Protection Profile for Virtualization

Version: 2.0
2026-05-21
National Information Assurance Partnership

Revision History

Version	Date	Comment
1.0	2016-11-17	Initial Publication
1.1	2021-06-14	Incorporate TDs, Reference TLS Package, Add Equivalency Guidelines, etc.
1.1.1	2021-08-19	Errata release. Fixes a few typos.
2.0	2026-05-21	CC:2022 conversion, remove Modules.

Contents

Appendix H - Bibliography

1 Introduction

1.1 Overview

The scope of this Protection Profile (PP) is to describe the security functionality of virtualization technologies in terms of [CC] and to define security functional and assurance requirements for such products.

Due to the increasing prevalence of virtualization technology in enterprise computing environments and the shift to cloud computing, it is essential to ensure that this technology is implemented securely in order to mitigate the risk introduced by sharing multiple computers and their resident data across a single physical system.

1.2 Terms

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

1.2.1 Common Criteria Terms

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

Administrator	Administrators perform management activities on the VS. These management functions do not include administration of software running within Guest VMs, such as the guest OS. Administrators need not be human as in the case of embedded or headless VMs. Administrators are often nothing more than software entities that operate within the VM.
Auditor	Auditors are responsible for managing the audit capabilities of the TOE. An Auditor may also be an administrator. It is not a requirement that the TOE be capable of supporting an Auditor role that is separate from that of an administrator.
Domain	A Domain or Information Domain is a policy construct that groups together execution environments and networks by sensitivity of information and access control policy. For example, classification levels represent information domains. Within classification levels, there might be other domains representing communities of interest or coalitions. In the context of a VS, information domains are generally implemented as collections of VMs connected by virtual networks. The VS itself can be considered an Information Domain, as can its Management Subsystem.
Guest Network	See Operational Network.
Guest Operating System (OS)	An operating system that runs within a Guest VM.
Guest VM	A Guest VM is a VM that contains a virtual environment for the execution of an independent computing system. Virtual environments execute mission workloads and implement customer-specific client or server functionality in Guest VMs, such as a web server or desktop productivity applications.
Helper VM	A Helper VM is a VM that performs services on behalf of one or more Guest VMs, but does not qualify as a Service VM—and therefore is not part of the VMM. Helper VMs implement functions or services that are particular to the workloads of Guest VMs. For example, a VM that provides a virus scanning service for a Guest VM would be considered a Helper VM. For the purposes of this document, Helper VMs are considered a type of Guest VM, and are therefore subject to all the same requirements, unless specifically stated otherwise.
Host Operating System (OS)	An operating system onto which a VS is installed. Relative to the VS, the Host OS is part of the Platform. There need not be a Host OS, but often VSes employ a Host OS or Control Domain to support guest access to host resources. Sometimes these domains are themselves encapsulated within VMs.
Hypercall	An API function that allows VM-aware software running within a VM to invoke VMM functionality.
Hypervisor	The hypervisor is part of the VMM. It is the software executive of the physical platform of a VS. A hypervisor's primary function is to mediate access to all CPU and memory resources, but it is also responsible for either the direct management or the delegation of the management of all other hardware devices on the hardware platform.
Information Domain	See Domain.
Introspection	A capability that allows a specially designated and privileged domain to have visibility into another domain for purposes of anomaly detection or monitoring.
Management Network	A network, which may have both physical and virtualized components, used to manage and administer a VS. Management networks include networks used by VS administrators to communicate with management components of the VS, and networks used by the VS for communications between VS components. For purposes of this document, networks that connect physical hosts and backend storage networks for purposes of VM transfer or backup are considered management networks.
Management Subsystem	Components of the VS that allow VS administrators to configure and manage the VMM, as well as configure Guest VMs. VMM management functions include VM configuration, virtualized network configuration, and allocation of physical resources.
Operational Network	An Operational Network is a network, which may have both physical and virtualized components, used to connect Guest VMs to each other and potentially to other entities outside of the VS. Operational Networks support mission workloads and customer-specific client or server functionality. Also called a "Guest Network."

Physical Platform	The hardware environment on which a VS executes. Physical platform resources include processors, memory, devices, and associated firmware.
Platform	The hardware, firmware, and software environment into which a VS is installed and executes.
Service VM	A Service VM is a VM whose purpose is to support the hypervisor in providing the resources or services necessary to support Guest VMs. Service VMs may implement some portion of hypervisor functionality, but also may contain important system functionality that is not necessary for hypervisor operation. As with any VM, Service VMs necessarily execute without full hypervisor privileges—only the privileges required to perform its designed functionality. Examples of Service VMs include device driver VMs that manage access to physical devices, VMs that provide life-cycle management and provisioning of hypervisor and Guest VMs, and name-service VMs that help establish communication paths between VMs.
System Security Policy (SSP)	The overall policy enforced by the VS defining constraints on the behavior of VMs and users.
User	Users operate Guest VMs and are subject to configuration policies applied to the VS by administrators. Users need not be human as in the case of embedded or headless VMs, users are often nothing more than software entities that operate within the VM.
Virtual Machine (VM)	A Virtual Machine is a virtualized hardware environment in which an operating system may execute.
Virtual Machine Manager (VMM)	A VMM is a collection of software components responsible for enabling VMs to function as expected by the software executing within them. Generally, the VMM consists of a hypervisor, Service VMs, and other components of the VS, such as virtual devices, binary translation systems, and physical device drivers. It manages concurrent execution of all VMs and virtualizes platform resources as needed.
Virtualization System (VS)	A software product that enables multiple independent computing systems to execute on the same physical hardware platform without interference from one another. For the purposes of this document, the VS consists of a Virtual Machine Manager (VMM), Virtual Machine abstractions, a management subsystem, and other components.

1.3 Compliant Targets of Evaluation

A Virtualization System (VS) is a software product that enables multiple independent computing systems to execute on the same physical hardware platform without interference from one another. A VS creates a virtualized hardware environment (virtual machines or VMs) for each instance of an operating system permitting these environments to execute concurrently while maintaining isolation and the appearance of exclusive control over assigned computing resources. For the purposes of this document, the VS consists of a Virtual Machine Manager (VMM), Virtual Machine (VM) abstractions, a management subsystem, and other components.

A VMM is a collection of software components responsible for enabling VMs to function as expected by the software executing within them. Generally, the VMM consists of a hypervisor, Service VMs, and other components of the VS, such as virtual devices, binary translation systems, and physical device drivers. It manages concurrent execution of all VMs and virtualizes platform resources as needed.

The hypervisor is the software executive of the physical platform of a VS. A hypervisor operates at the highest CPU privilege level and manages access to all of the physical resources of the hardware platform. It exports a well-defined, protected interface for access to the resources it manages. A hypervisor's primary function is to mediate access to all CPU and memory resources, but it is also responsible for either the direct management or the delegation of the management of all other hardware devices on the hardware platform. This document does not specify any hypervisor-specific requirements, though many VMM requirements would naturally apply to a hypervisor.

A Service VM is a VM whose purpose is to support the hypervisor in providing the resources or services necessary to support Guest VMs. Service VMs may implement some portion of hypervisor functionality, but also may contain important system functionality that is not necessary for hypervisor operation. As with any VM, Service VMs necessarily execute without full hypervisor privileges—only the privileges required to perform its designed functionality. Examples of Service VMs include device driver VMs that manage access to physical devices, VMs that provide life-cycle management and provisioning of hypervisor and Guest VMs, and name-service VMs that help establish communication paths between VMs.

A Guest VM is a VM that contains a virtual environment for the execution of an independent computing system. Virtual environments execute mission workloads and implement customer-specific client or server functionality in Guest VMs, such as a web server or desktop productivity applications. A Helper VM is a VM that performs services on behalf of one or more Guest VMs, but does not qualify as a Service VM—and therefore is not part of the VMM. Helper VMs implement functions or services that are particular to the workloads of Guest VMs. For example, a VM that provides a virus scanning service for a Guest VM would be considered a Helper VM. The line between Helper and Service VMs can easily be blurred. For instance, a VM that implements a cryptographic function—such as an in-line encryption VM—could be identified as either a Service or Helper VM depending on the particular virtualization solution. If the cryptographic functions are necessary only for the privacy of Guest VM data in support of the Guest's mission applications, it would be proper to classify the encryption VM as a Helper. But if the encryption VM is necessary for the VMM to isolate Guest VMs, it would be proper to classify the encryption VM as a Service VM. For the purposes of this document, Helper VMs are subject to all requirements that apply to Guest VMs, unless specifically stated otherwise.

1.3.1 TOE Boundary

Figure 1 shows a greatly simplified view of a generic virtualization system and platform. TOE components are displayed in Red. Non-TOE components are in Blue. The Platform is the hardware, firmware, and software onto which the VS is installed. The VMM includes the hypervisor, Service VMs, and VM containers, but not the software that runs inside Guest VMs or Helper VMs. The Management Subsystem is part of the TOE, but may or may not be part of the VMM.

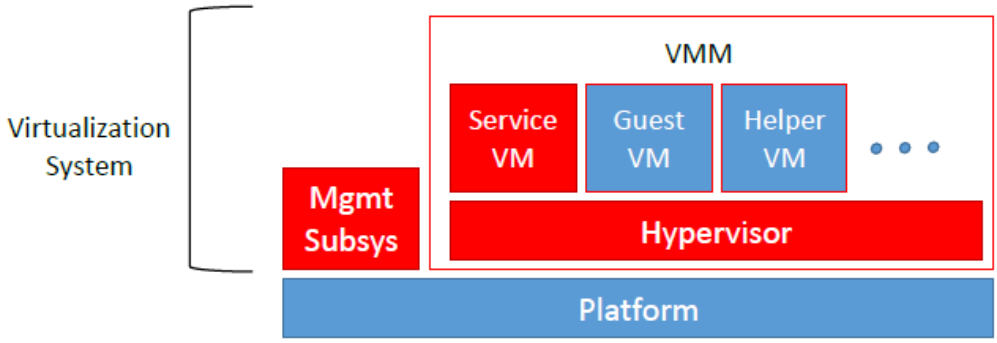


Figure 1: Virtualization System and Platform

For purposes of this Protection Profile, the virtualization system is the TOE, subject to some caveats. The Platform onto which the VS is installed (which includes hardware, platform firmware, and Host Operating System) is not part of the TOE. Software installed with the VS on the Host OS specifically to support the VS or implement VS functionality is part of the TOE. General purpose software—such as device drivers for physical devices and the Host OS itself—is not part of the TOE, regardless of whether it supports VS functionality or runs inside a Service VM or control domain. Software that runs within Guest and Helper VMs is not part of the TOE.

