Protection Profile for General Purpose Operating Systems

Version: 4.4
2025-03-31
National Information Assurance Partnership

Revision History

Version	Date	Comment
4.4	2024-09-06	• Updated to conform to CC:2022. • Incorporated X.509 package. • Incorporated applicable technical decisions.
4.3	2022-09-27	• Added compatibility with MDM Agent, Bluetooth, and TLS Modules. • Two factor authentication. • Aligned with CNSA.
4.2.1	2019-04-22	Formatting changes as a result of PP evaluation
4.2	2018-05-22	Multiple Technical Decisions applied
4.1	2016-03-09	Minor updates - cryptographic modes
4.0	2015-08-14	Release - significant revision

Contents

1 Introduction
1.1 Overview
1.2 Terms
1.2.1 Common Criteria Terms
1.2.2 Technical Terms
1.3 Compliant Targets of Evaluation
1.3.1 TOE Boundary
1.3.2 TOE Platform
1.4 Use Cases
2 Conformance Claims
3 Security Problem Definition
3.1 Threats
3.2 Assumptions
4 Security Objectives
4.1 Security Objectives for the Operational Environment
4.2 Security Objectives Rationale
5 Security Requirements
5.1 Security Functional Requirements
5.1.1 Auditable Events for Mandatory SFRs
5.1.2 Audit Data Generation (FAU)
5.1.3 Cryptographic Support (FCS)
5.1.4 User Data Protection (FDP)
5.1.5 Identification and Authentication (FIA)
5.1.6 Security Management (FMT)
5.1.7 Protection of the TSF (FPT)
5.1.8 Trusted Path/Channels (FTP)
5.1.9 TOE Security Functional Requirements Rationale
5.2 Security Assurance Requirements
5.2.1 Class ASE: Security Target
5.2.2 Class ADV: Development
5.2.3 Class AGD: Guidance Documentation
5.2.4 Class ALC: Life-cycle Support
5.2.5 Class ATE: Tests
5.2.6 Class AVA: Vulnerability Assessment
Appendix A - Optional Requirements
A.1 Strictly Optional Requirements
A.1.1 Auditable Events for Strictly Optional Requirements
A.1.2 Class ALC: Life-cycle Support
A.1.3 Cryptographic Support (FCS)
A.1.4 Protection of the TSF (FPT)
A.1.5 TOE Access (FTA)
A.2 Objective Requirements
A.2.1 Auditable Events for Objective Requirements
A.2.2 Protection of the TSF (FPT)
A.3 Implementation-dependent Requirements
Appendix B - Selection-based Requirements
B.1 Auditable Events for Selection-based Requirements
B.2 Cryptographic Support (FCS)
B.3 User Data Protection (FDP)
Appendix C - Extended Component Definitions
C.1 Extended Components Table
C.2 Extended Component Definitions
C.2.1 Cryptographic Support (FCS)
C.2.1.1 FCS_STO_EXT Storage of Sensitive Data
C.2.2 Protection of the TSF (FPT)
C.2.2.1 FPT_ACF_EXT Access Controls

C.2.2.2 FPT_ASLR_EXT Address Space Layout Randomization
C.2.2.3 FPT_BLT_EXT Limitation of Bluetooth Profile Support
C.2.2.4 FPT_SBOP_EXT Stack Buffer Overflow Protection
C.2.2.5 FPT_SRP_EXT Software Restriction Policies
C.2.2.6 FPT_TST_EXT Boot Integrity
C.2.2.7 FPT_TUD_EXT Trusted Update
C.2.2.8 FPT_W^X_EXT Write XOR Execute Memory Pages
C.2.3 Security Management (FMT)
C.2.3.1 FMT_MOF_EXT Management of Functions Behavior
C.2.3.2 FMT_SMF_EXT Specification of Management Functions
C.2.4 Trusted Path/Channels (FTP)
C.2.4.1 FTP_ITC_EXT Trusted Channel Communication
C.2.5 User Data Protection (FDP)
C.2.5.1 FDP_ACF_EXT Access Controls for Protecting User Data
C.2.5.2 FDP_IFC_EXT Information Flow Control
Appendix D - Implicitly Satisfied Requirements
Appendix E - Entropy Documentation and Assessment
E.1 Design Description
E.2 Entropy Justification
E.3 Operating Conditions
E.4 Health Testing
Appendix F - Validation Guidelines
Appendix G - Acronyms
Appendix H - Bibliography

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.
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.

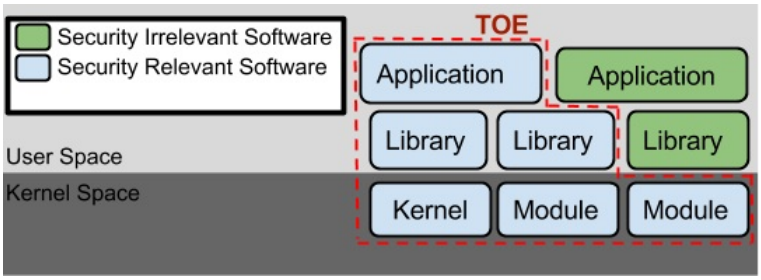


Figure 1: General TOE

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*.

