suite of the following self-tests [**selection**: *during initial start-up, periodically during normal operation, at the request of the authorized user, at the conditions [**assignment**: conditions under which self-test should occur]*] to demonstrate the correct operation of [*[TSF DRBG specified in FCS_RBG.1]*].

FPT_TST.1.2

The TSF shall provide authorized users with the capability to verify the integrity of [*[DRBG seed/output data]*].

FPT_TST.1.3

The TSF shall provide authorized users with the capability to verify the integrity of [*[TSF DRBG specified in FCS_RBG.1]*].

Application Note: This SFR is a required dependency of FCS_RBG.1. It is intended to require that any DRBG implemented by the TOE undergo health testing to ensure that the random bit generation functionality has not been degraded. If the TSF supports multiple DRBGs, this SFR should be iterated to describe the self-test behavior for each.

Evaluation Activities

FPT_TST.1
TSS
The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF along with how they are run. This description should include an outline of what the tests are actually doing. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the DRBG is operating correctly.

Note that this information may also be placed in the entropy documentation specified by Appendix D - Entropy Documentation and Assessment.

Guidance

If a self-test can be executed at the request of an authorized user, the evaluator shall verify that the operational guidance provides instructions on how to execute that self-test.

Tests

For each self-test, the evaluator shall verify that evidence is produced that the self-test is executed when specified by FPT_TST.1.1.

If a self-test can be executed at the request of an authorized user, the evaluator shall verify that following the steps documented in the operational guidance to perform the self-test will result in execution of the self-test.

FPT_STM.1 Reliable Time Stamps

FPT_STM.1.1

The TSF shall be able to **obtain** reliable time stamps **from a primary time source for its own use**.

Application Note: In the context of this SFR, a primary time source is that which can be synchronized with a time source that is considered to be authoritative in the context of the overall organization deploying the TOE. Specifically, primary time sources include connectivity to an external NTP server, connectivity to a cellular carrier network, or in the case of a TOE that runs on a virtualization system, a hypercall to the virtualization system's host platform (as this platform could feasibly have its own connectivity to a primary time source). A primary time source does not include the hardware clock on the TOE platform if it is localized only to that system.

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, and that it identifies the primary time source that can be considered to be reliable. The evaluator shall examine the TSS to verify that it describes the TOE's ability to obtain time data from an NTP server, a cellular carrier network, or a hypercall to the TOE's hardware platform if running on a virtualization system.

Guidance

If this functionality is configurable, the evaluator shall examines the operational guidance to ensure it describes how to set the time directly or how to configure the TOE's primary time source.

Tests

- *Test FPT_STM.1:1: If the time is configurable, the evaluator shall follow the operational guidance to set the time. The evaluator shall then use an available interface to observe that the time was set correctly. If the primary time source is configurable, the evaluator shall deliberately set the time to an incorrect value, verify the incorrect time setting is applied, configure the TOE to connect to the primary time source, and verify that the time is subsequently synchronized to the expected time.*

FPT_TST_EXT.1 Boot Integrity

FPT_TST_EXT.1.1

The TSF shall verify the integrity of the bootchain up through the OS kernel and [**selection**:

- *all executable code stored in mutable media*
- *[**assignment**: list of other executable code]*
- *no other executable code*

] prior to its execution through the use of [**selection**:

- *a digital signature using a hardware-protected asymmetric key*

- *a digital signature using an X.509 certificate with hardware-based protection*
- *a hardware-protected hash*

].

Application Note: The bootchain of the OS is the sequence of software, to include the OS loader, the kernel, system drivers or modules, and system files, which ultimately result in loading the OS. The first part of the OS, usually referred to as the first-stage bootloader, must be loaded by the platform. Assessing its integrity, while critical, is the platform's responsibility; and therefore outside the scope of this PP. All software loaded after this stage is potentially within the control of the OS and is in scope.

The verification may be transitive in nature: a hardware-protected public key, X.509 certificate, or hash may be used to verify the mutable bootloader code which contains a key, certificate, or hash used by the bootloader to verify the mutable OS kernel code, which contains a key, certificate, or hash to verify the next layer of executable code, and so on. However, the way in which the hardware stores and protects these keys is out of scope.

If all executable code (including bootloader(s), kernel, device drivers, pre-loaded applications, user-loaded applications, and libraries) is verified, all executable code stored in mutable media should be selected.

If certificates are used, they can be hardware-protected trust store elements or leaf certificates in a certificate chain that terminates in a root CA which is an element of a hardware protected trust store. If the certificates themselves are not trust store elements, revocation information is expected to be available for each CA certificate in the chain that is not a trust element, in accordance with FIA_X509_EXT.1 as defined in the Functional Package for X.509, version 1.0.

Evaluation Activities

FPT_TST_EXT.1

TSS

The evaluator shall verify that the TSS section of the ST includes a comprehensive description of the boot procedures, including a description of the entire bootchain, for the TSF. The evaluator shall ensure that the OS cryptographically verifies each piece of software it loads in the bootchain to include bootloaders and the kernel. Software loaded for execution directly by the platform (e.g. first-stage bootloaders) is out of scope. For each additional category of executable code verified before execution, the evaluator shall verify that the description in the TSS describes how that software is cryptographically verified.

The evaluator shall verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

The evaluator shall also perform the following tests:

- *Test FPT_TST_EXT.1:1: The evaluator shall perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the OS properly boots.*
- *Test FPT_TST_EXT.1:2: The evaluator shall modify a TSF executable that is part of the bootchain verified by the TSF (i.e. Not the first-stage bootloader) and attempt to boot. The evaluator shall ensure that an integrity violation is triggered and the OS does not boot (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that in such a way to invalidate the structure of the module.).*

The following content should be included if:

- *a digital signature using an X.509 certificate with hardware-based protection is selected from FPT_TST_EXT.1.1*

If the ST author indicates that the integrity verification is performed using public key in an X509 certificate, the evaluator shall verify that the boot integrity mechanism includes a certificate validation according to in accordance with FIA_X509_EXT.1 as defined in the Functional Package for X.509, version 1.0 for all certificates in the chain from the certificate used for boot integrity to a certificate in the trust

store that are not themselves in the trust store. This means that, for each X.509 certificate in this chain that is not a trust store element, the evaluator must ensure that revocation information is available to the TOE during the bootstrap mechanism (before the TOE becomes fully operational).

FPT_TUD_EXT.1 Integrity for Installation and Update

FPT_TUD_EXT.1.1

The TSF shall provide the ability to check for updates to the OS software itself and shall use a digital signature scheme specified in FCS_COP.1/SigVer to validate the authenticity of the response.

Application Note: This requirement is about the ability to check for the availability of authentic updates, while the installation of authentic updates is covered by FPT_TUD_EXT.1.2. Use of the digital signature scheme ensures that an attacker cannot influence the response, regarding of whether updates are available.

FPT_TUD_EXT.1.2

The TSF shall [**selection**: *cryptographically verify*, *invoke platform-provided functionality to cryptographically verify*] updates to itself using a digital signature prior to installation using schemes specified in FCS_COP.1/SigVer.

Application Note: The intent of the requirement is to ensure that only digitally signed and verified TOE updates are applied to the TOE.

Evaluation Activities

FPT_TUD_EXT.1

TSS

The evaluator shall examine the TSS to verify that it describes how the TSF checks for updates and how it verifies those updates are genuine.

Guidance

The evaluator shall examine the operational guidance to verify that it describes how to use the TOE to check for and initiate updates to itself and how to determine that an integrity check of the update has failed.

Tests

The evaluator shall check for an update using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require installing and temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update.

The evaluator is also to ensure that the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SigVer. The digital signature verification may be performed as part of a network protocol that uses a trusted channel as described in FTP_ITC_EXT.1. If the signature verification is not performed as part of a trusted channel, the evaluator shall send a query response with a bad signature and verify that the signature verification fails. The evaluator shall then send a query response with a good signature and verify that the signature verification is successful.

For the following tests, the evaluator shall initiate the download of an update and capture the update prior to installation. The download could originate from the vendor's website, an enterprise-hosted update repository, or another system (e.g. network peer). All update mechanisms supported by the TOE must be evaluated. However, this only includes those mechanisms for which the TSF is providing a trusted installation and update functionality. It does not include user or administrator-driven download and installation of arbitrary files.