In general, for virtualization products that are installed onto "bare metal," the entire set of installed components constitute the TOE, and the hardware constitutes the Platform. Also in general, for products that are hosted by or integrated into a commodity operating system, the components installed expressly for implementing and supporting virtualization are in the TOE, and the Platform comprises the hardware and Host OS.

1.3.2 Requirements Met by the Platform

Depending on the way the VS is installed, functions tested under this PP may be implemented by the TOE or by the Platform. There is no difference in the testing required whether the function is implemented by the TOE or by the Platform. In either case, the tests determine whether the function being tested provides a level of confidence acceptable to meet the goals of this Profile with respect to a particular product and platform. The equivalency guidelines are intended in part to address this TOE vs. Platform distinction, and to ensure that confidence in the evaluation results do not erode between instances of equivalent products on equivalent platforms—and also, of course, to ensure that the appropriate testing is done when the distinction is significant.

1.3.3 Scope of Certification

Successful evaluation of a Virtualization System against this profile does not constitute or imply successful evaluation of any Host Operating System or Platform—no matter how tightly integrated with the VS. The Platform, including any Host OS, supports the VS through provision of services and resources. Specialized VS components installed on or in a Host OS to support the VS may be considered part of the TOE. But general-purpose OS components and functions—whether or not they support the VS—are not part of the TOE, and thus are not evaluated under this PP.

1.3.4 Product and Platform Equivalence

The tests in this Protection Profile must be run on all product versions and Platforms with which the Vendor would like to claim compliance —subject to this Profile's equivalency guidelines (see Appendix F - Equivalency Guidelines).

1.4 Use Cases

This Protection Profile supports two use cases: Server Virtualization and Client Virtualization. A product must align with one use case or the other to be evaluated against this PP.

This document does not address virtualization on mobile devices (typically devices that use a baseband processor or connect to a cellular network), nor does it address application virtualization or containers.

[USE CASE 1] Server Virtualization

Server Virtualization, for the purposes of this PP, refers to a virtualization system that implements virtualized hardware components on server-class hardware. It creates a virtualized hardware environment for each instance of an operating system (virtual machines or VMs) permitting these environments to execute concurrently while maintaining isolation and the appearance of exclusive control over assigned computing resources. Each VM instance supports applications such as file servers, web servers, and mail servers. Server virtualization may also support client operating systems in a virtual desktop or thin-client environment. Typically, virtualized servers provide services to remote clients from a data center, and are generally not directly accessible by non-administrative users.

Claiming of the Server Virtualization use case requires that the ST Author:

- Not permit Users to invoke management function 4 (Table 10),
- Not permit Users to invoke management function 9 (Table 10),
- Require that Administrators be able to invoke management function 10 (Table 10), and
- Not permit Users to invoke management function 10 (Table 10).

[USE CASE 2] Client Virtualization

Client Virtualization, for the purposes of this PP, refers to a virtualization system that implements virtualized hardware components locally on an endpoint machine. Endpoints are typically client hardware such as desktop or laptop computers that a user interacts with directly, but may also include headless embedded systems without direct human interaction. A virtualization system creates a virtualized hardware environment for each instance of a guest operating system (a virtual machine) permitting these environments to execute concurrently while maintaining isolation and the appearance of exclusive control over assigned computing resources. Client virtualization is generally used on endpoint systems, making use of the local machine's resources (memory, CPU, etc.) to provide isolated user environments.

Claiming of the Client Virtualization use case requires that the ST Author:

- Optionally allow Users to invoke management function 4 (Table 10),
- Optionally allow Users to invoke management function 9 (Table 10), and
- Optionally allow Administrators and Users to invoke management function 10 (Table 10).

1.5 Package Usage

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

Functional Package for X.509, Version 1.0

No CA Claims Permitted

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

Limitations on Signature Algorithms in FIA_X509_EXT.1.1

The TOE must utilize appropriate cryptographic algorithms that conform to CNSA standards. Thus, the TOE shall utilize no other algorithms outside of those specified in RFC 8603 for certificate or CRL signatures. Additionally, the TOE shall not use ECDSA with SHA-512 signatures for OCSP responses, and shall utilize no other algorithms for OCSP responses.

Required Extension Processing for FIA_X509_EXT.1.2

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

CRL or OCSP-based Revocation Required for FIA_X509_EXT.1.3

The TOE must use revocation involving CRL or OCSP. Accordingly, the TOE shall select only from options involving CRL or OCSP in FIA_X509_EXT.1.3.

Connections to CRL or OCSP Servers Required for FIA_X509_EXT.1.4

The TOE is required to utilize CRL or OCSP, thus it shall not select to not obtain revocation status information. Additionally, the TOE shall not utilize local revocation information or utilize a network connection to the CA to obtain its revocation information. In other words, the TOE shall utilize a network connection to a CRL distribution point, OCSP responder, OCSP stapling, or OCSP multi-stapling. However, this PP permits exceptions to revocation checking (see FIA_X509_EXT.4). If there are circumstances where the TSF does not check revocation status of a certificate such as for firmware updates, the ST author shall also select "not obtain revocation status information by the TSF due to..." and identify any functions defined in FIA_X509_EXT.4 as well as the alternative method by which revocation status is validated (e.g., if the firmware signing certificate is not checked for revocation, it is validated implicitly." The ST author shall also claim FIA_X509_EXT.4.

Restrictions on Acceptable Key Usage Values for FIA_X509_EXT.1.5

The TOE will always utilize X.509 extended key usage values to verify the proper usage of certificates for TOE functions. Accordingly, the TOE shall not pass certificate information to supported functions and shall verify certificates by evaluating the extendedKeyUsage field in the leaf certificate.
Additionally, the PP does not define SMIME functionality for X.509, therefore the TOE shall not validate certificates against any rules for SMIME certificates.

Restrictions on Acceptable Functions for FIA_X509_EXT.2.1

The PP does not define SMIME or DTLS functionality, therefore the TOE shall not select options that utilize these protocols.

Restrictions on Acceptable Purposes for Certificate Acquisition for FIA_XCU_EXT.2.1

The PP does not define SMIME or DTLS functionality, therefore the TOE shall not acquire a certificate to represent the TOE for these protocols.

1.6 Product Features Mapped to Implementation-dependent Requirements

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

1.6.1 Key Encapsulation Support

A conformant TOE is required to implement trusted communications. Depending on the specific protocols used and the supported connection parameters for these protocols, it may be necessary to implement a key encapsulation algorithm as part of key establishment. In addition to the SFR referenced here, the corresponding selection for key encapsulation shall be made in FCS_CKM.2.1, which then requires the inclusion of FCS_COP.1/KeyEncap.

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

- **FCS_CKM.2**

1.6.2 Key Wrap Support

A conformant TOE is required to implement trusted communications. Depending on the specific protocols used and the supported connection parameters for these protocols, it may be necessary to implement a key wrap algorithm as part of key establishment. In addition to the SFR referenced here, the corresponding selection for key encapsulation shall be made in FCS_CKM.2.1, which then requires the inclusion of FCS_COP.1/KeyWrap.

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

- **FCS_CKM.2**

1.6.3 Key Agreement Support

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

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

- **FCS_CKM_EXT.7**

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.

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

Package Claim

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

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

3 Security Problem Definition

3.1 Threats

T.3P_SOFTWARE
Malicious or poorly-coded third-party software or firmware that supports the functionality of the TOE (e.g., device drivers, operating systems) may be used with the TSF (e.g., physical device drivers that are not encapsulated within a VM could expose entities outside of that VM to compromise).

T.DATA_LEAKAGE
A poorly-implemented mechanism for domain separation could allow data to be maliciously or inadvertently transmitted between two VMs that should not be permitted to share information.

T.DENIAL_OF_SERVICE
A maliciously-configured or poorly-implemented VM may block other VMs from accessing system resources (e.g., system memory, persistent storage, and processing time) via a resource exhaustion attack.

T.MISCONFIGURATION
A malicious or inattentive administrator may misconfigure the VS, or faulty configuration data my cause the VS to behave in an unintended manner, which could impact its functioning and security.

T.PLATFORM_COMPROMISE
A malicious user or process may operate a VM in a manner that allows it to compromise the behavior of the system on which the VS runs (e.g., through shared access to system-level resources), leading to either unauthorized modification of the system or the behavior of the VS.

T.UNAUTHORIZED_ACCESS
A malicious user or outside entity may gain access to the TSF or its data, either through poorly implemented access control or through insecure communications, allowing for unintended management actions to be performed against the VS without proper authorization.

T.UNAUTHORIZED_MODIFICATION
A malicious user or administrator could alter the behavior of the VS by having it execute illicitly modified code, whether as an alteration of the VS itself or as code running within a VM, that causes it to behave in an unintended manner.

T.UNAUTHORIZED_UPDATE
A malicious user or process may attempt to illicitly modify the behavior of the VS by applying a software update to it that was modified in a manner that will compromise its behavior.

T.UNPATCHED_SOFTWARE
Vulnerabilities in outdated or unpatched software can be exploited by adversaries to compromise the VS or platform.