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 2.4
- PP-Module for Virtual Private Network (VPN) Clients, version 2.5
- PP-Module for Bluetooth, version 1.0
- PP-Module for Mobile Device Management Agent, version 1.0
- PP-Module for Wireless LAN Clients, version 1.0
- 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 1.1 conformant.
- This PP is Functional Package for X.509 Version 1.0 conformant.
- This PP is Assurance Package for Flaw Remediation Version 1.0 conformant.

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	No events specified	N/A
FCS_CKM.2	No events specified	N/A
FCS_CKM.6	No events specified	N/A
FCS_COP.1/ENCRYPT	No events specified	N/A
FCS_COP.1/HASH	No events specified	N/A
FCS_COP.1/KEYHMAC	No events specified	N/A
FCS_COP.1/SIGN	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_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 1.0;*
- *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
TBD

Guidance
The evaluator shall check the administrative guide and ensure that it lists all of the auditable events. The evaluator shall check to make sure that every audit event type selected in the ST is included.

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 Cryptographic Key Generation

FCS_CKM.1.1

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

- *RSA schemes using a cryptographic key size of 3072-bits that meet the following: FIPS PUB 186-5, "Digital Signature Standard (DSS)", Appendix A.1*
- *ECC schemes using "NIST curve" P-384 that meet the following: FIPS PUB 186-5, "Digital Signature Standard (DSS)", Appendix A.2*
- *FFC schemes using "safe-prime" groups that meet the following: 'NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"', and [**selection**: RFC 3526, RFC 7919]*

].

Application Note: The ST author will select all key generation schemes used for key establishment and entity authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2 and selected cryptographic protocols must match the selection. When key generation is used for entity authentication, the public key is expected to be associated with an X.509v3 certificate.
If the OS acts only as a receiver in the RSA key establishment scheme, the OS does not need to implement RSA key generation.

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

Evaluation Activities

FCS_CKM.1
TSS
The evaluator shall ensure that the TSS identifies the key sizes supported by the OS. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. If “P-256” is selected, the evaluator shall examine the TSS to verify that it is only used for Bluetooth functions.

Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure the OS to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

Tests
Evaluation Activity Note: The following tests may require the vendor to furnish a developer environment and developer tools that are typically not available to end-users of the OS.

The following content should be included if:

- *RSA schemes is selected from FCS_CKM.1.1*

The following content should be included if:
- *the TOE implements ""*

Key Generation for FIPS PUB 186-5 RSA Schemes
The evaluator shall verify the implementation of RSA Key Generation by the OS using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

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

- *Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes*
- *Primes p1, p2, q1,q2, p and q shall all be probable primes*

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

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

The following content should be included if:

- *the TOE implements ""*

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-5 ECC Key Generation Test

For the supported NIST curve P-384, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-5 Public Key Verification (PKV) Test

For the supported NIST curve P-384, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

The following content should be included if:

- *the TOE implements ""*

Key Generation for Finite-Field Cryptography (FFC)

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

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

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

and two ways to generate the cryptographic group generator g:

- *Cryptographic Group Generator:*
  - *Generator g constructed through a verifiable process*
  - *Generator g constructed through an unverifiable process*

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

- *Private Key:*
  - *len(q) bit output of RBG where* $1 \leq x \leq q-1$
  - *len(q) + 64 bit output of RBG, followed by a mod q-1 operation where* $1 \leq x \leq q-1$

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm:

- *g != 0,1*
- *q divides p-1*
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

for each FFC parameter set and key pair.

FCS_CKM.2 Cryptographic Key Establishment

FCS_CKM.2.1

The TSF shall **implement functionality to perform cryptographic key establishment** in accordance with a specified cryptographic key **establishment** method: [**selection**:

- *Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"*
- *Finite field-based key establishment schemes that meets NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"*

].

Application Note: The ST author will select all key establishment schemes used for the selected cryptographic protocols.

The elliptic curves used for the key establishment scheme shall correlate with the curves specified in FCS_CKM.1.1. The domain parameters used for the finite field-based key establishment scheme are specified by the key generation according to FCS_CKM.1.1. The finite field-based key establishment schemes that conform to NIST SP 800-56A Revision 3 correspond to the "safe-prime" groups selection in FCS_CKM.1.1.

Validation Guidelines:

Rule #2

Rule #3

Evaluation Activities

FCS_CKM.2
TSS
TBD

Guidance
TBD

Tests
The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the OS to use the selected key establishment scheme(s).

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

Key Establishment Schemes
The evaluator shall verify the implementation of the key establishment schemes supported by the OS using the applicable tests below.

The following content should be included if:

- *Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", Finite field-based key establishment schemes that meets NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" is selected from FCS_CKM.2.1*

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

Function Test
The Function test verifies the ability of the OS to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the OS's supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester will generate 10 sets of

test vectors. The data set consists of one set of domain parameter values (FCC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

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

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

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

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

Validity Test

The Validity test verifies the ability of the OS to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A Revision 3 key agreement implementation to determine which errors the OS should be able to recognize. The evaluator generates a set of 24 FCC or 30 ECC test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the OS's public/private key pairs, MAC tag, and any inputs used in the KDF, such as the other info and OS id fields.