- *Test FPT_TUD_EXT.1:1: The evaluator shall ensure that the update has a digital signature belonging to the vendor prior to its installation. The evaluator shall modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator shall ensure that the OS does not install the modified update.*
- *Test FPT_TUD_EXT.1:2: The evaluator shall ensure that the update has a digital signature belonging to the vendor. The evaluator shall then attempt to install the update (or permit installation to continue). The evaluator shall ensure that the OS successfully installs the update.*

FPT_TUD_EXT.2 Integrity for Installation and Update of Application Software

FPT_TUD_EXT.2.1

The TSF shall provide the ability to check for updates to application software and shall use a digital signature scheme specified in FCS_COP.1/SigVer to validate the authenticity of the response.

Application Note: This requirement is about the ability to check for authentic updates, while the actual installation of such updates is covered by FPT_TUD_EXT.2.2. Use of the digital signature scheme ensures that an attacker cannot influence the response, regarding of whether updates are available.

FPT_TUD_EXT.2.2

The TSF shall cryptographically verify the integrity of updates to applications using a digital signature specified by FCS_COP.1/SigVer prior to installation.

Evaluation Activities

FPT_TUD_EXT.2

TSS

The evaluator shall examine the TSS to verify that it describes the TOE's ability to check for updates to application software and how the authenticity and integrity of these updates are verified.

Guidance

The evaluator shall examine the operational guidance to verify that it provides instructions for how to use the TOE to check for updates to application software, and how to determine if the check for authenticity or integrity of the update failed.

Tests

The evaluator shall check for updates to application software using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update.

The evaluator shall also ensure that the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SigVer. The digital signature verification may be performed as part of a network protocol that uses a trusted channel as described in FTP_ITC_EXT.1. If the signature verification is not performed as part of a trusted channel, the evaluator shall send a query response with a bad signature and verify that the signature verification fails. The evaluator shall then send a query response with a good signature and verify that the signature verification is successful.

The evaluator shall initiate an update to an application. This may vary depending on the application, but it could be through the application vendor's website, a commercial app store, or another system. All update mechanisms supported by the TOE must be evaluated. However, this only includes those mechanisms for which the TSF is providing a trusted installation and update functionality. It does not include user or administrator-driven download and installation of arbitrary files.

- *Test FPT_TUD_EXT.2:1: The evaluator shall ensure that the update has a digital signature which chains to the OS vendor or another trusted root managed through the OS. The evaluator shall modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator shall then attempt to install the modified update. The evaluator shall ensure that the OS does not install the modified update.*
- *Test FPT_TUD_EXT.2:2: The evaluator shall ensure that the update has a digital signature belonging to the OS vendor or another trusted root managed through the OS. The evaluator shall then attempt to install the update. The evaluator shall ensure that the OS successfully installs the update.*

FPT_W^X_EXT.1 Write XOR Execute Memory Pages

This is an optional component. However, applied modules or packages might redefine it as mandatory.

FPT_W^X_EXT.1.1

The TSF shall prevent allocation of any memory region with both write and execute permissions except for [**assignment**: *list of exceptions*].

Application Note: Requesting a memory mapping with both write and execute permissions subverts the platform protection provided by DEP. If the OS provides no exceptions (such as for just-in-time compilation), then "no exceptions" should be indicated in the assignment. Full realization of this requirement requires hardware support, but this is commonly available.

Evaluation Activities

FPT_W^X_EXT.1
TSS
The evaluator shall inspect the vendor-provided developer documentation and verify that no memory-mapping can be made with write and execute permissions except for the cases listed in the assignment.

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

Tests
The evaluator shall also perform the following tests.

- *Test FPT_W^X_EXT.1:1: The evaluator shall acquire or construct a test program which attempts to allocate memory that is both writable and executable. The evaluator shall run the program and confirm that it fails to allocate memory that is both writable and executable.*
- *Test FPT_W^X_EXT.1:2: The evaluator shall acquire or construct a test program which allocates memory that is executable and then subsequently requests additional write/modify permissions on that memory. The evaluator shall run the program and confirm that at no time during the lifetime of the process is the memory both writable and executable.*
- *Test FPT_W^X_EXT.1:3: The evaluator shall acquire or construct a test program which allocates memory that is writable and then subsequently requests additional execute permissions on that memory. The evaluator shall run the program and confirm that at no time during the lifetime of the process is the memory both writable and executable.*

5.1.8 TOE Access (FTA)

FTA_TAB.1 Default TOE access banners

This is an optional component. However, applied modules or packages might redefine it as mandatory.

FTA_TAB.1.1

Before establishing a user session, the [*TSF*] shall display an [*advisory warning*] message **regarding unauthorized use of the TSF**.

Evaluation Activities

FTA_TAB.1
TSS
The evaluator shall examine the TSS to verify that it identifies the user interfaces to the TOE and specifies that a configurable warning banner can be displayed on each interface.

Guidance
The evaluator shall verify that the operational guidance includes instructions on how to set the warning banner, and whether different interfaces to the TOE have independently-configurable banner text or share the configured value.

Tests
For each mechanism used to configure the banner as specified in the operational guidance and each user interface with which that banner is associated, the evaluator shall configure the banner to a different arbitrary text string, record that string, and access the TOE to verify that the configured banner is displayed prior to user authentication on that particular interface.

5.1.9 Trusted Path/Channels (FTP)

FTP_ITC_EXT.1 Trusted Channel Communication

FTP_ITC_EXT.1.1

The TSF shall use [**selection**:

- *TLS as conforming to the Functional Package for Transport Layer Security (TLS), version 2.1 as a [**selection**: client, server]*
- *DTLS as conforming to the Functional Package for Transport Layer Security (TLS), version 2.1 as a [**selection**: client, server]*
- *IPsec as conforming to the PP-Module for Virtual Private Network (VPN) Clients*

] and [**selection**:

- *SSH as conforming to the Functional Package for Secure Shell (SSH), version 2.0 as a [**selection**: client, server]*
- *no other protocols*

] to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: [**selection**: *audit server*, *authentication server*, *management server*, *[**assignment**: other capabilities]*] using [**selection**: *certificates as defined in Functional Package for X.509, version 1.0*, *SSH host keys as defined in Functional Package for Secure Shell (SSH), version 2.0*] that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

Application Note: The ST author must include the security functional requirements for the trusted channel protocol selected in FTP_ITC_EXT.1.1 as part of the TSF.

If TLS or DTLS is selected, the TSF must be validated against the appropriate requirements in the Functional Package for Transport Layer Security (TLS), version 2.1.

If IPsec as conforming to the PP-Module for Virtual Private Network (VPN) Clients is selected, then FDP_IFC_EXT.1 must be included in the ST.

If SSH is selected, the TSF must be validated against the Functional Package for Secure Shell (SSH), version 2.0 and the corresponding selection is expected to be made in FIA_UAU.5.1. The ST author must include the security functional requirements for the trusted channel protocol selected in FTP_ITC_EXT.1 as part of the TSF.

Claims from the Functional Package for X.509, version 1.0 are only required to the extent that they are needed to support the functionality required by the trusted protocols that are claimed.

If the TSF implements a protocol that requires the validation of a certificate presented by an external entity, FIA_X509_EXT.1 and FIA_X509_EXT.2 will be claimed, as will FIA_TSM_EXT.1 for management of the trust store.

If the TSF implements a protocol that requires the presentation of any certificates to an external entity, FIA_XCU_EXT.2 will be claimed. FIA_X509_EXT.3 will also be claimed, along with any applicable dependencies, depending on how the certificates presented by the TOE are obtained.

Evaluation Activities

FTP_ITC_EXT.1

TSS

The evaluator shall examine the TSS to verify that it describes all trusted channels implemented by the TOE and what the purpose is for each channel. The evaluator shall also check to verify that the ST claims conformance to any functional packages or PP-Modules that are needed to implement these protocols (e.g. if the TOE implements TLS, both the Functional Package for Transport Layer Security (TLS), version 2.1 and Functional Package for X.509, version 1.0 shall be included in the ST conformance claims).

Guidance

For each trusted channel, the evaluator shall verify that the operational guidance provides instructions on how to establish the trusted channel with a remote entity and how to specify any configuration options that are related to this.

Tests

The evaluator shall configure the OS to communicate with another trusted IT product as identified in the third selection. The evaluator shall monitor network traffic while the OS performs communication with each of the servers identified in the third selection. The evaluator shall ensure that for each session a trusted channel was established in conformance with the selected protocols.

FTP_TRP.1 Trusted Path

FTP_TRP.1.1

The TSF shall provide a communication path between itself and [**selection**: *remote*, *local*] users that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from [*modification*, *disclosure*].

Application Note: This requirement ensures that all remote administrative actions are protected. Authorized remote administrators must initiate all communication with the OS via a trusted path and all communication with the OS by remote administrators must be performed over this path. The data passed in this trusted communication channel is encrypted as defined in FTP_ITC_EXT.1.1.