T.USER_ERROR
A careless user may inadvertently introduce information to a particular VM that is not intended to receive it because the VMs are identified in an ambiguous manner.

T.VMM_COMPROMISE
A malicious user or process may alter the behavior of the VMM or cause it to fail in a way that eliminates domain separation between multiple VMs that are not intended to be visible to one another.

T.WEAK_CRYPTO
A malicious user or process could force the disclosure of sensitive data at rest or in transit because weak cryptographic algorithms were used to protect this data or because implementation flaws in the generation and use of cryptographic data did not maximize the effective strength of the implementation.

3.2 Assumptions

A.NON_MALICIOUS_USER
The user of the VS is not willfully negligent or hostile, and uses the VS in compliance with the applied enterprise security policy and guidance. At the same time, malicious applications could act as the user, so requirements which confine malicious applications are still in scope.

A.PHYSICAL
Physical security commensurate with the value of the TOE and the data it contains is assumed to be provided by the environment.

A.PLATFORM_INTEGRITY
The platform has not been compromised prior to installation of the VS.

A.TRUSTED_ADMIN

TOE administrators are trusted to follow and apply all administrator guidance.

3.3 Organizational Security Policies

This PP defines no Organizational Security Policies.

4 Security Objectives

4.1 Security Objectives for the Operational Environment

OE.CONFIG
TOE administrators will configure the VS correctly to create the intended security policy.

OE.NON_MALICIOUS_USER
Users are trusted to not be willfully negligent or hostile and use the VS in compliance with the applied enterprise security policy and guidance.

OE.PHYSICAL
Physical security, commensurate with the value of the TOE and the data it contains, is provided by the environment.

OE.TRUSTED_ADMIN
TOE administrators are trusted to follow and apply all administrator guidance in a trusted manner.

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.NON_MALICIOUS_USER	OE.NON_MALICIOUS_USER	If the organization properly vets and trains users, it is expected that they will be non-malicious.
	OE.CONFIG	If the TOE is administered by a non-malicious and non-negligent user, the expected result is that the TOE will be configured in a correct and secure manner.
A.PHYSICAL	OE.PHYSICAL	If the TOE is deployed in a location that has appropriate physical safeguards, it can be assumed to be physically secure.
A.PLATFORM_INTEGRITY	OE.PHYSICAL	If the underlying platform has not been compromised prior to installation of the TOE, its integrity can be assumed to be intact.
A.TRUSTED_ADMIN	OE.TRUSTED_ADMIN	Providing guidance to administrators and ensuring that individuals are properly trained and vetted before being given administrative responsibilities will ensure that they are trusted.

5 Security Requirements

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

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

5.1 Security Functional Requirements

5.1.1 Auditable Events for Mandatory SFRs

Table 2: Auditable Events for Mandatory Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	On failure of logging function, capture record of failure and record upon restart of logging function.	No additional information
FAU_SAR.1	No events specified	N/A
FAU_STG.1	No events specified	N/A
FAU_STG.2	No events specified	N/A
FAU_STG.5	Failure of audit data capture due to lack of disk space or pre-defined limit.	No additional information
FCS_CKM.1/AKG	No events specified	N/A
FCS_CKM.1/SKG	No events specified	N/A
FCS_CKM.6	No events specified	N/A
FCS_COP.1/AEAD	No events specified	N/A
FCS_COP.1/Hash	No events specified	N/A
FCS_COP.1/KeyedHash	No events specified	N/A
FCS_COP.1/SigGen	No events specified	N/A
FCS_COP.1/SigVer	No events specified	N/A
FCS_ENT_EXT.1	No events specified	N/A
FCS_RBG.1	No events specified	N/A
FDP_HBI_EXT.1	No events specified	N/A
FDP_PPR_EXT.1	Successful and failed VM connections to physical devices where connection is governed by configurable policy.	VM and physical device identifiers.
	Security policy violations.	Identifier for the security policy that was violated.
FDP_RIP_EXT.1	No events specified	N/A

FDP_RIP_EXT.2	No events specified	N/A
FDP_VMS_EXT.1	No events specified	N/A
FDP_VNC_EXT.1	Successful and failed attempts to connect VMs to virtual and physical networking components.	VM and virtual or physical networking component identifiers.
	Security policy violations.	• Identifier for the security policy that was violated. • VM and virtual or physical networking component identifiers.
	Administrator configuration of inter-VM communications channels between VMs.	VM and virtual or physical networking component identifiers.
FIA_UAU.5	No events specified	N/A
FIA_UIA_EXT.1	Administrator authentication attempts.	Provided user identity, origin of the attempt (e.g., console, remote IP address).
	All use of the identification and authentication mechanism.	Provided user identity, origin of the attempt (e.g., console, remote IP address).
	[**selection, choose one of**: *Start and end of administrator session.*, *None*]	[**selection, choose one of**: *Start time and end time of administrator session*, *No additional information*]
FMT_MOF_EXT.1	Attempts to invoke any of the management functions listed in Table 10.	• Success or failure of attempt • Identity of actor
FMT_SMO_EXT.1	No events specified	N/A
FPT_DVD_EXT.1	No events specified	N/A
FPT_EEM_EXT.1	No events specified	N/A
FPT_FLS.1	Failure of the TSF.	No additional information
FPT_HAS_EXT.1	No events specified	N/A
FPT_HCL_EXT.1	[**selection, choose one of**: *Invalid parameter to hypercall detected*, *None*]	[**selection, choose one of**: *Hypercall interface for which access was attempted*, *No additional information*]
	[**selection, choose one of**: *Hypercall interface invoked when documented preconditions are not met*, *None*]	No additional information
FPT_RDM_EXT.1	Connection/disconnection of removable media or device to/from a VM.	VM Identifier, Removable media/device identifier, event description or identifier (connect/disconnect, ejection/insertion, etc.).
	Ejection/insertion of removable media or device from/to an already connected VM.	VM Identifier, Removable media/device identifier, event description or identifier (connect/disconnect, ejection/insertion, etc.).
FPT_TST.1	Indication that the TSF self-tests were completed and any failures of the tests.	No additional information
	Execution of the TSF self-tests and the results of the tests.	No additional information
FPT_TUD_EXT.1	Initiation of update.	No additional information
	Failure of signature verification.	No additional information
FPT_VDP_EXT.1	No events specified	N/A
FPT_VIV_EXT.1	No events specified	N/A

FTA_TAB.1	No events specified	N/A
FTP_ITC_EXT.1	Initiation of the trusted channel.	User ID and remote source (IP Address) if feasible.
	Termination of the trusted channel.	User ID and remote source (IP Address) if feasible.
	Failures of the trusted path functions.	User ID and remote source (IP Address) if feasible.
FTP_UIF_EXT.1	No events specified	N/A
FTP_UIF_EXT.2	No events specified	N/A

5.1.2 Class FAU: Security Audit

FAU_GEN.1 Audit Data Generation

FAU_GEN.1.1

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

a. Start-up and shutdown of audit functions
b. [**Auditable events defined in Table 2**]
c. **[selection:**
- ***Auditable events defined in Table 13 for Strictly Optional SFRs***
- ***Auditable events defined in Table 14 for Objective SFRs***
- ***Auditable events defined in Table 17 for Selection-Based SFRs***
- ***Auditable events for the Functional Package for Secure Shell (SSH), version 2.0***
- ***Auditable events for the Functional Package for Transport Layer Security (TLS), version 2.1***
- ***Auditable events for the Functional Package for X.509, version 1.0***
- ***no other auditable events***

]

FAU_GEN.1.2

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

a. Date and time of the event
b. Type of event
c. Subject and object identity (if applicable)
d. The outcome (success or failure) of the event
e. [**Additional information defined in Table 2**]
f. **[selection:**
- ***Additional information defined in Table 13 for Strictly Optional SFRs***
- ***Additional information defined in Table 14 for Objective SFRs***
- ***Additional information defined in Table 17 for Selection-Based SFRs***
- ***Additional information defined in Functional Package for Secure Shell (SSH), version 2.0***
- ***Additional information defined in Functional Package for Transport Layer Security (TLS), version 2.1***
- ***Additional information defined in Functional Package for X.509, version 1.0***
- ***no other information***

]

Application Note: “Subject identity” in FAU_GEN.1.2 could be a user id or an identifier specifying a VM, for example.

Appropriate entries from Table 13, Table 14, and Table 17 should be included in the ST if the associated SFRs and selections are included.

Each functional package includes auditable events for the SFRs defined in the package. The ST author is expected to include the auditable events (if any) for each SFR that the ST claims from a functional package. The Table 2 entry for FDP_VNC_EXT.1 refers to configuration settings that attach VMs to virtualized network components. Changes to these configurations can be made during VM execution or when VMs are not running. Audit records must be generated for either case.

The intent of the audit requirement for FDP_PPR_EXT.1 is to log that the VM is connected to a physical device (when the device becomes part of the VM's hardware view), not to log every time that the device is accessed. Generally, this is only once at VM startup. However, some devices can be connected and disconnected during operation (e.g., virtual USB devices such as CD-ROMs). All such connection/disconnection events must be logged.

Evaluation Activities

FAU_GEN.1

TSS

The evaluator shall check the TSS and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type shall be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP-Configuration is described in the TSS.

Guidance

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

Tests

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

FAU_SAR.1 Audit Review

FAU_SAR.1.1

The TSF shall provide [*administrators*] with the capability to read [*all information*] from the audit data.

FAU_SAR.1.2

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

Evaluation Activities

FAU_SAR.1

TSS

There are no additional TSS evaluation activities for this component.

Guidance

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

Tests

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

FAU_STG.1 External Audit Storage

FAU_STG.1.1

The TSF shall be able to ~~store generated audit data on the~~ [*transmit the generated audit data to an external IT entity using a trusted channel according to FTP_ITC_EXT.1*].