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

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

The following content should be included if:

- *Finite field-based key establishment schemes that meets NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" is selected from FCS_CKM.2.1*

FFC Schemes using "safe-prime" groups (identified in Appendix D of SP 800-56A Revision 3)

The evaluator shall verify the correctness of the TSF's implementation of "safe-prime" groups by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses "safe-prime" groups. This test must be performed for each "safe-prime" group that each protocol uses.

FCS_CKM.6 Timing and Event of Cryptographic Key Destruction

FCS_CKM.6.1

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

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

  ]
- *For non-volatile memory that consists of [**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*
  - *the invocation of an interface provided by the underlying platform that [**selection**:*
    - *logically addresses the storage location of the key and performs a*

*[**selection**: single, [**assignment**: ST author defined multi-pass]]overwrite consisting of [**selection**: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [**assignment**: any value that does not contain any CSP]]*
- *instructs the underlying platform to destroy the abstraction that represents the key*

]

]

] .

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 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 , 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 help 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_COP.1/ENCRYPT Cryptographic Operation - Encryption/Decryption

FCS_COP.1.1/ENCRYPT

The TSF shall perform [*encryption and decryption services for data*] in accordance with a specified cryptographic algorithm **[selection:**

- ***AES-XTS (as defined in NIST SP 800-38E)***
- ***AES-CBC (as defined in NIST SP 800-38A)***
- ***AES-CTR (as defined in NIST SP 800-38A)***

] and [selection:

- ***AES Key Wrap (KW) (as defined in NIST SP 800-38F)***
- ***AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F)***
- ***AES-CCMP-256 (as defined in NIST SP 800-38C and IEEE 802.11ac-2013)***
- ***AES-GCMP-256 (as defined in NIST SP 800-38D and IEEE 802.11ac-2013)***
- ***AES-CCM (as defined in NIST SP 800-38C)***
- ***AES-GCM (as defined in NIST SP 800-38D)***
- ***no other modes***

] and cryptographic key sizes **256-bit and [selection: *128-bit, no other bit size*]**.

Application Note: AES CCMP (which uses AES in CCM as specified in SP 800-38C) becomes mandatory and must be selected if the ST includes the PP-Module for Wireless LAN Clients.

AES-CCM becomes mandatory and must be selected if the ST includes the PP-Module for Bluetooth.

For the second selection, the ST author should choose the mode or modes in which AES operates.

For the third selection, the ST author may only choose 128-bit if the ST includes the PP-Module for Bluetooth, and it may only be used specifically with AES-CCM for Bluetooth functions.

Validation Guidelines:

Rule #4

Rule #11

Evaluation Activities

FCS_COP.1/ENCRYPT
TSS
If "128-bit" is selected, the evaluator shall examine the TSS to verify that 128-bit is only used with AES-CCM for Bluetooth functions.

Guidance
The evaluator shall verify that the AGD documents contains instructions required to configure the OS to use the required modes and key sizes.

Tests
The evaluator shall execute all instructions as specified to configure the OS to the appropriate state. The evaluator shall perform all of the following tests for each algorithm implemented by the OS and used to satisfy the requirements of this PP:

The following content should be included if:

- *AES-XTS (as defined in NIST SP 800-38E) is selected from FCS_COP.1.1/ENCRYPT*

XTS-AES Test

The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

- *512 bit (for AES-256) key*
- *Three data unit (i.e., plaintext) lengths. One of the data unit lengths will be a nonzero integer multiple of 256 bits, if supported. One of the data unit lengths will be an integer multiple of 256 bits, if supported. The third data unit length will be either the longest supported data unit length or 216 bits, whichever is smaller.*

using a set of 100 (key, plaintext and 256-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

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

The following content should be included if:

- *AES-CBC (as defined in NIST SP 800-38A) is selected from FCS_COP.1.1/ENCRYPT*

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values will be 256-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the

resulting values to those obtained by submitting the same inputs to a known good implementation.

- *Test FCS_COP.1/ENCRYPT:1: To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. The plaintext values will encrypted with a 256-bit all-zeros key. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 5 ciphertext values as input and AES-CBC decryption.*
- *Test FCS_COP.1/ENCRYPT:2: To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five 256-keys and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.*
- *Test FCS_COP.1/ENCRYPT:3: To test the encrypt functionality of AES-CBC, the evaluator shall supply the a sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Key i will have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. To test the decrypt functionality of AES-CBC, the evaluator shall supply the set of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The set of key/ciphertext pairs will have 256 256-bit key/ciphertext pairs. Key i in each set will have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair will be the value that results in an all-zeros plaintext when decrypted with its corresponding key.*
- *Test FCS_COP.1/ENCRYPT:4: To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 256 plaintext values described below and obtain the ciphertext values that result from AES-CBC encryption of the given plaintext using a 256-bit key value of all zeros with an IV of all zeros. Plaintext value i in each set will have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1,256].*