If "local" is selected and no unprotected traffic is sent to remote users, then this requirement is met.

If "remote" is selected, the ST author must include the security functional requirements for the trusted channel protocol selected in FTP_ITC_EXT.1.1 as part of the TSF.

FTP_TRP.1.2

The TSF shall permit [**selection**: *the TSF*, *local users*, *remote users*] to initiate communication via the trusted path.

FTP_TRP.1.3

The TSF shall require use of the trusted path for [**selection**: *initial user authentication*, *[all remote administrative actions]*]

Application Note: This requirement ensures that authorized remote administrators initiate all communication with the OS via a trusted path, and that all communication with the OS by remote administrators is performed over this path. The data passed in this trusted communication channel is encrypted as defined in FTP_ITC_EXT.1.

If *"remote"* is selected in FTP_TRP.1.1, *"all remote administrative actions"* must be selected in FTP_TRP.1.3.

If *"local"* is selected in FTP_TRP.1.1, then *"initial user authentication"* must be selected in FTP_TRP.1.3.

Evaluation Activities

FTP_TRP.1
TSS
The evaluator shall examine the TSS to determine that the methods of remote or local OS administration are indicated, along with how those communications are protected. [Conditional: if "remote" is selected in FTP_TRP.1.1], the evaluator shall also confirm that all protocols listed in the TSS in support of OS 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 or initial user authentication for each supported method.

Tests
The evaluator shall also perform the following tests:

- *Test FTP_TRP.1:1: The evaluator shall ensure that communications using each remote or local administration method is tested during the course of the evaluation, setting up the connections or initial user authentication as described in the operational guidance and ensuring that communication is successful.*
- *Test FTP_TRP.1:2: [Conditional: if "remote" is selected in FTP_TRP.1.1]: 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 a remote administrative sessions without invoking the trusted path.*
- *Test FTP_TRP.1:3: [Conditional: if "remote" is selected in FTP_TRP.1.1]: The evaluator shall ensure, for each method of remote administration, the channel data is not sent in plaintext.*
- *Test FTP_TRP.1:4: [Conditional: if "remote" is selected in FTP_TRP.1.1]: The evaluator shall ensure, for each method of remote administration, modification of the channel data is detected by the OS.*

5.1.10 TOE Security Functional Requirements Rationale

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

Table 16: SFR Rationale

Threat	Addressed by	Rationale
T.NETWORK_ ATTACK	FAU_GEN.1	FAU_GEN.1 helps mitigate the threat of a network attack by logging evidence of potential malicious activity.
	FCS_CKM.1/AKG	FCS_CKM.1/AKG helps mitigate the threat of a network attack by ensuring the generation of strong keys used for trusted communications.
	FCS_CKM.1/SKG	FCS_CKM.1/SKG helps mitigate the threat of a network attack by ensuring the generation of strong keys used for trusted communications.
	FCS_CKM.6	FCS_CKM.6 helps mitigate the threat of a network attack by ensuring that keys used for trusted communications are destroyed in a secure manner.
	FCS_COP.1/AEAD	FCS_COP.1/AEAD helps mitigate the threat of a network attack by ensuring the use of authenticated encryption algorithms in trusted communications.
	FCS_COP.1/Hash	FCS_COP.1/Hash helps mitigate the threat of a network attack by ensuring that secure hash algorithms are used for trusted communications.
	FCS_COP.1/KeyedHash	FCS_COP.1/KeyedHash helps mitigate the threat of a network attack by ensuring that secure HMAC algorithms are used for trusted communications.

FCS_COP.1/SigGen	FCS_COP.1/SigGen helps mitigate the threat of a network attack by ensuring that secure digital signature algorithms are used for trusted communications.
FCS_COP.1/SigVer	FCS_COP.1/SigVer helps mitigate the threat of a network attack by implementing signature verification functions used for protected storage.
FCS_COP.1/SKC	FCS_COP.1/SKC helps mitigate the threat of a network attack by ensuring that secure symmetric algorithms are used for trusted communications.
FCS_RBG.1	FCS_RBG.1 helps mitigate the threat of a network attack by ensuring that keys used for trusted communications are generated using a secure DRBG.
FIA_AFL.1	FIA_AFL.1 helps mitigate the threat of a network attack by preventing an unprivileged user from logging into a network interface by brute force guessing the credential.
FIA_UAU.5	FIA_UAU.5 helps mitigate the threat of a network attack by providing specified authentication mechanisms for network user authentication.
FMT_MOF_EXT.1	FMT_MOF_EXT.1 helps mitigate the threat of a network attack by limiting the management functions that are available to a given user.
FMT_SMF_EXT.1	FMT_SMF_EXT.1 helps mitigate the threat of a network attack by limiting the management functions that are available to a given user.
FPT_ACF_EXT.1	FPT_ACF_EXT.1 helps mitigate the threat of a network attack by limiting the ability of an unprivileged user to modify the behavior of the TSF.
FPT_ASLR_EXT.1	helps mitigate the threat of a network attack by limiting the ability to modify the behavior of the TSF via memory overflow.
FPT_FLS.1	FPT_FLS.1 helps mitigate the threat of a network attack by ensuring that a malfunctioning DRBG function cannot be used to generate potentially insecure keys.
FPT_SBOP_EXT.1	helps mitigate the threat of a network attack by limiting the ability to modify the behavior of the TSF via stack overflow.
FPT_TST.1	FPT_TST.1 helps mitigate the threat of a network attack by implementing a mechanism to detect when the DRBG may be failing to generate secure cryptographic keys.
FTP_ITC_EXT.1	FTP_ITC_EXT.1 helps mitigate the threat of a network attack by requiring the TSF to implement trusted protocols for network communication.
FTP_TRP.1	FTP_TRP.1 helps mitigate the threat of a network attack by requiring the use of a trusted path for any remote administration that can be performed on the TOE.
FCS_RBG.6 (optional)	FCS_RBG.6 helps mitigate the threat of a network attack by providing a secure DRBG service for third-party applications running on the TOE which may use this service to generate their own cryptographic keys for trusted communications.
FPT_STM.1	FPT_STM.1 helps mitigate the threat of malicious activity by providing reliable time stamps for the audit trail.
FPT_W^X_EXT.1 (optional)	FPT_W^X_EXT.1 helps mitigate the threat of a network attack by enforcing data execution prevention so that an external interface cannot attempt to write data to executable memory.
FTA_TAB.1 (optional)	FTA_TAB.1 helps mitigate the threat of a network attack by providing actionable consequences for misuse of the TSF.

	FPT_BLT_EXT.1 (objective)	FPT_BLT_EXT.1 helps mitigate the threat of a network attack by enforcing least functionality of the TOE's Bluetooth interface.
	FCS_CKM.2 (implementation-dependent)	FCS_CKM.2 helps mitigate the threat of a network attack by implementing secure methods to perform key distribution for trusted communications.
	FCS_CKM_EXT.7 (implementation-dependent)	FCS_CKM_EXT.7 helps mitigate the threat of a network attack by implementing secure methods to perform key agreement for trusted communications.
	FCS_COP.1/KeyEncap (selection-based)	FCS_COP.1/KeyEncap helps mitigate the threat of a network attack by using a secure key encapsulation method to transmit a symmetric key to a third party as part of key establishment for trusted communications.
	FCS_COP.1/KeyWrap (selection-based)	FCS_COP.1/KeyWrap helps mitigate the threat of a network attack by using a secure key wrap method to distribute key material to a third party for use in trusted communications.
	FCS_COP.1/XOF (selection-based)	FCS_COP.1/XOF helps mitigate the threat of a network attack by supporting extendable output function implementations that are dependencies of some secure key generation and signature verification algorithms.
	FCS_RBG.2 (selection-based)	FCS_RBG.2 helps mitigate the threat of a network attack by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.
	FCS_RBG.3 (selection-based)	FCS_RBG.3 helps mitigate the threat of a network attack by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.
	FCS_RBG.4 (selection-based)	FCS_RBG.4 helps mitigate the threat of a network attack by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.
	FCS_RBG.5 (selection-based)	FCS_RBG.5 helps mitigate the threat of a network attack by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.
	FDP_IFC_EXT.1 (selection-based)	FDP_IFC_EXT.1 helps mitigate the threat of a network attack by ensuring that the TOE has the ability to enforce the use of an IPsec VPN for all network traffic.
T.NETWORK_EAVESDROP	FCS_CKM.1/AKG	FCS_CKM.1/AKG helps mitigate the threat of network eavesdropping by ensuring the generation of strong keys used for trusted communications.
	FCS_CKM.1/SKG	FCS_CKM.1/SKG helps mitigate the threat of network eavesdropping by ensuring the generation of strong keys used for trusted communications.
	FCS_CKM.6	FCS_CKM.6 helps mitigate the threat of network eavesdropping by ensuring that keys used for trusted communications are destroyed in a secure manner.
	FCS_COP.1/AEAD	FCS_COP.1/AEAD helps mitigate the threat of network eavesdrop by enforcing the use of a cryptographic algorithm to protect data in transit that includes assurance of both authenticity and confidentiality.
	FCS_COP.1/Hash	FCS_COP.1/Hash helps mitigate the threat of network eavesdropping by ensuring that secure hash algorithms are used for trusted communications.
	FCS_COP.1/KeyedHash	FCS_COP.1/KeyedHash helps mitigate the threat of network eavesdropping by ensuring that secure HMAC algorithms are used for trusted communications.