Evaluation Activities

FAU_STG.1
TSS
The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server (i.e., which of the trusted channels identified in FTP_ITC_EXT.1 is used), and how the trusted channel is provided.

Guidance
The evaluator shall examine the operational guidance to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

Tests
Testing of the trusted channel mechanism is to be performed as specified in the evaluation activities for FTP_ITC_EXT.1. The evaluator shall perform the following test for this requirement: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing.

FAU_STG.2 Protected Audit Trail Storage

FAU_STG.2.1

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

FAU_STG.2.2

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

Application Note: The evaluation activity for this SFR is not intended to imply that the TOE must support an administrator's ability to designate individual audit records for deletion. That level of granularity is not required.

Evaluation Activities

FAU_STG.2
TSS
The evaluator shall ensure that the TSS describes how the audit records are protected from unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.

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

Tests
The evaluator shall perform the following tests:

- *Test FAU_STG.2:1: The evaluator shall access the audit trail as an unauthorized administrator and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail.*
- *Test FAU_STG.2:2: The evaluator shall access the audit trail as an authorized administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.*

FAU_STG.5 Prevention of Audit Data Loss

FAU_STG.5.1

The TSF shall [**selection**: *ignore audited events, prevent audited events except those taken by the authorized user with special rights, overwrite the oldest stored audit records*], [**assignment**: *other actions to be taken in case of audit storage failure and conditions for the actions*] if the audit data storage is full.

Application Note: An external log server, if available, might be used as alternative storage space in case the local storage space is full. An 'other action' could be defined in this case as 'send the new audit data to an external IT entity'.

Evaluation Activities

FAU_STG.5
TSS
The evaluator shall examine the TSS to ensure it describes what happens when the local audit data store is full.

Guidance
The evaluator shall also examine the operational guidance to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and "cleared" periodically by sending the data to the audit server.

Tests
The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behavior defined in the ST for FAU_STG.5

5.1.3 Class FCS: Cryptographic Support

FCS_CKM.1/AKG Cryptographic Key Generation

FCS_CKM.1.1/AKG

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

Table 3 provides the allowable choices for completion of the selection operations of FCS_CKM.1/AKG.

Table 3: Allowable Choices for FCS_CKM.1/AKG

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

		MODP-6144, MODP-8192] • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	[**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
FFC-RS	FFC-RS - Rejection Sampling	Static domain parameters approved for [**selection**: • *IKE Groups [**selection**: MODP-3072, MODP-4096, MODP-6144, MODP-8192]* • *TLS Groups [**selection**: ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]*]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.3) [key pair generation] [**selection**: *RFC 3526 [IKE groups], RFC 7919 [TLS groups]*]
ML-KEM	ML-KEM KeyGen	Parameter set = ML-KEM-1024	NIST FIPS 203 (Section 7.1)
ML-DSA	ML-DSA KeyGen	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.1)

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

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

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

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

Evaluation Activities

FCS_CKM.1/AKG

TSS

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

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

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

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

Guidance

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

Tests

The following tests are conditional based on the selections made in the SFR. The evaluator shall perform the following tests or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug

mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

RSA Key Generation

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

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

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

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

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

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

Testing for Random Provable Primes and Conditional Methods *To test the key generation method for the random provable primes method and for all the primes with conditions methods (methods 1, 3-5), the evaluator shall seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. For each supported combination of the above input parameters, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.*

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

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

Elliptic Curve Key Generation

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

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

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

ECC Key Generation Test

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

ECC Key Validation Test

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

Finite Field Cryptography Key Generation

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

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

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

Safe Primes Generation Test

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

Safe Primes Verification Test

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

ML-KEM Key Generation

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

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

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

- *Random seed z [32 bytes]*

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

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

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

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

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

FCS_CKM.1/SKG Cryptographic Key Generation - Symmetric Key

FCS_CKM.1.1/SKG

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

Table 4 provides the allowable choices for completion of the selection operations of FCS_CKM.1/SKG.

Table 4: Allowable Choices for FCS_CKM.1/SKG

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

Evaluation Activities

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

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

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

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

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

practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described. To test the TOE's ability to generate symmetric cryptographic keys using a random bit generator, the evaluator shall configure the symmetric cryptographic key generation capability for each claimed key size. The evaluator shall use the description of the DRBG interface to verify that the TOE requests and receives an amount of DRBG output greater than or equal to the requested key size.

FCS_CKM.6 Timing and Event of Cryptographic Key Destruction

FCS_CKM.6.1

The TSF shall destroy [*all plaintext keys and keying material*] when [**selection**: *no longer needed*, *[**assignment**: other circumstances for key or keying material destruction]*].

FCS_CKM.6.2

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

- *invoking platform-provided functionality with the following rules: [**selection**:*
  - *For volatile memory, the destruction shall be executed by [**selection**:*
    - *a single direct overwrite consisting of [**selection**: a pseudo-random pattern using the TSF or platform DRBG (as specified in FCS_RBG.1, zeroes, ones, a new value of a key, [**assignment**: some value that does not contain any keying material]]*
    - *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 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]] direct overwrite consisting of [**selection**: a pseudo-random pattern using the TSF or platform DRBG (as specified in FCS_RBG.1, zeroes, ones, a new value of a key, [**assignment**: some value that does not contain any keying material]]*
    - *instructs the underlying platform to destroy the abstraction that represents the key*

    ]

  ]
- *implementing key destruction in accordance with the following rules: [**selection**:*
  - *For volatile memory, the destruction shall be executed by a [**selection**:*
    - *a pseudo-random pattern using the TSF or platform DRBG (as specified in FCS_RBG.1)*
    - *zeroes*
    - *ones*

    ]
  - *For non-volatile memory, the destruction shall be executed by a single overwrite consisting of: [**selection**:*
    - *a pseudo-random pattern using the TSF or platform DRBG (as specified in FCS_RBG.1)*
    - *zeroes*
    - *ones*

    ]

  ]

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

Application Note: For the purposes of this requirement, keying material refers to authentication data, passwords, secret/private symmetric keys, private asymmetric keys, data used to derive keys, values derived from passwords, etc. “Plaintext keying material” may refer to a KEK that is used to encrypt other keying material. Destruction of encrypted keying material may be accomplished by destroying the KEK used to encrypt it. If different mechanisms are used for destroying different keying material, all relevant claims should be selected and the TSS should identify which keying material is destroyed by which mechanism.

Key storage areas in non-volatile storage can be overwritten with any value that renders the keys unrecoverable. The value used can be all zeroes, all ones, or any other pattern or combination of values significantly different than the value of the key itself. When ‘a value that does not contain any keying

material' is chosen, it means that the TOE uses some other specified data not drawn from a source that may contain keying material or reveal information about it or any other TSF-protected data. In other words, the data used for overwriting is carefully selected and not taken from a general 'pool' that might contain current or residual data that itself requires confidentiality protection. If multiple copies exist, all copies must be destroyed.

Since this is a software-only TOE, the hardware controllers that manage non-volatile storage media are necessarily outside the TOE boundary. Thus, the TOE developer is likely to have little control over—or insight into—the functioning of these storage devices. The TOE must make a "best-effort" to destroy disused cryptographic keys by invoking the appropriate hardware interfaces—recognizing that the specific actions taken by the hardware are out of the TOE's control. But in cases where the TOE has insight into the non-volatile storage technologies used by the hardware, or where the TOE can specify a preference or method for destroying keys, the destruction should be executed by a single, direct overwrite consisting of pseudorandom data or a new key, by a repeating pattern of any static value, or by a block erase.

The interface referenced in the requirement could take different forms, the most likely of which is an API 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 where an application 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 should be described in the TSS.

Evaluation Activities

FCS_CKM.6

TSS

The evaluator shall verify that the TSS identifies all plaintext keys and keying material stored by the TOE, the type of memory in which it is stored, and when and how the keying material is erased. If the TOE uses one or more KEKs to protect stored keying material, the evaluator shall verify that the TSS describes the destruction of that keying material, either directly or by destruction of the KEK used to encrypt it.

If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting one time with a random pattern that is changed before each write"). For block erases, the evaluator shall also ensure that the TSS identifies the block erase command that is used and shall verify that the command used also addresses any copies of the plaintext key material that may be created, e.g., in order to optimize the use of Flash memory.

If platform-provided functionality is invoked for key destruction, the evaluator shall verify that the TSS identifies the platform functions used for this.

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

Evaluation Activity Note: *The following tests likely require the TOE developer to provide access to a test platform that provides the evaluator with tools that are typically not found on the end consumer version of the TOE. The evaluator shall perform the following tests for all keys and key material subject to destruction by the TOE, whether the TSF destroys this key material itself or invokes the platform to do so. For these tests, the evaluator shall utilize an appropriate development environment (e.g., a virtual machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.*

- *Test FCS_CKM.6:1: Applied to each key held as plaintext in volatile memory and subject to destruction by overwrite by the TOE (whether or not the plaintext value is subsequently encrypted for storage in volatile or non-volatile memory). 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.*
  7. *Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.*

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a minuscule chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is found the test fails.

- *Test FCS_CKM.6:2: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:*
  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 non-volatile memory the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.*
  5. *Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for test 1 above), and if a fragment is found in the repeated test then the test fails.*

- *Test FCS_CKM.6:3: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:*
  1. *Record the 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 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.

In cases where testing reveals that third-party software modules or programming language run-time environments do not properly overwrite keys, this fact must be documented. Likewise, it must be documented if there is no practical way to determine whether such modules or environments destroy keys properly. In cases where it is impossible or impracticable to perform the above tests, the evaluator shall describe how keys are destroyed in such cases, to include:

- *Which keys are affected*
- *The reasons why testing is impossible or impracticable*
- *Evidence that keys are destroyed appropriately (e.g., citations to component documentation, component developer/vendor attestation, component vendor test results)*
- *Aggravating and mitigating factors that may affect the timeliness or execution of key destruction (e.g., caching, garbage collection, operating system memory management)*

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

FCS_COP.1.1/AEAD

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

Table 5 provides the allowable choices for completion of the selection operations of FCS_COP.1/AEAD.