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i ≤ 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext will be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i ≤10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext will be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 100 plaintext, IV, and key 3-tuples. The keys, plaintext, and IV values are each 256-bits. For each 3-tuple, 1000 iterations will be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
  if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
  else:
    CT[i] = AES-CBC-Encrypt(Key, PT)
    PT = CT[i-1]
```

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result will be compared to the result of running 1000 iterations with the same values using a known good implementation.

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

The following content should be included if:

- *AES-CTR (as defined in NIST SP 800-38A) is selected from FCS_COP.1.1/ENCRYPT*

AES-CTR Test

Known Answer Tests (KATs)

There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, initialization vector (IV), and ciphertext values shall be 256-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- *Test FCS_COP.1/ENCRYPT:5: To test the encrypt functionality, the evaluator shall supply 5 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a 256-bit key value of all zeros and an IV of all zeros. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using*

the 5 ciphertext values as input.

- *Test FCS_COP.1/ENCRYPT:6: To test the encrypt functionality, the evaluator shall supply 5 256-bit key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.*
- *Test FCS_COP.1/ENCRYPT:7: To test the encrypt functionality, the evaluator shall supply a set of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values and an IV of all zeros. The set of keys shall have shall have 256 256-bit keys. Keyi shall have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator shall supply the set of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The set of key/ciphertext pairs shall have 256 256-bit pairs. Keyi shall have the leftmost i bits be ones and the rightmost 256-i bits be zeros for i in [1, N]. The ciphertext value in each pair shall be the value that results in an all zeros plaintext when decrypted with its corresponding key.*
- *Test FCS_COP.1/ENCRYPT:8: To test the encrypt functionality, the evaluator shall supply the set of 256 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 256 bit key value of all zeros, respectively, and an IV of all zeros. Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 256-i bits be zeros, for i in [1, 256]. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.*

Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

Monte-Carlo Test

For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine.

The evaluator shall test the encrypt functionality using 100 plaintext/key pairs. Each key shall be 256-bit. The plaintext values shall be 256-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode

```
# Input: PT, Key

for i = 1 to 1000:
CT[i] = AES-ECB-Encrypt(Key, PT)
PT = CT[i]
```

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The following content should be included if:

- *AES Key Wrap (KW) (as defined in NIST SP 800-38F), AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F) is selected from FCS_COP.1.1/ENCRYPT*

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator shall test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- *256-bit key encryption keys (KEKs)*
- *Three plaintext lengths. One of the plaintext lengths will be two semi-blocks (256 bits). One of the plaintext lengths will be three semi-blocks (192 bits). The third data unit length will be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).*

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

The following content should be included if:

- *AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F) is selected from FCS_COP.1.1/ENCRYPT*

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

- *One plaintext length will be one octet. One plaintext length will be 20 octets (160 bits).*

- *One plaintext length will be the longest supported plaintext length less than or equal to 512 octets (4096 bits).*

The evaluator shall test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

The following content should be included if:

- *AES-CCMP-256 (as defined in NIST SP 800-38C and IEEE 802.11ac-2013), AES-CCM (as defined in NIST SP 800-38C) is selected from FCS_COP.1.1/ENCRYPT*

AES-CCM Tests

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

- *128-bit (if selected) and 256-bit keys*
- *Two payload lengths. One payload length will be the shortest supported payload length, greater than or equal to zero bytes. The other payload length will be the longest supported payload length, less than or equal to 32 bytes (256 bits).*
- *Two or three associated data lengths. One associated data length will be 0, if supported. One associated data length will be the shortest supported payload length, greater than or equal to zero bytes. One associated data length will be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2 16 bytes, an associated data length of 216 bytes will be tested.*
- *Nonce lengths. The evaluator shall test all nonce lengths between 7 and 13 bytes, inclusive, that are supported by the OS.*
- *Tag lengths. The evaluator shall test all of the following tag length values that are supported by the OS: 4, 6, 8, 10, 12, 14 and 16 bytes.*

To test the generation-encryption functionality of AES-CCM, the evaluator shall perform the following four tests:

- *Test FCS_COP.1/ENCRYPT:9: For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- *Test FCS_COP.1/ENCRYPT:10: For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- *Test FCS_COP.1/ENCRYPT:11: For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.*
- *Test FCS_COP.1/ENCRYPT:12: For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*

To determine correctness in each of the above tests, the evaluator shall compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator will supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator shall supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

Additionally, the evaluator shall use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AESCCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

The following content should be included if:

- *AES-GCMP-256 (as defined in NIST SP 800-38D and IEEE 802.11ac-2013) is selected from FCS_COP.1.1/ENCRYPT*

AES-GCMP Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

- *256 bit keys*
- *Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.*
- *Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.*
- *Two IV lengths. If 96 bit IV is supported, 96 bits will be one of the two IV lengths tested.*

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length will be tested at least once per set of 10. The IV value may be supplied by the evaluator or the

implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set will include five tuples that Pass and five that Fail.

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

AES-GCMP Monte Carlo Tests

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

- *256 bit keys*
- *Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.*
- *Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.*
- *Two IV lengths. If 96 bit IV is supported, 96 bits will be one of the two IV lengths tested.*

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length will be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set will include five tuples that Pass and five that Fail.

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

FCS_COP.1/HASH Cryptographic Operation - Hashing

FCS_COP.1.1/HASH

The TSF shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm **of SHA-384 and a message digest size of 384-bits** that meets the following: [*FIPS Pub 180-4*].

Application Note: The intent of this requirement is to specify the hashing function. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used.

Evaluation Activities

FCS_COP.1/HASH
TSS
TBD

Guidance
TBD

Tests
The evaluator shall check that the association of the hash function with other application cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented test MACs. The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

- *Test FCS_COP.1/HASH:1: Short Messages Test (Bit oriented Mode) - The evaluator shall generate an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text will be pseudorandomly generated. The evaluator shall compute the message digest for each of the messages and ensure that the correct result is produced when the messages are*

provided to the TSF.

- *Test FCS_COP.1/HASH:2: Short Messages Test (Byte oriented Mode) - The evaluator shall generate an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text will be pseudorandomly generated. The evaluator shall compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- *Test FCS_COP.1/HASH:3: Selected Long Messages Test (Bit oriented Mode) - The evaluator shall generate an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99·i, where 1 ≤ i ≤ m. The message text will be pseudorandomly generated. The evaluator shall compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- *Test FCS_COP.1/HASH:4: Selected Long Messages Test (Byte oriented Mode) - The evaluator shall generate an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8·99·i, where 1 ≤ i ≤ m/8. The message text will be pseudorandomly generated. The evaluator shall compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- *Test FCS_COP.1/HASH:5: Pseudorandomly Generated Messages Test - This test is for byte-oriented implementations only. The evaluator shall randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluator shall then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluator shall then ensure that the correct result is produced when the messages are provided to the TSF.*

FCS_COP.1/KEYHMAC Cryptographic Operation - Keyed-Hash Message Authentication

FCS_COP.1.1/KEYHMAC

The TSF shall perform [*keyed-hash message authentication services*] in accordance with a specified cryptographic algorithm **of SHA-384 with key sizes [assignment: *key size (in bits) used in HMAC*] and a message digest** size of 384-bits that meets the following: [*FIPS Pub 198-1 The Keyed-Hash Message Authentication Code and FIPS Pub 180-4 Secure Hash Standard*].

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.

Evaluation Activities

FCS_COP.1/KEYHMAC
TSS
TBD

Guidance
TBD

Tests
The evaluator shall perform the following activities based on the selections in the ST.

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set consists of a key and message data. The evaluator shall have the OS generate HMAC tags for these sets of test data. The resulting MAC tags will be compared against the result of generating HMAC tags with the same key using a known-good implementation.