FCS_COP.1/SigGen	FCS_COP.1/SigGen helps mitigate the threat of network eavesdropping by ensuring that secure digital signature algorithms are used for trusted communications.
FCS_COP.1/SigVer	FCS_COP.1/SigVer helps mitigate the threat of network eavesdropping by implementing signature verification functions used for protected storage.
FCS_COP.1/SKC	FCS_COP.1/SKC helps mitigate the threat of network eavesdropping by ensuring that secure symmetric algorithms are used for trusted communications.
FCS_RBG.1	FCS_RBG.1 helps mitigate the threat of network eavesdropping by ensuring that keys used for trusted communications are generated using a secure DRBG.
FPT_FLS.1	FPT_FLS.1 helps mitigate the threat of network eavesdropping by ensuring that a malfunctioning DRBG function cannot be used to generate potentially insecure keys.
FPT_TST.1	FPT_TST.1 helps mitigate the threat of network eavesdropping by implementing a mechanism to detect when the DRBG may be failing to generate secure cryptographic keys.
FTP_ITC_EXT.1	FTP_ITC_EXT.1 helps mitigate the threat of network eavesdropping by requiring the TSF to implement trusted protocols for network communication.
FTP_TRP.1	FTP_TRP.1 helps mitigate the threat of network eavesdropping by requiring the use of a trusted path for any remote administration that can be performed on the TOE.
FCS_RBG.6 (optional)	FCS_RBG.6 helps mitigate the threat of network eavesdropping by providing a secure DRBG service for third-party applications running on the TOE which may use this service to generate their own cryptographic keys for trusted communications.
FPT_BLT_EXT.1 (objective)	FPT_BLT_EXT.1 helps mitigate the threat of network eavesdropping by enforcing least functionality of the TOE's Bluetooth interface.
FCS_CKM.2 (implementation-dependent)	FCS_CKM.2 helps mitigate the threat of network eavesdropping by enforcing the use of a cryptographic algorithm to protect key data that is distributed in transit to a third party for use in trusted communications.
FCS_CKM_EXT.7 (implementation-based)	FCS_CKM_EXT.7 helps mitigate the threat of network eavesdropping by ensuring that a third party cannot obtain the key used by the TOE to communicate securely with a remote entity.
FCS_COP.1/KeyEncap (selection-based)	FCS_COP.1/KeyWrap helps mitigate the threat of network eavesdrop by using a key encapsulation algorithm to protect data in transit
FCS_COP.1/KeyWrap (selection-based)	FCS_COP.1/KeyWrap helps mitigate the threat of network eavesdrop by using a key wrap algorithm to protect data in transit.
FCS_RBG.2 (selection-based)	FCS_RBG.2 helps mitigate the threat of network eavesdropping by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.
FCS_RBG.3 (selection-based)	FCS_RBG.3 helps mitigate the threat of network eavesdropping by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.
FCS_RBG.4 (selection-based)	FCS_RBG.4 helps mitigate the threat of network eavesdropping by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.

	FCS_RBG.5 (selection-based)	FCS_RBG.5 helps mitigate the threat of network eavesdropping by ensuring that the TOE's DRBG is seeded with sufficient entropy to ensure the generation of strong cryptographic keys.
	FCS_COP.1/XOF (selection-based)	FCS_COP.1/XOF helps mitigate the threat of network eavesdrop by supporting extendable output function implementations that are dependencies of algorithms used for protection of data in transit.
	FDP_IFC_EXT.1 (selection-based)	FDP_IFC_EXT.1 helps mitigate the threat of network eavesdropping by ensuring that the TOE has the ability to enforce the use of an IPsec VPN for all network traffic.
T.LOCAL_ ATTACK	FAU_GEN.1	FAU_GEN.1 helps mitigate the threat of a local attack by logging evidence of potential malicious activity.
	FCS_COP.1/Hash	FCS_COP.1/Hash helps mitigate the threat of a local attack by ensuring that secure hash algorithms are used for trusted updates.
	FCS_COP.1/KeyedHash	FCS_COP.1/KeyedHash helps mitigate the threat of a local attack by ensuring that secure HMAC algorithms are used for trusted updates.
	FCS_COP.1/SigGen	FCS_COP.1/SigGen helps mitigate the threat of a local attack by ensuring that secure digital signature algorithms are used for trusted updates.
	FCS_COP.1/SigVer	FCS_COP.1/SigVer helps mitigate the threat of local attack by implementing signature verification functions used for protected storage.
	FCS_STO_EXT.1	FCS_STO_EXT.1 helps mitigate the threat of a local attack by providing a mechanism to protect sensitive data at rest.
	FDP_ACF_EXT.1	FDP_ACF_EXT.1 helps mitigate the threat of a local attack by providing a mechanism to restrict the ability of one user account to access data owned by another user.
	FIA_AFL.1	FIA_AFL.1 helps mitigate the threat of a local attack by preventing an unprivileged user from gaining access to the TSF by brute force guessing the credential.
	FIA_UAU.5	FIA_UAU.5 helps mitigate the threat of a local attack by providing specified authentication mechanisms for user authentication.
	FMT_MOF_EXT.1	FMT_MOF_EXT.1 helps mitigate the threat of a local attack by limiting the management functions that are available to a given user.
	FMT_SMF_EXT.1	FMT_SMF_EXT.1 helps mitigate the threat of a local attack by limiting the management functions that are available to a given user.
	FPT_ACF_EXT.1	FPT_ACF_EXT.1 helps mitigate the threat of a local attack by limiting the ability of an unprivileged user to modify the behavior of the TSF.
	FPT_ASLR_EXT.1	helps mitigate the threat of a local attack by limiting the ability of an application to modify the behavior of the TSF via memory overflow.
	FPT_SBOP_EXT.1	helps mitigate the threat of a local attack by limiting the ability of an application to modify the behavior of the TSF via stack overflow.
	FPT_TST_EXT.1	helps mitigate the threat of a local attack by ensuring the integrity of the TSF on boot.
	FPT_TUD_EXT.1	helps mitigate the threat of a local attack by ensuring the authenticity and integrity of updates applied to the TOE.
	FPT_TUD_EXT.2	helps mitigate the threat of a local attack by ensuring the integrity of updates applied to applications running the TOE.

	FPT_W^X_EXT.1 (optional)	FPT_W^X_EXT.1 helps mitigate the threat of a local attack by enforcing data execution prevention so that an application cannot attempt to write data to executable memory.
	FTA_TAB.1 (optional)	FTA_TAB.1 helps mitigate the threat of a local attack by providing actionable consequences for misuse of the TSF.
	FPT_SRP_EXT.1 (objective)	FPT_SRP_EXT.1 helps mitigate the threat of a local attack by preventing the execution of unknown or untrusted software.
T.LIMITED_PHYSICAL_ACCESS	FAU_GEN.1	FAU_GEN.1 helps mitigate the threat of by logging evidence of potential malicious activity should illicit access to the TSF be gained.
	FCS_STO_EXT.1	FCS_STO_EXT.1 helps mitigate the threat by providing a mechanism to protect sensitive data at rest which prevents exfiltration of sensitive data during a limited access window.
	FIA_AFL.1	FIA_AFL.1 helps mitigate the threat by preventing an unprivileged user from gaining access to the TSF by brute force guessing the credential in a limited time window.
	FIA_UAU.5	FIA_UAU.5 helps mitigate the threat by providing specified authentication mechanisms for user authentication to prevent unauthorized access to the TOE.
	FMT_MOF_EXT.1	FMT_MOF_EXT.1 helps mitigate the threat by limiting the management functions that are available to a given user which minimizes the impact of compromise should illicit access be gained.
	FMT_SMF_EXT.1	FMT_SMF_EXT.1 helps mitigate the threat by limiting the management functions that are available to a given user which minimizes the impact of compromise should illicit access be gained.
	FPT_ACF_EXT.1	FPT_ACF_EXT.1 helps mitigate the threat by limiting the ability of an unprivileged user to modify the behavior of the TSF should illicit access be gained.