Table 5: Allowable choices for FCS_COP.1/AEAD

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CCM	AES in CCM mode with unpredictable, non-repeating nonce, minimum size of 64 bits	256 bits	[**selection**: *ISO/IEC 18033-3:2010 (Subclause 5.2)*, *FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 19772:2020 (Clause 7)*, *NIST SP 800-38C*] [CCM]
AES-GCM	AES in GCM mode with non-repeating IVs using [**selection**: *deterministic*, *DRBG-based*], IV construction; the tag	256 bits	[**selection**: *ISO/IEC 18033-3:2010*

	must be of length [**selection**: *96, 104, 112, 120, 128*] bits.	*(Subclause 5.2), FIPS PUB 197*] [AES] [**selection**: *ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D*] [GCM]

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

Evaluation Activities

FCS_COP.1/AEAD

TSS

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

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

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

AES-CCM

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

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

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

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

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

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

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

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

Decryption Verification Test *For each claimed key size, for each supported associated data size from 0 through 256 bits in*

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

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

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

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

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

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

Note that the IV size is always 96 bits. The evaluator shall compare the output from each test case against results generated by a known- good implementation with the same input parameters.
***Decryption Algorithm Functional Tests** The evaluator shall test the authenticated decrypt functionality of AES-GCM by supplying 15 test cases for the supported combinations of the parameters as described above. For each parameter combination the evaluator shall introduce an error into either the Ciphertext or the Tag such that approximately half of the cases are correct and half the cases contain errors.*

FCS_COP.1/Hash Cryptographic Operation (Hashing)

FCS_COP.1.1/Hash

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

Application Note: In accordance with CNSA:

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

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

Evaluation Activities

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests
The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products. The following tests are conditional, based on the selections made in the SER. The evaluator shall perform the following tests or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.
SHA-256, SHA-384, SHA-512 *To test the TOE's ability to generate hash digests using SHA2, the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Large Data Test for each claimed SHA2 algorithm.*
Algorithm Functional Test *The evaluator shall generate a number of test cases equal to the block size of the hash (512 for SHA2-256; 1024 for the other SHA2 algorithms). Each test case is to consist of random data of a random length between 0 and 65536 bits, or the largest size supported.*
Monte Carlo Test *Monte Carlo tests begin with a single seed and run 100 iterations of the chained computation. There are two versions of the Monte Carlo Test for SHA-1 and SHA-2. Either one is acceptable. For the standard Monte Carlo test the message hashed is always three times the length of the initial seed.*

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

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

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

The evaluator shall compare the output against results generated by a known good implementation with the same input.
Large Data Test *The implementation must be tested against one test case each on large data messages of 1GB, 2GB, 4GB, and 8GB of data as supported. The data need not be random. It may, for example, consist of a repeated pattern of 64 bits. The evaluator shall compare the output against results generated by a known good implementation with the same input.*
SHA3-384, SHA3-512 *To test the TOE's ability to generate hash digests using SHA3 the evaluator shall perform the Algorithm Functional Test, Monte Carlo Test, and Large Data Tests for each claimed SHA3 algorithm.*
Algorithm Functional Test *Generate a test case consisting of random data for every message length from 0 bits (or the smallest supported message size) to rate bits, where rate equals*

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

Additionally, generate tests cases of random data for messages of every multiple of (rate+1) bits starting at length rate, and continuing until 65535 is exceeded. The evaluator shall compare the output against results generated by a known good implementation with the same input.
Monte Carlo Test *Monte Carlo tests begin with a single seed and run 100 iterations of the chained computation. For this Monte Carlo Test, the hashed message is always the same length as the seed.*

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

The evaluator shall compare the output against results generated by a known good implementation with the same input.
Large Data Test *The implementation must be tested against one test case each on large data messages of 1GB, 2GB, 4GB,*

and 8GB of data as supported. The data need not be random. It may, for example, consist of a repeated pattern of 64 bits. The evaluator shall compare the output against results generated by a known good implementation with the same input.

FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithms)

FCS_COP.1.1/KeyedHash

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

Table 6 provides the allowable choices for completion of the selection operations of FCS_COP.1/KeyedHash.

Table 6: Allowable Choices for FCS_COP.1/KeyedHash

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

Application Note: The selection in this requirement must be consistent with the key size specified for the size of the keys used in conjunction with the keyed-hash message authentication.

Evaluation Activities

FCS_COP.1/KeyedHash

TSS

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

HMAC

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

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

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

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

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

- *The MAC size includes the minimum, the maximum, and two other random values.*

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

FCS_COP.1/SigGen Cryptographic Operation (Signature Generation)

FCS_COP.1.1/SigGen

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

Table 7 provides the allowable choices for completion of the selection operations in FCS_COP.1/SigGen.

Table 7: Allowable Choices for FCS_COP.1/SigGen

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection**: *3072*, *4096*, *6144*, *8192*] bits, hash [**selection**: *SHA-384*, *SHA-512*]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [**selection**: *3072*, *4096*, *6144*, *8192*] bits, hash [**selection**: *SHA-384*, *SHA-512*], Salt Length (*sLen*) such that [**assignment**: *0 ≤ sLen ≤ hLen (Hash Output Length)*] and Mask Generation Function = MGF1	RFC 8017 (Section 8.1) [PKCS#1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]
ECDSA	ECDSA	Elliptic Curve [**selection**: *P-384*, *P-521*], per-message secret number generation [**selection**: *extra random bits*, *rejection sampling*, *deterministic*] and hash function using [**selection**: *SHA-384*, *SHA-512*]	[**selection**: *ISO/IEC 14888-3:2018 (Subclause 6.6)*, *FIPS PUB 186-5 (Sections 6.3.1, 6.4.1*] [ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]
ML-DSA	ML-DSA Signature Generation	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.2)

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 and selections to specify the parameters that are implemented for that algorithm.

Evaluation Activities

FCS_COP.1/SigGen
TSS
The evaluator shall examine the TSS and verify that any hash function is the appropriate security strength for the signing algorithm.

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

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

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

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

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

To test the TOE's ability to perform RSA Digital Signature Generation using PKCS1-v1_5 signature type, the evaluator shall perform the Generated Data Test using the following input parameters:

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

Generated Data Test *For each supported combination of the above parameters, the evaluator shall cause the TOE to generate three test cases using random data. The evaluator shall compare the results against those from a known good implementation.*
RSA-PSS Signature Generation

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

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

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

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

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

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

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

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

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

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

ML-DSA Signature Generation

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

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

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

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

Known Answer Test for Rejection Cases *For each supported parameter set, the evaluator shall cause the TOE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in the fourth row for each corresponding test case below. The test values are defined as follows:*

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

ML-DSA-87 Test Cases for Rejection Cases

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

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

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

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

Test case 87-RC-05
Seed:                   9C7AD524F65854C27E565BCEDF8E86D650F13A40D0448F9AE10C05F10F777120
Hash of Keys:   0EA872CA5A4BEA94F4E8EF7ED31800727899A51059FDEE111E5CB15F0233B534
Message:                CE09831294AA96CAF684B9E667947B021C57B24C138EC7D4DA270694C82F2E08
Hash of sig:    3B9526CEE6587F2418BFE603ADB0F7DF0D69EBA31C9F9F005C60C993945EBD33

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

Test case 87-RC-07
Seed:                   E4E80CCE8B26DF1B02B99949851EE2F907FE4F0CC34790352C76D5D91634D073
```

```
Hash of Keys:   84B7E61684A12698400B09EA332EA3C4FBCFA47FE37FD6AE725CBC5FA8A99D3F
Message:                89E6AB43C9CB1CC59C3986D53217A558357E62102A26F666F2B64CD1DBB7A536
Hash of sig:    7C4AABD163CAEF8F6EBFDA3E3EEBC0A9604675B0E991ABAFD284F1AE8BA07B2A

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

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

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

Known Answer Test for Large Number of Rejection Cases (Total Rejection Count) *For each supported parameter set, the evaluator shall cause the TQE to generate signatures using the data below and a deterministic seed of all 0's. Correctness is determined by comparing the hash of the resulting signature with the hash in the fourth row of the corresponding test case below.* ***ML-DSA-87 Test Cases for Total Rejection Count***

```
Test case 87-LN-01
Seed:                   98B6298051D92BF37293C93C97370747BF527B87B71F6C4264182F45155ADE4C
Hash of Keys:   04A135B5C9B7020332C7B16E7108E8FF7FC1EAE1C23C5FA0B5D5CED0FEEE7424
Message:                D7B03412692590 83ABF3C8DC47559A19D57669B4486E0224F376DC43E577A3D8
Hash of sig:    58D72D76EC0FB65BFB9893C4479366B79DD7B8B7577E4291D13514FCC76C26DD

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

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

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

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

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

Test case 87-LN-07
Seed:                   AB01D0E591B7DDCD3C03395AED808FA2763C0A486D44119D621BE0FD0B022B25
Hash of Keys:   93B6ADE34F78A4ADB36B2F6D2C51DB793E659E1243E80488AE1C03B65125D6D7
Message:                8DE8122D89D15FE84A4C34F6B59B2C4B11F33B6A053154D199B634F557FDF5F6
Hash of sig:    0483045999A79B583F403DB96A736F0F0B24E2DFBC4E5CFA9B50E3D910786F07

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

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

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