FCS_COP.1/SIGN Cryptographic Operation - Signing

FCS_COP.1.1/SIGN

The TSF shall perform [*cryptographic signature services (generation and verification)*] in accordance with a specified cryptographic algorithm [**selection**:

- *RSA schemes using cryptographic key sizes of [**selection**: 2048-bit (for secure boot only) or greater, 3072-bit or greater]*
- *ECDSA schemes using "NIST curve" P-384 and no other curves*

] that meet the following: [*FIPS PUB 186-5, "Digital Signature Standard (DSS)"*].

Application Note: The ST author should choose the algorithm implemented to perform digital signatures; if more than one algorithm is available, this requirement should be iterated to specify the functionality. For the algorithm chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.

Evaluation Activities

FCS_COP.1/SIGN

TSS

[Conditional: If "2048-bit (for secure boot only) or greater" is selected] The evaluator shall check that the TSS documents that 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

Guidance

[Conditional: If "2048-bit (for secure boot only) or greater" is selected] The evaluator shall check that the AGD documents any configuration needed to ensure 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

Tests

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

The following content should be included if:

- *ECDSA schemes using "NIST curve" P-384 and no other curves is selected from FCS_COP.1.1/SIGN*

ECDSA Algorithm Tests

- *Test FCS_COP.1/SIGN:1: ECDSA FIPS 186-5 Signature Generation Test. For the supported NIST curve P-384 and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.*
- *Test FCS_COP.1/SIGN:2: ECDSA FIPS 186-5 Signature Verification Test. For the supported NIST curve P-384 and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall verify that 5 responses indicate success and 5 responses indicate failure.*

The following content should be included if:

- *RSA schemes using cryptographic key sizes of [**selection**: 2048-bit (for secure boot only) or greater, 3072-bit or greater] is selected from FCS_COP.1.1/SIGN*

RSA Signature Algorithm Tests

- *Test FCS_COP.1/SIGN:3: Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the OS using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the OS use its private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF' signature using a known good implementation and the associated public keys to verify the signatures.*
- *Test FCS_COP.1/SIGN:4: Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the OS to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The evaluator shall verify that the OS returns failure when validating each signature.*

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. It 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 E - 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_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/ENCRYPT.

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

Tests
TBD

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
TBD

Tests
The evaluator shall create two new standard user accounts on the system and conduct the following 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.*

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
TBD

Guidance
TBD

Tests
The evaluator shall set an administrator-configurable threshold for failed 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. 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 1.0***

] 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 1.0.

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
TBD

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	O	O

	in password		
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.

Evaluation Activities

FMT_SMF_EXT.1
TSS
TBD

Guidance
The evaluator shall 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
TBD

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
TBD

Guidance
TBD

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_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
TBD

Guidance
TBD

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, such as 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: 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_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 E - 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_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
TBD

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.).*
- *Test FPT_TST_EXT.1:3[conditional, to be performed 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/SIGN 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/SIGN.

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
TBD

Guidance
TBD

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/SIGN. The digital signature verification may be performed as part of a network protocol occurs over 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 supported origins for the update must be indicated in the TSS and evaluated.

- *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/SIGN 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/SIGN prior to installation.

Evaluation Activities

FPT_TUD_EXT.2
TSS
TBD

Guidance
TBD

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/SIGN. The digital signature verification may be performed as part of a network protocol 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 origins supported by the OS must be indicated in the TSS and evaluated. However, this only includes those mechanisms for which the OS 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.*

5.1.8 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 1.0 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 certificates as defined in [*Functional Package for X.509, version 1.0*] 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 in the main body of the ST.
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 1.0 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 in the main body of the ST.

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.

Validation Guidelines:

Rule #6

Rule #7

Rule #8

Rule #9

Rule #10

Evaluation Activities

FTP_ITC_EXT.1
TSS
TBD

Guidance
TBD

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 users access is selected and no unprotected traffic is sent to remote users, then this requirement is met. If remote users access is selected, the ST author must include the security functional requirements for the trusted channel protocol selected in FTP_ITC_EXT.1.1 in the main body of the ST.

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.*

5.1.9 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 3: 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	FCS_CKM.1 helps mitigate the threat of a network attack by ensuring the generation of strong keys used for trusted communications.
	FCS_CKM.2	FCS_CKM.2 helps mitigate the threat of a network attack by implementing secure methods to perform key establishment 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/ENCRYPT	FCS_COP.1/ENCRYPT helps mitigate the threat of a network attack by ensuring that secure symmetric algorithms are used for 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/KEYHMAC	FCS_COP.1/KEYHMAC helps mitigate the threat of a network attack by ensuring that secure HMAC algorithms are used for trusted communications.
	FCS_COP.1/SIGN	FCS_COP.1/SIGN helps mitigate the threat of a network attack by ensuring that secure digital signature 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_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_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	FCS_CKM.1 helps mitigate the threat of network eavesdropping by ensuring the generation of strong keys used for trusted communications.