5.2 Security Assurance Requirements

The Security Functional Requirements (SFRs) in Section 5.1 Security Functional Requirements are specified to mitigate the threats defined in Section 3.1 Threats. The PP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of SARs from CC part 3 that are required in evaluations against this PP. Individual evaluation activities to be performed are specified both in Section 5 Security Requirements as well as in this section.

The general model for evaluation of TOEs against STs written to conform to this PP is as follows:
After the ST has been approved for evaluation, the ITSEF will obtain the OS, supporting environmental IT, and the administrative/user guides for the OS. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The ITSEF also performs the evaluation activities contained within Section 5 Security Requirements, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the OS. The evaluation activities that are captured in Section 5 Security Requirements also provide clarification as to what the developer needs to provide to demonstrate the OS is compliant with the PP.

5.2.1 Class ASE: Security Target

The following ASE components as defined in [CEM] are required:

- Conformance claims (ASE_CCL.1)
- Extended components definition (ASE_ECD.1)
- ST introduction (ASE_INT.1)
- Security objectives for the operational environment (ASE_OBJ.1)
- Direct rationale security requirements (ASE_REQ.1)
- Security problem definition (ASE_SPD.1)

- TOE summary specification (ASE_TSS.1)

The requirements for exact conformance of the Security Target are described in Section 2 Conformance Claims.

5.2.2 Class ADV: Development

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

ADV_FSP.1 Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TSFIs. It is not necessary to have a formal or complete specification of these interfaces. Additionally, because OSes conforming to this PP will necessarily have interfaces to the operational environment that are not directly invokable by OS 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 evaluation 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.

ADV_FSP.1.2D

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 is comprised of 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 evaluation activities in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

Content and presentation elements:

ADV_FSP.1.1C

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

ADV_FSP.1.2C

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

ADV_FSP.1.3C

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

ADV_FSP.1.4C

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

Evaluator action elements:

ADV_FSP.1.1E

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

ADV_FSP.1.2E

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

Evaluation Activities

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

5.2.3 Class AGD: Guidance Documentation

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

AGD_OPE.1 Operational User Guidance (AGD_OPE.1)

Developer action elements:

AGD_OPE.1.1D

The developer shall provide operational user guidance.

Application Note: The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages. Rather than repeat information here, the developer should review the evaluation activities for this component to ascertain the specifics of the guidance that the evaluator shall 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 **OS** 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 **OS** (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AGD_OPE.1.6C

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

AGD_OPE.1.7C

The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.1E

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

Evaluation Activities

AGD_OPE.1
Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements and evaluation of the OS according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the OS, the operational guidance ill contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the OS. It will provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the OS. The documentation must describe the process for verifying updates to the OS by verifying a digital signature – this may be done by the OS or the underlying platform. The evaluator shall 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 OS (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 OS will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance will 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 **OS**, including its preparative procedures.

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

Content and presentation elements:

AGD_PRE.1.1C

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered **OS** 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 **OS** 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 **OS** 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 OS functional requirements. The evaluator will check to ensure that the guidance provided for the OS adequately addresses all platforms claimed for the OS in the ST.

5.2.4 Class ALC: Life-cycle Support

At the assurance level provided for OSes conformant to this PP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the OS 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 OS 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 **OS** and a reference for the **OS**.

Content and presentation elements:

ALC_CMC.1.1C

The TSF shall be labeled with a unique reference.

Application Note: Unique reference information includes:

- OS Name
- OS Version
- OS 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 shall 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 shall check the AGD guidance and OS 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 OS, the evaluator shall 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 OS and its associated evaluation evidence requirements, this component's evaluation activities are covered by the evaluation activities listed for ALC_CMC.1.

Developer action elements:

ALC_CMS.1.1D

The developer shall provide a configuration list for the **OS**.

Content and presentation elements:

ALC_CMS.1.1C

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

ALC_CMS.1.2C

The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.1E

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

Evaluation Activities

ALC_CMS.1

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

The evaluator shall 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 will 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 shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall 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 OS developer, in conjunction with any other necessary parties, to provide information as to how the end-user devices are 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 OS.

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 the OS software.

ALC_TSU_EXT.1.2C

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

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 shall 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 shall verify that this description addresses the entire application. The evaluator shall also verify that, in addition to the OS developer's process, any third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

The evaluator shall 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 OS patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days.

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

5.2.5 Class ATE: Tests

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

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

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operational) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 5.1 Security Functional Requirements being met, although some additional testing is specified for SARs in Section 5.2 Security Assurance Requirements. The evaluation activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/OS combinations that are claiming conformance to this PP. Given the scope of the OS and its associated evaluation evidence requirements, this component's evaluation activities are covered by the evaluation activities listed for ALC_CMC.1.

Developer action elements:

ATE_IND.1.1D

The developer shall provide the **OS** for testing.

Content and presentation elements:

ATE_IND.1.1C

The **TSF** 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 shall test the OS on the most current fully patched version of the platform.

Evaluation Activities

ATE_IND.1

The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator will 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 evaluation activities.

While it is not necessary to have one test case per test listed in an evaluation 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 will 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 shall not be expected to test for these vulnerabilities in the OS. 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 **OS** for testing.

Content and presentation elements:

AVA_VAN.1.1C

The **TSF** 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 **OS**.

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 **OS** is resistant to attacks performed by an attacker possessing Basic attack potential.

Evaluation Activities

AVA_VAN.1

The evaluator shall 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 - Implementation-dependent Requirements

Implementation-dependent Requirements Appendix defines requirements that must be claimed in the ST if the TOE implements particular product features. For this technology type, the following product features require the claiming of additional SFRs:

A.0.1 Bluetooth Support

A conformant TOE may implement Bluetooth functionality. If so, it is necessary for the TOE to support specific cryptographic algorithms and key sizes to support Bluetooth communications. The TOE will also claim a PP-Configuration that includes the PP-Module for Bluetooth.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM_EXT.7**

A.0.2 Key Encapsulation Support

A conformant TOE is required to implement trusted communications. Depending on the specific protocols used and the supported connection parameters for these protocols, it may be necessary to implement a key encapsulation algorithm as part of key establishment.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM.2**

A.0.3 Key Agreement Support

A conformant TOE is required to implement trusted communications. Depending on the specific protocols used and the supported connection parameters for these protocols, it may be necessary to implement one or more key agreement algorithms as part of key establishment.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM_EXT.7**

A.0.4 WLAN Support

A conformant TOE may implement WLAN client functionality. If so, it is necessary to implement authenticated encryption and key wrapping in support of secure wireless communications. The TOE will also claim a PP-Configuration that includes the PP-Module for Wireless LAN Clients.

If this feature is implemented by the TOE, the following requirements must be claimed in the ST:

- **FCS_CKM.2**

Appendix B - Extended Component Definitions

This appendix contains the definitions for all extended requirements specified in the PP.

B.1 Extended Components Table

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

Table 17: Extended Component Definitions

Functional Class	Functional Components
Cryptographic Support (FCS)	FCS_CKM_EXT Cryptographic Key Management FCS_STO_EXT Storage of Sensitive Data
Protection of the TSF (FPT)	FPT_ACF_EXT Access Controls FPT_ASLR_EXT Address Space Layout Randomization FPT_BLT_EXT Limitation of Bluetooth Profile Support FPT_SBOP_EXT Stack Buffer Overflow Protection FPT_SRP_EXT Software Restriction Policies FPT_TST_EXT Boot Integrity FPT_TUD_EXT Trusted Update FPT_W^X_EXT Write XOR Execute Memory Pages
Security Management (FMT)	FMT_MOF_EXT Management of Functions Behavior FMT_SMF_EXT Specification of Management Functions
Trusted Path/Channels (FTP)	FTP_ITC_EXT Trusted Channel Communication
User Data Protection (FDP)	FDP_ACF_EXT Access Controls for Protecting User Data FDP_IFC_EXT Information Flow Control

B.2 Extended Component Definitions

B.2.1 Cryptographic Support (FCS)

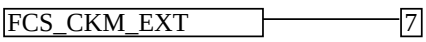
This PP defines the following extended components as part of the FCS class originally defined by CC Part 2:

B.2.1.1 FCS_CKM_EXT Cryptographic Key Management

Family Behavior

This family defines requirements for management of cryptographic keys using mechanisms beyond what are specified in CC Part 2.

Component Leveling



FCS_CKM_EXT.7, Cryptographic Key Agreement, requires that cryptographic key agreement be performed in accordance with specified standards.