```
Hash of sig:    46C669D2FEB643A38E54FF87B790CC33F44043A1B6B31DB9474D301328CA2A7F
```

FCS_COP.1/SigVer Cryptographic Operation - Signature Verification

FCS_COP.1.1/SigVer

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

Table 8 provides the allowable choices for completion of the selection operations in FCS_COP.1/SigVer.

Table 8: Allowable Choices for FCS_COP.1/SigVer

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [**selection**: *3072*, *4096*, *6144*, *8192*] bits and hash [**selection**: *SHA-384*, *SHA-512*]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [**selection**: *3072*, *4096*, *6144*, *8192*] bits and hash [**selection**: *SHA-384*, *SHA-512*]	RFC 8017 (Section 8.1) [PKCS#1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]
ECDSA	ECDSA	Elliptic Curve [**selection**: *P-384*, *P-521*] using hash [**selection**: *SHA-384*, *SHA-512*]	[**selection**: *ISO/IEC 14888-3:2018 (Subclause 6.6)*, *FIPS PUB 186-5 (Section 6.4.2)*] [ECDSA] NIST SP-800 186 (Section 4) [NIST Curves]
LMS	LMS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Winternitz parameter = [**selection**: *1*, *2*, *4*, *8*] Tree height = [**selection**: *5*, *10*, *15*, *20*, *25*]	RFC 8554 [LMS] NIST SP 800-208 [parameters]
XMSS	XMSS	Private key size = [**selection**: • *192 bits with [**selection**: SHA-256/192, SHAKE256/192]* • *256 bits with [**selection**: SHA-256, SHAKE256]*] Tree height = [**selection**: *10*, *16*, *20*]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]
ML-DSA	ML-DSA Signature Verification	Parameter set = ML-DSA-87	NIST FIPS 204 (Section 5.3)

Application Note: 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 and selections to specify the parameters that are implemented for that algorithm.

Evaluation Activities

FCS_COP.1/SigVer

TSS

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

RSA-PKCS Signature Verification

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

To test the TOE's ability to perform RSA Digital Signature Verification using PKCS1-v1_5 signature type, the evaluator shall perform Generated Data Test using the following input parameters:

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

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

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

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

RSA-PSS Signature Verification

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

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

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

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

- *One test case is left unmodified*
- *For one test case the Message is modified*
- *For one test case the Signature is modified*

- *For one test case the exponent (e) is modified*
- *For one test case the IR is moved*
- *For one test case the Trailer is moved*

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

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

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

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

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

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

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

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

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

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

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

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

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

Identifier	***Cryptographic Algorithm Parameters***	***Cryptographic Key Sizes***	***List of Standards***
XMSS	*XMSS*	*Private key size = [**selection:** 192 bits with [**selection:** SHA256/192, SHAKE256/192], 256 bits with [**selection:** SHA-*	*RFC 8391 [XMSS]*

		*256, SHAKE256]], and tree height = [**selection:** 10, 16, 20]*	*NIST SP 800-208 [parameters]*

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

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

***XMSS Digital Signature Verification Test** For each supported combination of the above parameters, the evaluator shall generate four test cases consisting of signed messages and keys, such that*

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

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

ML-DSA Signature Verification

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

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

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

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

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

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

FCS_ENT_EXT.1 Entropy for Virtual Machines

FCS_ENT_EXT.1.1

The TSF shall provide a mechanism to make entropy available to VMs that is generated obtained via mechanisms specified in FCS_RBG.1.2 through [**selection**: *hypercall interface, virtual device interface, pass-through access to hardware entropy source*].

FCS_ENT_EXT.1.2

The TSF shall provide independent entropy across multiple VMs.

Application Note: This requirement ensures that sufficient entropy is available to any VM that requires it. The entropy need not provide high-quality entropy for every possible method that a VM might acquire it. The VMM must, however, provide some means for VMs to get sufficient entropy. For example, the VMM can provide an interface that returns entropy to a Guest VM. Alternatively, the VMM could provide pass-through access to entropy sources provided by the host platform.

This requirement allows for three general ways of providing entropy to guests: 1) The VS can provide a hypercall accessible to VM-aware guests, 2) access to a virtualized device that provides entropy, or 3) pass-through access to a hardware entropy source (including a source of random numbers). In all cases, it is possible that the guest is made VM-aware through installation of software or drivers. For the second and third cases, it is possible that the guest could be VM-unaware. There is no requirement that the TOE provide entropy sources as expected by VM-unaware guests. That is, the TOE does not have to anticipate every way a guest might try to acquire entropy as long as it supplies a mechanism that can be used by VM-aware guests, or provides access to a standard mechanism that a VM-unaware guest would use.

The ST author should select “hypercall interface” if the TSF provides an API function through which

guest-resident software can obtain entropy or random numbers. The ST author should select “virtual device interface” if the TSF presents a virtual device interface to the guest OS through which it can obtain entropy or random numbers. Such an interface could present a virtualized real device, such as a TPM, that can be accessed by VM-unaware guests, or a virtualized fictional device that would require the guest OS to be VM-aware. The ST author should select “pass-through access to hardware entropy source” if the TSF permits Guest VMs to have direct access to hardware entropy or random number source on the platform. The ST author should select all items that are appropriate.

For FCS_ENT_EXT.1.2, the VMM must ensure that the provision of entropy to one VM cannot affect the quality of entropy provided to another VM on the same platform.

Evaluation Activities

FCS_ENT_EXT.1
TSS
The evaluator shall verify that the TSS describes how the TOE provides entropy to Guest VMs, and how to access the interface to acquire entropy or random numbers. The evaluator shall verify that the TSS describes the mechanisms for ensuring that one VM does not affect the entropy acquired by another.

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

Tests
The evaluator shall perform the following tests:

- *Test FCS_ENT_EXT.1:1: The evaluator shall invoke entropy from each Guest VM. The evaluator shall verify that each VM acquires values from the interface.*
- *Test FCS_ENT_EXT.1:2: The evaluator shall invoke entropy from multiple VMs as nearly simultaneously as practicable. The evaluator shall verify that the entropy used in one VM is not identical to that invoked from the other VMs.*

FCS_RBG.1 Random Bit Generation (RBG)

FCS_RBG.1.1

The TSF shall perform deterministic random bit generation services using [**selection**: *DRBG Algorithm*] in accordance with [**selection**: *List of standards*] after initialization.

Table 9 provides the allowable choices for completion of the selection operations of FCS_RBG.1.

Table 9: Allowable choices for FCS_RBG.1.1

Identifier	DRBG Algorithm	List of standards
HASH_DRBG	Hash_DRBG with [**selection**: *SHA-384, SHA-512*]	[**selection**: *ISO/IEC 18031: 2011 (Section C.2.2), NIST SP 800-90A Revision 1 Section 10.1.1*]
HMAC_DRBG	HMAC_DRBG with [**selection**: *SHA-384, SHA-512*]	[**selection**: *ISO/IEC 18031: 2011 (Section C.2.3), NIST SP 800-90A Revision 1 Section 10.1.2*]
CTR_DRBG	CTR_DRBG with AES-CTR-256	[**selection**: *ISO/IEC 18031: 2011 (Section C.3.2), NIST SP800-90A Revision 1 Section 10.2.1*]

FCS_RBG.1.2

The TSF shall use a [**selection**: *TSF entropy source [**assignment**: name of entropy source], **multiple independent TSF entropy sources [assignment: names of entropy sources]**, TSF interface for obtaining entropy*] for initialization and reseeding.

Application Note: Following the approach in SP800-90B/C, noise sources are required to be independent. Non-independent noise sources that share dependencies are treated as a single raw source requiring a single entropy source validation. If a TOE has multiple dependent noise sources, they should be grouped according to their dependencies for the purposes of these requirements. For the selection in this requirement, the ST author selects "TSF entropy source..." if a single entropy source is used as input to the DRBG. The ST author selects "multiple independent TSF entropy sources..." if a seed is formed from a combination of two or more independent entropy sources within the TOE boundary. If the TSF implements two or more separate DRBGs that are seeded in separate manners,

this SFR should be iterated for each DRBG. If multiple distinct entropy sources exist such that each DRBG only uses one of them, then each iteration would select "TSF entropy source..."; "multiple independent TSF entropy sources..." is only selected if a single DRBG uses multiple independent entropy sources for its seed. The ST author selects "TSF interface for obtaining entropy" if entropy source data is generated outside the TOE boundary.

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

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

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

FCS_RBG.1.3

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

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

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

Evaluation Activities

FCS_RBG.1.1
Documentation will be produced and the evaluator shall perform the activities in accordance with and the Clarification to the Entropy Documentation and Assessment Annex.
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 DRBG implementation. If the DRBG 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 DRBG functionality. If the DRBG has prediction resistance enabled, each trial consists of:

1. *Instantiate the DRBG.*
2. *Generate a block of random bits.*
3. *Generate a second block of random bits.*
4. *Uninstantiate.*

The evaluator shall verify 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 DRBG does not have prediction resistance, each trial consists of:

1. *Instantiate the DRBG.*
2. *Generate a block of random bits.*
3. *Reseed.*
4. *Generate a second block of random bits.*
5. *Uninstantiate.*

The evaluator shall verify 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
There are no additional EAs for this element.

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.

5.1.4 Class FDP: User Data Protection

FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

FDP_HBI_EXT.1

The TSF shall use [**assignment**: *list of platform-provided, hardware-based mechanisms*] to constrain a Guest VM's direct access to the following physical devices: [**selection**: *all devices, all devices except for [**assignment**: list of devices to which Guest VMs may access]*].