	FCS_CKM.2	FCS_CKM.2 helps mitigate the threat of network eavesdropping by implementing secure methods to perform key establishment 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/ENCRYPT	FCS_COP.1/ENCRYPT helps mitigate the threat of network eavesdropping by ensuring that secure symmetric algorithms are used for trusted communications.
	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/KEYHMAC	FCS_COP.1/KEYHMAC helps mitigate the threat of network eavesdropping by ensuring that secure HMAC algorithms are used for trusted communications.
	FCS_COP.1/SIGN	FCS_COP.1/SIGN helps mitigate the threat of network eavesdropping by ensuring that secure digital signature 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_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.
	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/KEYHMAC	FCS_COP.1/KEYHMAC helps mitigate the threat of a local attack by ensuring that secure HMAC algorithms are used for trusted updates.
	FCS_COP.1/SIGN	FCS_COP.1/SIGN helps mitigate the threat of a local attack by ensuring that secure digital signature algorithms are used for trusted updates.
	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 - Optional Requirements

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

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

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

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

A.1 Strictly Optional Requirements

A.1.1 Auditable Events for Strictly Optional Requirements

Table 4: Auditable Events for Strictly Optional Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_RBG.6	No events specified	N/A
FPT_W^X_EXT.1	No events specified	N/A
FTA_TAB.1	No events specified	N/A

A.1.2 Class ALC: Life-cycle Support

ALC_FLR.1 Basic Flaw Remediation (ALC_FLR.1)

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

Developer action elements:

ALC_FLR.1.1D
The developer shall document and provide flaw remediation procedures addressed to TOE developers.

Content and presentation elements:

ALC_FLR.1.1C
The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.1.2C
The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.1.3C
The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.1.4C
The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

Evaluator action elements:

ALC_FLR.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_FLR.1
Evaluated as specified by [CEM].

ALC_FLR.2 Flaw Reporting Procedures (ALC_FLR.2)

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

Developer action elements:

ALC_FLR.2.1D
The developer shall document and provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.2.2D
The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.

ALC_FLR.2.3D
The developer shall provide flaw remediation guidance addressed to TOE users.

Content and presentation elements:

ALC_FLR.2.1C
The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.2.2C
The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.2.3C
The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.2.4C
The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.2.5C
The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.2.6C
The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users.

ALC_FLR.2.7C
The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.2.8C
The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

Evaluator action elements:

ALC_FLR.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_FLR.2
Evaluated as specified by [CEM].

ALC_FLR.3 Systematic Flaw Remediation (ALC_FLR.3)

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

Developer action elements:

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

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

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

Content and presentation elements:

ALC_FLR.3.1C

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

ALC_FLR.3.2C

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

ALC_FLR.3.3C

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

ALC_FLR.3.4C

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

ALC_FLR.3.5C

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

ALC_FLR.3.6C

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

ALC_FLR.3.7C

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

ALC_FLR.3.8C

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

ALC_FLR.3.9C

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

ALC_FLR.3.10C

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

ALC_FLR.3.11C

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

Evaluator action elements:

ALC_FLR.3.1E

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

Evaluation Activities

ALC_FLR.3
Evaluated as specified by [CEM].

A.1.3 Cryptographic Support (FCS)

FCS_RBG.6 Random Bit Generation Service

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

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.

A.1.4 Protection of the TSF (FPT)

FPT_W^X_EXT.1 Write XOR Execute Memory Pages

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
TBD

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.*

A.1.5 TOE Access (FTA)

FTA_TAB.1 Default TOE access banners

FTA_TAB.1.1

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

Evaluation Activities

FTA_TAB.1
TSS
TBD

Guidance
TBD

Tests
The evaluator shall configure the OS, per instructions in the OS manual, to display the advisory warning message "TEST TEST Warning Message TEST TEST". The evaluator shall then log out and confirm that the advisory message is displayed before logging in can occur.

A.2 Objective Requirements

A.2.1 Auditable Events for Objective Requirements

Table 5: Auditable Events for Objective Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FPT_BLT_EXT.1	No events specified	N/A
FPT_SRP_EXT.1	No events specified	N/A

A.2.2 Protection of the TSF (FPT)

FPT_BLT_EXT.1 Limitation of Bluetooth Profile Support

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_SRP_EXT.1 Software Restriction Policies

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.

- *Test FPT_SRP_EXT.1:1[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:2[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:3[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:4[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:5[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:6[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:7[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:8[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:9[conditional, to be performed 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.
- *Test FPT_SRP_EXT.1:10[conditional, to be performed 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.

A.3 Implementation-dependent Requirements

This PP does not define any Implementation-dependent requirements.

Appendix B - Selection-based Requirements

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

B.1 Auditable Events for Selection-based Requirements

Table 6: Auditable Events for Selection-based Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_RBG.2	No events specified	N/A
FCS_RBG.3	No events specified	N/A
FCS_RBG.4	No events specified	N/A
FCS_RBG.5	No events specified	N/A
FDP_IFC_EXT.1	No events specified	N/A

B.2 Cryptographic Support (FCS)

FCS_RBG.2 Random Bit Generation (External Seeding)

The inclusion of this selection-based component depends upon selection in FCS_RBG.1.2.

FCS_RBG.2.1

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

The inclusion of this selection-based component depends upon selection in FCS_RBG.1.2.

FCS_RBG.3.1

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

Application Note: 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)

The inclusion of this selection-based component depends upon selection in FCS_RBG.1.2.

FCS_RBG.4.1

The TSF shall be able to seed the 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)

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

B.3 User Data Protection (FDP)