Management: FCS_CKM_EXT.7

There are no management functions foreseen.

Audit: FCS_CKM_EXT.7

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP, PP-Module, functional package or ST:

- minimal: Success and failure of the activity;
- basic: The object attribute(s), and object value(s) excluding any sensitive information.

FCS_CKM_EXT.7 Cryptographic Key Agreement

Hierarchical to: No other components.

Dependencies to:

[FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation, or
FCS_CKM.5 Cryptographic key derivation, or
FCS_CKM_EXT.8 Password-based key derivation],
[FCS_CKM.2 Cryptographic key distribution, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.6 Timing and event of cryptographic key destruction
FCS_COP.1 Cryptographic operation

FCS_CKM_EXT.7.1

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

The following table provides the allowable choices for completion of the selection operations of FCS_CKM_EXT.7.

Table 18: Allowable choices for FCS_CKM_EXT.7

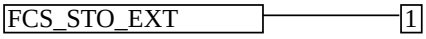
Identifier	Cryptographic algorithm	Cryptographic parameters	List of standards
KAS2	RSA	Modulus size [**selection**: *3072*, *4096*, *6144*, *8192*] bits	NIST SP 800-56B Revision 2 (Section 8.3) [KAS2]
DH	Finite Field Cryptography Diffie-Hellman	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]* • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	NIST SP 800-56A Revision 3 (Section 5.7.1.1) [DH] [**selection**: *RFC 3526 [IKE groups]*, *RFC 7919 [TLS groups]*]
ECDH	Elliptic Curve Diffie-Hellman	Elliptic Curve [**selection**: *P-256*, *P-384*, *P-521*]	NIST SP 800-56A Revision 3 (Section 5.7.1.2) [ECDH] NIST SP 800-186 (Section 3.2.1) [NIST Curves]

B.2.1.2 FCS_STO_EXT Storage of Sensitive Data

Family Behavior

Components in this family describe the requirements for storing sensitive data (such as cryptographic keys). This is a new family defined for the FCS class.

Component Leveling



FCS_STO_EXT.1, Storage of Sensitive Data, requires the TSF to include a mechanism that encrypts sensitive data and that can be invoked by third-party applications in addition to internal TSF usage.

Management: FCS_STO_EXT.1

There are no management activities foreseen.

Audit: FCS_STO_EXT.1

There are no auditable events foreseen.

FCS_STO_EXT.1 Storage of Sensitive Data

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_STO_EXT.1.1

The TSF shall implement functionality to encrypt sensitive data stored in non-volatile storage and provide interfaces to applications to invoke this functionality.

B.2.2 Protection of the TSF (FPT)

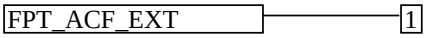
This PP defines the following extended components as part of the FPT class originally defined by CC Part 2:

B.2.2.1 FPT_ACF_EXT Access Controls

Family Behavior

This family defines specific TOE components that are protected against unprivileged access. This is a new family defined for the FPT class.

Component Leveling



FPT_ACF_EXT.1, Access Controls, requires the TSF to prohibit unauthorized users from reading or modifying specific TSF data.

Management: FPT_ACF_EXT.1

There are no management functions foreseen.

Audit: FPT_ACF_EXT.1

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

- Unauthorized attempts to perform operations against protected data

FPT_ACF_EXT.1 Access Controls

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_ACF_EXT.1.1

The TSF shall implement access controls which prohibit unprivileged users from modifying:

- Kernel and its drivers/modules
- Security audit logs
- Shared libraries

- System executables
- System configuration files
- [**assignment**: *other objects*]

.

FPT_ACF_EXT.1.2

The TSF shall implement access controls which prohibit unprivileged users from reading:

- Security audit logs
- System-wide credential repositories
- [**assignment**: *list of other objects*]

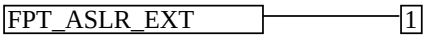
.

B.2.2.2 FPT_ASLR_EXT Address Space Layout Randomization

Family Behavior

This family defines the ability of the TOE to implement address space layout randomization (ASLR). This is a new family defined for the FPT class.

Component Leveling



FPT_ASLR_EXT.1, Address Space Layout Randomization, defines the ability of the TOE to use ASLR as well as the objects that ASLR is applied to.

Management: FPT_ASLR_EXT.1

There are no management functions foreseen.

Audit: FPT_ASLR_EXT.1

There are no auditable events foreseen.

FPT_ASLR_EXT.1 Address Space Layout Randomization

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_ASLR_EXT.1.1

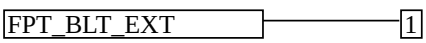
The TSF shall always randomize process address space memory locations with [**selection**: *8, [**assignment**: number greater than 8]*] bits of entropy except for [**assignment**: *list of explicit exceptions*].

B.2.2.3 FPT_BLT_EXT Limitation of Bluetooth Profile Support

Family Behavior

This family defines requirements for limiting Bluetooth capabilities without user action. This is a new family defined for the FPT class.

Component Leveling



FPT_BLT_EXT.1, Limitation of Bluetooth Profile Support, requires the TSF to maintain a disabled by default posture for Bluetooth profiles.

Management: FPT_BLT_EXT.1

There are no management activities foreseen.

Audit: FPT_BLT_EXT.1

There are no auditable events foreseen.

FPT_BLT_EXT.1 Limitation of Bluetooth Profile Support

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_BLT_EXT.1.1

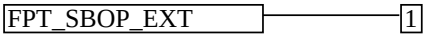
The TSF shall disable support for [**assignment**: *list of Bluetooth profiles*] Bluetooth profiles when they are not currently being used by an application on the TOE and shall require explicit user action to enable them.

B.2.2.4 FPT_SBOP_EXT Stack Buffer Overflow Protection

Family Behavior

This family requires the TSF to be compiled using stack-based buffer overflow protections. This is a new family defined for the FPT class.

Component Leveling



FPT_SBOP_EXT.1, Stack Buffer Overflow Protection, requires the TSF to be compiled using stack-based buffer overflow protections or to store data in such a manner that a stack-based buffer overflow cannot compromise the TSF.

Management: FPT_SBOP_EXT.1

There are no management activities foreseen.

Audit: FPT_SBOP_EXT.1

There are no auditable events foreseen.

FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_SBOP_EXT.1.1

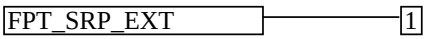
The TSF shall [**selection**: *employ stack-based buffer overflow protections*, *not store parameters or variables in the same data structures as control flow values*].

B.2.2.5 FPT_SRP_EXT Software Restriction Policies

Family Behavior

This family defines the ability of the TOE to restrict the execution of software unless it meets defined criteria. This is a new family defined for the FPT class.

Component Leveling



FPT_SRP_EXT.1, Software Restriction Policies, defines the criteria the TSF can use to prevent execution of restricted programs.

Management: FPT_SRP_EXT.1

The following actions could be considered for the management functions in FMT:

- Specification of restriction policies

Audit: FPT_SRP_EXT.1

There are no auditable events foreseen.

FPT_SRP_EXT.1 Software Restriction Policies

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_SRP_EXT.1.1

The TSF shall restrict execution to only programs which match an administrator-specified [**selection**:

- *file path*
- *file digital signature*
- *version*
- *hash*
- *[**assignment**: other characteristics]*

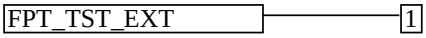
] .

B.2.2.6 FPT_TST_EXT Boot Integrity

Family Behavior

This family defines the ability of the TOE to provide a mechanism that can be used to verify its integrity when started.

Component Leveling



FPT_TST_EXT.1, Boot Integrity, defines the mechanisms that the TSF uses to assert its own integrity at startup.

Management: FPT_TST_EXT.1

There are no management functions foreseen.

Audit: FPT_TST_EXT.1

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

- Failure of the integrity checking mechanism

FPT_TST_EXT.1 Boot Integrity

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation
FIA_X509_EXT.1 X.509 Certificate Validation

FPT_TST_EXT.1.1

The TSF shall verify the integrity of the bootchain up through the OS kernel and [**selection**:

- *all executable code stored in mutable media*
- *[**assignment**: list of other executable code]*
- *no other executable code*

] prior to its execution through the use of [**selection**:

- *a digital signature using a hardware-protected asymmetric key*
- *a digital signature using an X.509 certificate with hardware-based protection*
- *a hardware-protected hash*

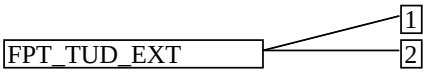
].