Application Note: The TSF must use available hardware-based isolation mechanisms to constrain VMs when VMs have direct access to physical devices. “Direct access” in this context means that the VM can read or write device memory or access device I/O ports without the VMM being able to intercept and validate every transaction. The selection for "all devices" is made when VMs have no access to physical devices. Even if the VM has no inherent need to access a device, hardware-based isolation mechanisms must be enforced in case some malicious action, bug, or misconfiguration causes attempts to access that device.

Evaluation Activities

FDP_HBI_EXT.1
TSS
The evaluator shall ensure that the TSS provides evidence that hardware-based isolation mechanisms are used to constrain VMs when VMs have direct access to physical devices, including an explanation of the conditions under which the TSF invokes these protections.

Guidance
The evaluator shall verify that the operational guidance contains instructions on how to ensure that the platform-provided, hardware-based mechanisms are enabled.

Tests
There are no test activities for this component.

FDP_PPR_EXT.1 Physical Platform Resource Controls

FDP_PPR_EXT.1.1

The TSF shall allow an authorized administrator to control Guest VM access to the following physical platform resources: [**assignment**: *list of physical platform resources the VMM is able to control access to*].

FDP_PPR_EXT.1.2

The TSF shall explicitly deny all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources, [**assignment**: list of physical platform resources to which access is explicitly denied]*].

FDP_PPR_EXT.1.3

The TSF shall explicitly allow all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources, [**assignment**: list of physical platform resources to which access is always allowed]*].

Application Note: For purposes of this requirement, physical platform resources are divided into three categories:

- those to which guest OS access is configurable and moderated by the VMM
- those to which the guest OS is never allowed to have direct access, and
- those to which the guest OS is always allowed to have direct access.

For element 1, the ST author lists the physical platform resources that can be configured for Guest VM access by an administrator. For element 2, the ST author lists the physical platform resources to which Guest VMs may never be allowed direct access. If there are no such resources, the ST author selects "no physical platform resources." Likewise, any resources to which all Guest VMs automatically have access to are to be listed in the third element. If there are no such resources, then "no physical platform resources" is selected.

Evaluation Activities

FDP_PPR_EXT.1

TSS

The evaluator shall examine the TSS to determine that it describes the mechanism by which the VMM controls a Guest VM's access to physical platform resources. This description shall cover all of the physical platforms allowed in the evaluated configuration by the ST. It should explain how the VMM distinguishes among Guest VMs, and how each physical platform resource that is controllable (that is, listed in the assignment statement in the first element) is identified to an administrator.

The evaluator shall ensure that the TSS describes how the Guest VM is associated with each physical resource, and how other Guest VMs cannot access a physical resource without being granted explicit access. For TOEs that implement a robust interface (other than just "allow access" or "deny access"), the evaluator shall ensure that the TSS describes the possible operations or modes of access between a Guest VM's and physical platform resources.

If physical resources are listed in the second element, the evaluator shall examine the TSS and operational guidance to determine that there appears to be no way to configure those resources for access by a Guest VM. The evaluator shall document in the evaluation report their analysis of why the controls offered to configure access to physical resources can't be used to specify access to the resources identified in the second element (for example, if the interface offers a drop-down list of resources to assign, and the denied resources are not included on that list, that would be sufficient justification in the evaluation report).

Guidance

The evaluator shall examine the operational guidance to determine that it describes how an administrator is able to configure access to physical platform resources for Guest VMs for each platform allowed in the evaluated configuration according to the ST. The evaluator shall also determine that the operational guidance identifies those resources listed in the second and third elements of the component and notes that access to these resources is explicitly denied/allowed, respectively.

Tests

Using the operational guidance, the evaluator shall perform the following tests for each physical platform identified in the ST:

- *Test FDP_PPR_EXT.1:1: For each physical platform resource identified in the first element, the evaluator shall configure a Guest VM to have access to that resource and show that the Guest VM is able to successfully access that resource.*
- *Test FDP_PPR_EXT.1:2: For each physical platform resource identified in the first element, the evaluator shall configure the system such that a Guest VM does not have access to that resource and show that the Guest VM is unable to successfully access that resource.*
- *Test FDP_PPR_EXT.1:3: [conditional]: For TOEs that have a robust control interface, the evaluator shall exercise each element of the interface as described in the TSS and the operational guidance to ensure that the behavior described in the operational guidance is exhibited.*
- *Test FDP_PPR_EXT.1:4: [conditional]: If the TOE explicitly denies access to certain physical resources, the evaluator shall attempt to access each listed (in FDP_PPR_EXT.1.2) physical resource from a Guest VM and observe that access is denied.*
- *Test FDP_PPR_EXT.1:5: [conditional]: If the TOE explicitly allows access to certain physical resources, the evaluator shall attempt to access each listed (in FDP_PPR_EXT.1.3) physical resource from a Guest VM and observe that the access is allowed. If the operational guidance specifies that access is allowed simultaneously by*

more than one Guest VM, the evaluator shall attempt to access each resource listed from more than one Guest VM and show that access is allowed.

FDP_RIP_EXT.1 Residual Information in Memory

FDP_RIP_EXT.1.1

The TSF shall ensure that any previous information content of physical memory is cleared prior to allocation to a Guest VM.

Application Note: Physical memory must be zeroed before it is made accessible to a VM for general use by a guest OS.

The purpose of this requirement is to ensure that a VM does not receive memory containing data previously used by another VM or the host.

"For general use" means for use by the guest OS in its page tables for running applications or system software.

This does not apply to pages shared by design or policy between VMs or between the VMMs and VMs, such as read-only OS pages or pages used for virtual device buffers.

Evaluation Activities

FDP_RIP_EXT.1
TSS
The evaluator shall ensure that the TSS documents the process used for clearing physical memory prior to allocation to a Guest VM, providing details on when and how this is performed. Additionally, the evaluator shall ensure that the TSS documents the conditions under which physical memory is not cleared prior to allocation to a Guest VM, and describes when and how the memory is cleared.

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

Tests
There are no test activities for this component.

FDP_RIP_EXT.2 Residual Information on Disk

FDP_RIP_EXT.2.1

The TSF shall ensure that any previous information content of physical disk storage is cleared to zeroes upon allocation to a Guest VM.

Application Note: The purpose of this requirement is to ensure that a VM does not receive disk storage containing data previously used by another VM or by the host.

Clearing of disk storage only upon deallocation does not meet this requirement.

This does not apply to disk-resident files shared by design or policy between VMs or between the VMMs and VMs, such as read-only data files or files used for inter-VM data transfers permitted by policy.

Evaluation Activities

FDP_RIP_EXT.2
TSS
The evaluator shall ensure that the TSS documents how the TSF ensures that disk storage is zeroed upon allocation to Guest VMs. Also, the TSS must document any conditions under which disk storage is not cleared prior to allocation to a Guest VM. Any file system format and metadata information needed by the evaluator to perform the below test shall be made available to the evaluator, but need not be published in the TSS.

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

Tests

The evaluator shall perform the following test: On the host, the evaluator creates a file that is more than half the size of a connected physical storage device (or multiple files whose individual sizes add up to more than half the size of the storage media). This file (or files) shall be filled entirely with a non-zero value. Then, the file (or files) shall be released (freed for use but not cleared). Next, the evaluator (as a VS administrator) creates a virtual disk at least that large on the same physical storage device and connects it to a powered-off VM. Then, from outside the Guest VM, scan through and check that all the non-metadata (as documented in the TSS) in the file corresponding to that virtual disk is set to zero.

FDP_VMS_EXT.1 VM Separation

FDP_VMS_EXT.1.1

The VS shall provide the following mechanisms for transferring data between Guest VMs: [**selection**:

- *no mechanism*
- *virtual networking*
- *[**assignment**: other inter-VM data sharing mechanisms]*

].

FDP_VMS_EXT.1.2

The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs.

FDP_VMS_EXT.1.3

The TSF shall allow administrators to configure the mechanisms selected in FDP_VMS_EXT.1.1 to enable and disable the transfer of data between Guest VMs.

FDP_VMS_EXT.1.4

The VS shall ensure that no Guest VM is able to read or transfer data to or from another Guest VM except through the mechanisms listed in FDP_VMS_EXT.1.1.

Application Note: The fundamental requirement of a virtualization system is the ability to enforce separation between information domains implemented as Virtual Machines and Virtual Networks. The intent of this requirement is to ensure that VMs, VMMs, and the VS as a whole is implemented with this fundamental requirement in mind.

The ST author should select “no mechanism” in the unlikely event that the VS implements no mechanisms for transferring data between Guest VMs. Otherwise, the ST author should select “virtual networking” and identify all other mechanisms through which data can be transferred between Guest VMs.

Examples of non-network inter-VM sharing mechanisms are:

- User interface-based mechanisms, such as copy-paste and drag-and-drop
- Shared virtual or physical devices
- API-based mechanisms such as hypercalls

For data transfer mechanisms implemented in terms of hypercall functions, FDP_VMS_EXT.1.3 is met if FPT_HCL_EXT.1.1 is met for those hypercall functions (hypercall function parameters are checked).

For data transfer mechanisms that use shared physical devices, FDP_VMS_EXT.1.3 is met if the device is listed in and meets FDP_PPR_EXT.1.1 (VM access to the physical device is configurable).

For data transfer mechanisms that use virtual networking, FDP_VMS_EXT.1.3 is met if FDP_VNC_EXT.1.1 is met (VM access to virtual networks is configurable).

Evaluation Activities

FDP_VMS_EXT.1
TSS
The evaluator shall examine the TSS to verify that it documents all inter-VM communications mechanisms (as defined above), and explains how the TSF prevents the transfer of data between VMs outside of the mechanisms listed in FDP_VMS_EXT.1.1.

Guidance
The evaluator shall examine the operational guidance to ensure that it documents how to configure all inter-VM communications mechanisms, including how they are invoked and how they are disabled.

Tests
The evaluator shall perform the following tests for each documented inter-VM communications channel:

1. *Create two VMs without specifying any communications mechanism or overriding the default configuration.*
2. *Test that the two VMs cannot communicate through the mechanisms selected in FDP_VMS_EXT.1.1.*