FDP_IFC_EXT.1 Information Flow Control

The inclusion of this selection-based component depends upon selection in FTP_ITC_EXT.1.1.
This component may also be included in the ST as if optional.

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
TBD

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.*

Appendix C - Extended Component Definitions

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

C.1 Extended Components Table

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

Table 7: Extended Component Definitions

Functional Class	Functional Components
Cryptographic Support (FCS)	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

C.2 Extended Component Definitions

C.2.1 Cryptographic Support (FCS)

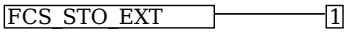
This PP defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.1 FCS_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.

C.2.2 Protection of the TSF (FPT)

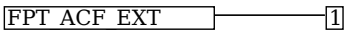
This PP defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.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*]

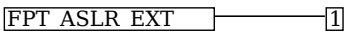
.

C.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

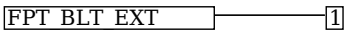
The TSF shall always randomize process address space memory locations with [**selection**: *8, [**assignment**: number greater than 8]*] bits of entropy except for [**assignment**: *list of explicit exceptions*].

C.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

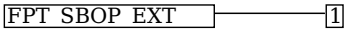
The TSF shall disable support for [**assignment**: *list of Bluetooth profiles*] Bluetooth profiles when they are not currently being used by an application on the TOE and shall require explicit user action to enable them.

C.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

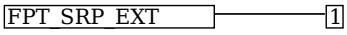
The TSF shall [**selection**: *employ stack-based buffer overflow protections, not store parameters or variables in the same data structures as control flow values*].

C.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]*

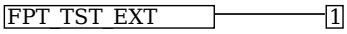
] .

C.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*

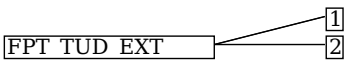
] .

C.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/SIGN 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/SIGN.

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/SIGN to validate the authenticity of the response.

FPT_TUD_EXT.2.2

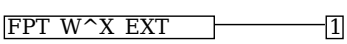
The TSF shall cryptographically verify the integrity of updates to applications using a digital signature specified by FCS_COP.1/SIGN prior to installation.

C.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*].

C.2.3 Security Management (FMT)

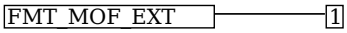
This PP defines the following extended components as part of the FMT class originally defined by CC Part 2:

C.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

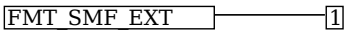
The TSF shall restrict the ability to perform the function indicated in the "Administrator" column in FMT_SMF_EXT.1.1 to the administrator.

C.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

.

C.2.4 Trusted Path/Channels (FTP)

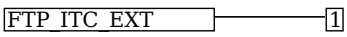
This PP defines the following extended components as part of the FTP class originally defined by CC Part 2:

C.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

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 certificates as defined in [**assignment**: *requirement or standard defining the use of certificates*] 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.

C.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:

C.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.

C.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 D - Implicitly Satisfied Requirements

This appendix lists requirements that should be considered satisfied by products successfully evaluated against this PP. These requirements are not featured explicitly as SFRs and should not be included in the ST. They are not included as standalone SFRs because it would increase the time, cost, and complexity of evaluation. This approach is permitted by [CC] Part 1, 8.3 Dependencies between components.

This information benefits systems engineering activities which call for inclusion of particular security controls. Evaluation against the PP provides evidence that these controls are present and have been evaluated.

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.
FPT_STM.1 - Reliable time stamps	FAU_GEN.1.2 explicitly requires that the OS associate timestamps with audit data; therefore it is duplicative to include a separate timestamp requirement.
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 E - 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.

E.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.

E.2 Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source 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.

E.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.

E.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 F - 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.

Rule #1

IF	
THEN	

Rule #2

IF	From FCS_CKM.2.1: * select Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”
THEN	

Rule #3

IF	From FCS_CKM.2.1: * select Finite field-based key establishment schemes that meets NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”
THEN	

Rule #4

DECISION A	**CHOICE A1** Exclude the PP-Module for Wireless LAN Clients, version 1.0 module from the ST	
	CHOICE A2 Include the PP-Module for Wireless LAN Clients, version 1.0 module in the ST	
	DECISION B	**CHOICE B1** From FCS_COP.1.1/ENCRYPT: * select AES-CCMP-256 (as defined in NIST SP 800-38C and IEEE 802.11ac-2013)
		CHOICE B2 From FCS_COP.1.1/ENCRYPT: * select AES-GCMP-256 (as defined in NIST SP 800-38D and IEEE 802.11ac-2013)

Rule #5

From the Functional Package for Transport Layer Security (TLS):
From FCS_TLS_EXT.1.1:
* select TLS as a client

Rule #6

IF	From FTP_ITC_EXT.1.1: * select TLS as conforming to the Functional Package for Transport Layer Security (TLS), version 2.1 as a [**selection**: *client*, *server*] * select server(TLS)
	From the Functional Package for Transport Layer Security (TLS):

THEN	From FCS_TLS_EXT.1.1: * select TLS as a server

Rule #7

IF	From FTP_ITC_EXT.1.1: * select DTLS as conforming to the Functional Package for Transport Layer Security (TLS), version 2.1 as a [**selection**: *client*, *server*] * select client(DTLS)

THEN	From the Functional Package for Transport Layer Security (TLS): From FCS_TLS_EXT.1.1: * select DTLS as a client

Rule #8

IF	From FTP_ITC_EXT.1.1: * select DTLS as conforming to the Functional Package for Transport Layer Security (TLS), version 2.1 as a [**selection**: *client*, *server*] * select server(DTLS)

THEN	From the Functional Package for Transport Layer Security (TLS): From FCS_TLS_EXT.1.1: * select DTLS as a server

Rule #9

IF	From FTP_ITC_EXT.1.1: * select SSH * select client(SSH)

THEN	From the Functional Package for Secure Shell (SSH): From FCS_SSH_EXT.1.1: * select client

Rule #10

IF	From FTP_ITC_EXT.1.1: * select SSH * select server(SSH)

THEN	From the Functional Package for Secure Shell (SSH): From FCS_SSH_EXT.1.1: * select server

Rule #11

DECISION C	**CHOICE C1** Exclude the PP-Module for Bluetooth, version 1.0 module from the ST
	CHOICE C2 Include the PP-Module for Bluetooth, version 1.0 module in the ST

DECISION D	**CHOICE D1** From FCS_COP.1.1/ENCRYPT: Do not choose: * AES-CCM (as defined in NIST SP 800-38C) Do not choose: * 128-bit
	CHOICE D2 From FCS_COP.1.1/ENCRYPT: * select AES-CCM (as defined in NIST SP 800-38C) * select 128-bit

Appendix G - Acronyms

Table 8: 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 H - Bibliography

Table 9: 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.