B.2.2.7 FPT_TUD_EXT Trusted Update

Family Behavior

This family defines the ability of the TOE to provide mechanisms for assuring the integrity of updates to the TSF or to non-TOE components that rely on the TSF to function. This is a new family defined for the FPT class.

Component Leveling



FPT_TUD_EXT.1, Integrity for Installation and Update, requires the TOE to provide a mechanism to verify the integrity of updates to itself.

FPT_TUD_EXT.2, Integrity for Installation and Update of Application Software, requires the TOE to provide a mechanism to verify the integrity of updates to non-TSF applications that are running on the TOE.

Management: FPT_TUD_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of update checking mechanism
- Initiation of update

Audit: FPT_TUD_EXT.1

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

- Failure of the integrity checking mechanism
- Successful completion of updates

FPT_TUD_EXT.1 Integrity for Installation and Update

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FPT_TUD_EXT.1.1

The TSF shall provide the ability to check for updates to the OS software itself and shall use a digital signature scheme specified in FCS_COP.1/SigVer to validate the authenticity of the response.

FPT_TUD_EXT.1.2

The TSF shall [**selection**: *cryptographically verify*, *invoke platform-provided functionality to cryptographically verify*] updates to itself using a digital signature prior to installation using schemes specified in FCS_COP.1/SigVer.

Management: FPT_TUD_EXT.2

The following actions could be considered for the management functions in FMT:

- Configuration of update checking mechanism
- Initiation of update

Audit: FPT_TUD_EXT.2

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

- Failure of the integrity checking mechanism

- Successful completion of updates

FPT_TUD_EXT.2 Integrity for Installation and Update of Application Software

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FPT_TUD_EXT.2.1

The TSF shall provide the ability to check for updates to application software and shall use a digital signature scheme specified in FCS_COP.1/SigVer to validate the authenticity of the response.

FPT_TUD_EXT.2.2

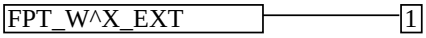
The TSF shall cryptographically verify the integrity of updates to applications using a digital signature specified by FCS_COP.1/SigVer prior to installation.

B.2.2.8 FPT_W^X_EXT Write XOR Execute Memory Pages

Family Behavior

This family defines the ability of the TOE to implement data execution prevention (DEP) by preventing memory from being both writable and executable. This is a new family defined for the FPT class.

Component Leveling



FPT_W^X_EXT.1, Write XOR Execute Memory Pages, defines the ability of the TOE to prevent memory from being simultaneously writable and executable unless otherwise specified.

Management: FPT_W^X_EXT.1

There are no management functions foreseen.

Audit: FPT_W^X_EXT.1

There are no auditable events foreseen.

FPT_W^X_EXT.1 Write XOR Execute Memory Pages

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_W^X_EXT.1.1

The TSF shall prevent allocation of any memory region with both write and execute permissions except for [**assignment**: *list of exceptions*].

B.2.3 Security Management (FMT)

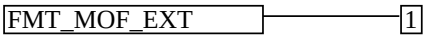
This PP defines the following extended components as part of the FMT class originally defined by CC Part 2:

B.2.3.1 FMT_MOF_EXT Management of Functions Behavior

Family Behavior

This family defines the administrative privileges required to modify the behavior of the security functions that are defined specifically for operating systems.

Component Leveling



FMT_MOF_EXT.1, Management of Functions Behavior, requires the TSF to define a set of management functions for the TOE and the privileges that are required to administer them.

Management: FMT_MOF_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of the roles that may manage the behavior of the TSF management functions

Audit: FMT_MOF_EXT.1

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

- Successful or unsuccessful management of the behavior of any TOE functions
- Change in permissions to a set of users that have the ability to manage a given function

FMT_MOF_EXT.1 Management of Functions Behavior

Hierarchical to: No other components.

Dependencies to: FMT_SMF_EXT.1 Specification of Management Functions

FMT_MOF_EXT.1.1

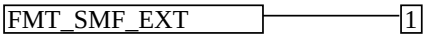
The TSF shall restrict the ability to perform the function indicated in the "Administrator" column in FMT_SMF_EXT.1.1 to the administrator.

B.2.3.2 FMT_SMF_EXT Specification of Management Functions

Family Behavior

This family defines management functions that are defined specifically for operating systems.

Component Leveling



FMT_SMF_EXT.1, Specification of Management Functions, requires the TSF to define a set of management functions for the TOE.

Management: FMT_SMF_EXT.1

There are no management functions foreseen.

Audit: FMT_SMF_EXT.1

There are no auditable events foreseen.

FMT_SMF_EXT.1 Specification of Management Functions

Hierarchical to: No other components.

Dependencies to: No dependencies.

FMT_SMF_EXT.1.1

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

#	Management Function	Administrator	User
1	Enable/disable [**selection**: *screen lock, session timeout*]	M	O
2	Configure [**selection**: *screen lock, session*] inactivity timeout	M	O
3	import keys/secrets into the secure key storage	O	O
4	Configure local audit storage capacity	O	O
5	Configure minimum password length	O	O
6	Configure minimum number of special characters in password	O	O
7	Configure minimum number of numeric characters in password	O	O
8	Configure minimum number of uppercase characters in password	O	O
9	Configure minimum number of lowercase characters in password	O	O
10	Configure lockout policy for unsuccessful authentication attempts through [**selection**: *timeouts between attempts, limiting number of attempts during a time period*]	O	O
11	Configure host-based firewall	O	O
12	Configure name/address of directory server with which to bind	O	O
13	Configure name/address of remote management server from which to receive management settings	O	O
14	Configure name/address of audit/logging server to which to send audit/logging records	O	O
15	Configure audit rules	O	O
16	Configure name/address of network time server	O	O
17	Enable/disable automatic software update	O	O
18	Configure Wi-Fi interface	O	O
19	Enable/disable Bluetooth interface	O	O
20	Enable/disable [**assignment**: *list of other external interfaces*]	O	O
21	[**assignment**: *list of other management functions to be provided by the TSF*]	O	O

.

B.2.4 Trusted Path/Channels (FTP)

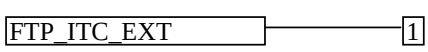
This PP defines the following extended components as part of the FTP class originally defined by CC Part 2:

B.2.4.1 FTP_ITC_EXT Trusted Channel Communication

Family Behavior

This family defines the ability of the TOE to use specific trusted communications channels to communicate with specific non-TOE entities in the Operational Environment. This family differs from FTP_ITC in Part 2 by defining technology-specific details for the implementation of these functions.

Component Leveling



FTP_ITC_EXT.1, Trusted Channel Communication, defines the specific secure communications protocols the TSF uses to communicate with a specific set of non-TOE entities in the Operational Environment.

Management: FTP_ITC_EXT.1

There are no management functions foreseen.

Audit: FTP_ITC_EXT.1

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

- Initiation of trusted channel
- Termination of trusted channel
- Failure of trusted channel functions

FTP_ITC_EXT.1 Trusted Channel Communication

Hierarchical to:	No other components.
Dependencies to:	FCS_DTLSC_EXT.1 DTLS Client Protocol FCS_IPSEC_EXT.1 IPsec FCS_SSH_EXT.1 SSH Protocol FCS_TLSC_EXT.1 TLS Client Protocol

FTP_ITC_EXT.1.1

The TSF shall use [**assignment**: *trusted protocol*], to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: [**selection**: *audit server*, *authentication server*, *management server*, ***[assignment**: other capabilities]*] using [**assignment**: *peer authentication*] that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

B.2.5 User Data Protection (FDP)

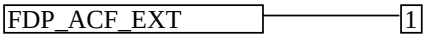
This PP defines the following extended components as part of the FDP class originally defined by CC Part 2:

B.2.5.1 FDP_ACF_EXT Access Controls for Protecting User Data

Family Behavior

This family specifies methods for ensuring that data stored or maintained by the TSF cannot be accessed without authorization. This family differs from FDP_ACF in CC Part 2 by defining technology-specific details for the implementation of these functions.

Component Leveling



FDP_ACF_EXT.1, Access Controls for Protecting User Data, requires the TSF to prevent unprivileged users from accessing operating system objects owned by other users.

Management: FDP_ACF_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of object ownership and allowed access

Audit: FDP_ACF_EXT.1

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

- Successful and unsuccessful attempts to access data

FDP_ACF_EXT.1 Access Controls for Protecting User Data

Hierarchical to: No other components.

Dependencies to: No dependencies.

FDP_ACF_EXT.1.1

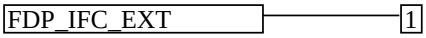
The TSF shall implement access controls which can prohibit unprivileged users from accessing files and directories owned by other users.

B.2.5.2 FDP_IFC_EXT Information Flow Control

Family Behavior

This family defines the ability of the TSF to control information flows by ensuring that it is possible to use IPsec to encapsulate all traffic bound to or from the TOE. This family differs from FDP_IFC in CC Part 2 by defining technology-specific details for the implementation of these functions.

Component Leveling



FDP_IFC_EXT.1, Information Flow Control, requires the TSF to provide the ability to protect IP traffic using IPsec.

Management: FDP_IFC_EXT.1

There are no management activities foreseen.

Audit: FDP_IFC_EXT.1

There are no auditable events foreseen.

FDP_IFC_EXT.1 Information Flow Control

Hierarchical to: No other components.

Dependencies to: FTP_ITC_EXT.1 Trusted Channel Communication

FDP_IFC_EXT.1.1

The OS shall [**selection**:

- *provide an interface which allows a VPN client to protect all IP traffic using IPsec*
- *provide a VPN client that can protect all IP traffic using IPsec*

] with the exception of IP traffic required to establish the VPN connection and [**selection**: *signed updates directly from the OS vendor, no other traffic*] .

Appendix C - Implicitly Satisfied Requirements

This appendix lists requirements that should be considered satisfied by products successfully evaluated against this PP. These requirements are not featured explicitly as SFRs and should not be included in the ST. They are not included as standalone SFRs because it would increase the time, cost, and complexity of evaluation. This approach is permitted by [CC] Part 1, 8.3 Dependencies between components.

This information benefits systems engineering activities which call for inclusion of particular security controls. Evaluation against the PP provides evidence that these controls are present and have been evaluated.

Requirement	Rationale for Satisfaction
FIA_UAU.1 - Timing of authentication	FIA_AFL.1 implicitly requires that the OS perform all necessary actions, including those on behalf of the user who has not been authenticated, in order to authenticate; therefore it is duplicative to include these actions as a separate assignment and test.
FIA_UID.1 - Timing of identification	FIA_AFL.1 implicitly requires that the OS perform all necessary actions, including those on behalf of the user who has not been identified, in order to authenticate; therefore it is duplicative to include these actions as a separate assignment and test.
FMT_SMR.1 - Security roles	FMT_MOF_EXT.1 specifies role-based management functions that implicitly defines user and privileged accounts; therefore, it is duplicative to include separate role requirements.
FTA_SSL.1 - TSF-initiated session locking	FMT_MOF_EXT.1 defines requirements for managing session locking; therefore, it is duplicative to include a separate session locking requirement.
FTA_SSL.2 - User-initiated locking	FMT_MOF_EXT.1 defines requirements for user-initiated session locking; therefore, it is duplicative to include a separate session locking requirement.
FAU_STG.2 - Protected audit data storage	FPT_ACF_EXT.1 defines a requirement to protect audit logs; therefore, it is duplicative to include a separate protection of audit trail requirements.
FAU_GEN.2 - User identity association	FAU_GEN.1.2 explicitly requires that the OS record any user account associated with each event; therefore, it is duplicative to include a separate requirement to associate a user account with each event.
FAU_SAR.1 - Audit review	FPT_ACF_EXT.1.2 requires that audit logs (and other objects) are protected from reading by unprivileged users; therefore, it is duplicative to include a separate requirement to protect only the audit information.

Appendix D - Entropy Documentation and Assessment

This appendix describes the required supplementary information for the entropy source used by the OS.

The documentation of the entropy source should be detailed enough that, after reading, the evaluator shall thoroughly understand the entropy source and why it can be relied upon to provide sufficient entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS.

D.1 Design Description

Documentation will include the design of the entropy source as a whole, including the interaction of all entropy source components. Any information that can be shared regarding the design should also be included for any third-party entropy sources that are included in the product.

The documentation will describe the operation of the entropy source to include, 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 entropy comes from, where the entropy output 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.

If implemented, the design description will include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on will be included.

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 delivering sufficient entropy for the uses made of the RBG output (by this particular OS). This argument will include a description of the expected min-entropy rate (i.e. the minimum entropy (in bits) per bit or byte of source data) and explain that sufficient entropy is going into the OS 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.

The amount of information necessary to justify the expected min-entropy rate depends on the type of entropy source included in the product.

For developer provided entropy sources, in order to justify the min-entropy rate, it is expected that a large number of raw source bits will be collected, statistical tests will be performed, and the min-entropy rate determined from the statistical tests. While no particular statistical tests are required at this time, it is expected that some testing is necessary in order to determine the amount of min-entropy in each output.

For third-party provided entropy sources, in which the OS vendor has limited access to the design and raw entropy data of the source, the documentation will indicate an estimate of the amount of min-entropy obtained from this third-party source. It is acceptable for the vendor to "assume" an amount of min-entropy, however, this assumption must be clearly stated in the documentation provided. In particular, the min-entropy estimate must be specified and the assumption included in the ST.

Regardless of type of entropy source, the justification will also include how the DRBG is initialized with the entropy stated in the ST, for example by verifying that the min-entropy rate is multiplied by the amount of source data used to seed the DRBG or that the rate of entropy expected based on the amount of source data is explicitly stated and compared to the statistical rate. If the amount of source data used to seed the DRBG is not clear or the calculated rate is not explicitly related to the seed, the documentation will not be considered complete.

The entropy justification will not include any data added from any third-party application or from any state saving between restarts.

D.3 Operating Conditions

The entropy rate may be affected by conditions outside the control of the entropy source itself. For example, voltage, frequency, temperature, and elapsed time after power-on are just a few of the factors that may affect the operation of the entropy source. As such, 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 will 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 will be included.

D.4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This includes a description of the health tests, the rate and conditions under which each health test is performed (e.g., at start, 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 - Validation Guidelines

This appendix contains "rules" specified by the PP Authors that indicate whether certain selections require the making of other selections in order for a Security Target to be valid. For example, selecting "HMAC-SHA-3-384" as a supported keyed-hash algorithm would require that "SHA-3-384" be selected as a hash algorithm.

This appendix contains only such "rules" as have been defined by the PP Authors, and does not necessarily represent all such dependencies in the document.

Appendix F - Acronyms

Table 19: Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
API	Application Programming Interface
app	Application
ASLR	Address Space Layout Randomization
Base-PP	Base Protection Profile
CC	Common Criteria
CEM	Common Evaluation Methodology
CESG	Communications-Electronics Security Group
CMC	Certificate Management over CMS
CMS	Cryptographic Message Syntax
CN	Common Names
cPP	Collaborative Protection Profile
CRL	Certificate Revocation List
CSA	Computer Security Act
CSP	Critical Security Parameters
DAR	Data At Rest
DEP	Data Execution Prevention
DES	Data Encryption Standard
DHE	Diffie-Hellman Ephemeral
DNS	Domain Name System
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
DT	Date/Time Vector
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm

EP	Extended Package
EST	Enrollment over Secure Transport
FIPS	Federal Information Processing Standards
FP	Functional Package
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
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OE	Operational Environment
OID	Object Identifier
OMB	Office of Management and Budget
OS	Operating System
PII	Personally Identifiable Information
PIN	Personal Identification Number
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
S/MIME	Secure/Multi-purpose Internet Mail Extensions
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm

SIP	Session Initiation Protocol
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
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
VPN	Virtual Private Network
XCCDF	eXtensible Configuration Checklist Description Format
XOR	Exclusive Or

Appendix G - Bibliography

Table 20: Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and general model, CCMB-2022-11-001, CC:2022, Revision 1, November 2022. • Part 2: Security functional requirements, CCMB-2022-11-002, CC:2022, Revision 1, November 2022. • Part 3: Security assurance requirements, CCMB-2022-11-003, CC:2022, Revision 1, November 2022. • Part 4: Framework for the specification of evaluation methods and activities, CCMB-2022-11-004, CC:2022, Revision 1, November 2022. • Part 5: Pre-defined packages of security requirements, CCMB-2022-11-005, CC:2022, Revision 1, November 2022.
[CEM]	Common Methodology for Information Technology Security Evaluation - • Evaluation methodology, CCMB-2022-11-006, CC:2022, Revision 1, November 2022.
[CSA]	Computer Security Act of 1987, H.R. 145, June 11, 1987.
[OMB]	Reporting Incidents Involving Personally Identifiable Information and Incorporating the Cost for Security in Agency Information Technology Investments, OMB M-06-19, July 12, 2006.
[NCSC]	National Cyber Security Centre - End User Device (EUD) Security Guidance
[SHAVS]	The Secure Hash Algorithm Validation System, NIST, 22 July 2004
[x509]	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May 2008.