3. *Create two new VMs, overriding the default configuration to allow communications through a channel selected in FDP_VMS_EXT.1.1.*
4. *Test that communications can be passed between the VMs through the channel.*
5. *Create two new VMs, the first with the inter-VM communications channel currently being tested enabled, and the second with the inter-VM communications channel currently being tested disabled.*
6. *Test that communications cannot be passed between the VMs through the channel.*
7. *As an administrator, enable inter-VM communications between the VMs on the second VM.*
8. *Test that communications can be passed through the inter-VM channel.*
9. *As an administrator again, disable inter-VM communications between the two VMs.*
10. *Test that communications can no longer be passed through the channel.*

FDP_VMS_EXT.1.2 is met if communication is unsuccessful in step 2. FDP_VMS_EXT.1.3 is met if communication is successful in step 4 and unsuccessful in step 6.

FDP_VNC_EXT.1 Virtual Networking Components

FDP_VNC_EXT.1.1

The TSF shall allow administrators to configure virtual networking components to connect VMs to each other and to physical networks.

FDP_VNC_EXT.1.2

The TSF shall ensure that network traffic visible to a Guest VM on a virtual network–or virtual segment of a physical network–is visible only to Guest VMs configured to be on that virtual network or segment.

Application Note: Virtual networks must be separated from one another to provide isolation commensurate with that provided by physically separate networks. It must not be possible for data to cross between properly configured virtual networks regardless of whether the traffic originated from a local Guest VM or a remote host.

Unprivileged users must not be able to connect VMs to each other or to external networks.

Evaluation Activities

FDP_VNC_EXT.1
TSS
The evaluator shall examine the TSS (or a proprietary annex) to verify that it describes the mechanism by which virtual network traffic is ensured to be visible only to Guest VMs configured to be on that virtual network.

Guidance
The evaluator shall ensure that the Operational Guidance describes how to create virtualized networks and connect VMs to each other and to physical networks.

Tests

- *Test FDP_VNC_EXT.1:1: The evaluator shall assume the role of the administrator and attempt to configure a VM to connect to a network component. The evaluator shall verify that the attempt is successful. The evaluator shall then assume the role of an unprivileged user and attempt the same connection. If the attempt fails, or there is no way for an unprivileged user to configure VM network connections, the requirement is met.*
- *Test FDP_VNC_EXT.1:2: The evaluator shall assume the role of the administrator and attempt to configure a VM to connect to a physical network. The evaluator shall verify that the attempt is successful. The evaluator shall then assume the role of an unprivileged user and make the same attempt. If the attempt fails, or there is no way for an unprivileged user to configure VM network connections, the requirement is met.*

5.1.5 Class FIA: Identification and Authentication

FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UAU.5.1

The TSF shall provide [**selection**:

- *[**selection**: local, directory-based] authentication based on username and password*

- *authentication based on username and a PIN that releases an asymmetric key stored in OE-protected storage*
- *[**selection**: local, directory-based] authentication based on X.509 certificates*
- *[**selection**: local, directory-based] authentication based on an SSH public key credential*

] to support user authentication.

Application Note: "Local" authentication refers to the authentication factors residing within the TSF, not whether the authentication mechanism itself is local to the TOE or remote. Selection of 'authentication based on username and password' requires that FIA_PMG_EXT.1 be included in the ST. This also requires that the ST claim management function 2 for password management. Selection of either 'authentication based on username and password' or 'authentication based on username and a PIN...' requires that FIA_AFL_EXT.1 be included in the ST. If the ST author selects 'authentication based on an SSH public-key credential,' the TOE is expected to conform to Functional Package for Secure Shell (SSH), version 2.0, If the ST author selects 'authentication based on X.509 certificates,' the TOE is expected to conform to Functional Package for X.509, version 1.0.

PINs used to access OE-protected storage are set and managed by the OE-protected storage mechanism. Thus requirements on PIN management are outside the scope of the TOE.

FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the [*following rules: [**selection**:*

- *If authentication is based on local password, the TSF shall perform Password-Based Key Derivation Function in accordance with a specified cryptographic algorithm [**selection**: HMAC-SHA-384, HMAC-SHA-512], with [**selection**: [**assignment**: positive integer of 10,000 or more] iterations, value supported by the platform greater than 1,000] and output cryptographic key sizes of 256 bits that meet the following: NIST SP 800-132,*
- *If directory-based authentication is used, the TSF shall abide by any decision returned by the directory with regards to the validity of the authentication request,*
- *[**assignment**: other rules describing how the multiple authentication mechanisms provide authentication],*
- *no other rules*

]].

Application Note: The intent behind FIA_UAU.5.2 is to specify how the TSF determines that authentication credentials are valid. In the case of password-based credentials, it is expected that the password is used to derive a key that is used to for validation, rather than comparing a cleartext or hashed password to a corresponding stored value.

In the case of directory-based authentication, the directory is expected to determine if the authentication request is valid and return a response to the TOE that the TSF will enforce. Specifically, a directory-based authentication attempt could be rejected not just because of invalid credentials but also other factors such as exceeding a failed authentication threshold (when passwords are used), suspended account, or time of day restrictions.

Evaluation Activities

FIA_UAU.5
TSS
There are no additional TSS evaluation activities for this component.

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

Tests
If 'username and password authentication' is selected, The evaluator shall configure the VS with a known username and password and conduct the following tests:

- *Test FIA_UAU.5:1: The evaluator shall attempt to authenticate to the VS using the known username 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 VS using the known username but an incorrect password. The evaluator shall ensure that the authentication attempt is unsuccessful.*

If 'username and PIN that releases an asymmetric key' is selected, 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 VS can interface. The evaluator shall then conduct the following tests:

- *Test FIA_UAU.5:3: The evaluator shall attempt to authenticate to the VS using the known user name and PIN. The evaluator will ensure that the authentication attempt is successful.*

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

If 'X.509 certificate authentication' is selected, The evaluator shall generate an X.509v3 certificate for an administrator user with the Client Authentication Enhanced Key Usage field set. The evaluator will provision the VS for authentication with the X.509v3 certificate. The evaluator shall ensure that the certificates are validated by the VS as per FIA_X509_EXT.1.1 in the and then conduct the following tests:

- *Test FIA_UAU.5:5: The evaluator shall attempt to authenticate to the VS using the X.509v3 certificate. The evaluator will ensure that the authentication attempt is successful.*
- *Test FIA_UAU.5:6: The evaluator shall generate a second certificate identical to the first except for the public key and any values derived from the public key. The evaluator shall attempt to authenticate to the VS with this certificate. The evaluator shall ensure that the authentication attempt is unsuccessful.*

If 'SSH public-key credential authentication' is selected, the evaluator shall generate a public-private host key pair on the TOE using RSA or ECDSA, and a second public-private key pair on a remote client. The evaluator shall provision the VS with the client public key for authentication over SSH, and conduct the following tests:

- *Test FIA_UAU.5:7: The evaluator shall attempt to authenticate to the VS using a message signed by the client private key that corresponds to provisioned client public key. The evaluator shall ensure that the authentication attempt is successful.*
- *Test FIA_UAU.5:8: The evaluator shall generate a second client key pair and will attempt to authenticate to the VS with the private key over SSH without first provisioning the VS to support the new key pair. The evaluator shall ensure that the authentication attempt is unsuccessful.*

FIA_UIA_EXT.1 Administrator Identification and Authentication

FIA_UIA_EXT.1.1

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

Application Note: Users do not have to authenticate, only administrators need to authenticate.

Evaluation Activities

FIA_UIA_EXT.1
TSS
The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed or used, any protocol transactions that take place, and what constitutes a "successful logon."

Guidance
The evaluator shall examine the operational guidance to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-shared keys, tunnels, certificates) to logging in are described. For each supported login method, the evaluator shall ensure the operational guidance provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the operational guidance provides sufficient instruction on limiting the allowed services.

Tests
There are no test activities for this component.

5.1.6 Class FMT: Security Management

FMT_MOF_EXT.1 Management of Security Functions Behavior

FMT_MOF_EXT.1.1

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

Application Note: Selection of "remote" requires the selection-based requirement FTP_TRP.1 to be claimed in the ST.

FMT_MOF_EXT.1.2

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

Table 10: Management Functions

Status Markers:
M = Mandatory (TOE must provide that function to that role)
O = Optional/Selectable/Conditional (TOE may provide the function to that role)
X = Not Permitted (TOE must not provide that function to that role)

#	Management Function	Admin	User	Application Notes
1	Ability to update the virtualization system.	M	X	See FPT_TUD_EXT.1.
2	[**selection**: *Ability to configure administrator password policy as defined in FIA_PMG_EXT.1*, *Not applicable*]	O	X	Must be selected if FIA_PMG_EXT.1 is claimed.
3	Ability to create, configure and delete VMs	M	O	
4	Ability to set default initial VM configurations	M	O	Optional for Users in the Client use case. Forbidden for Users in the Server use case.
5	Ability to configure virtual networks including VM	M	O	See FDP_VNC_EXT.1.
6	Ability to configure and manage the audit system and audit data	M	X	
7	Ability to configure VM access to physical devices	M	O	See FDP_PPR_EXT.1
8	Ability to configure inter-VM data sharing	M	O	See FDP_VMS_EXT.1
9	Ability to configure removable media policy	M	O	See FPT_RDM_EXT.1 Optional for Users in the Client use case. Forbidden for Users in the Server use case.
10	Ability to configure the cryptographic functionality	O	O	Mandatory for Admin role for the Server use case. Forbidden for Users for the Server use case. Optional for all roles for the Client use case.
11	Ability to change default authorization factors	M	X	See FIA_PMG_EXT.1
12	Ability to enable/disable screen lock	O	O	
13	Ability to configure screen lock inactivity timeout	O	O	
14	Ability to configure remote connection inactivity timeout	M	X	
15	Ability to configure lockout policy for unsuccessful authentication attempts through [**selection**: *timeouts between attempts*, *limiting number of attempts during a time period*]	O	X	Must be selected if "an administrator configurable..." is selected in FIA_AFL_EXT.1.1.
16	Ability to configure name/address of directory server to bind with	O	X	Must be selected if any selections with "directory-based" are made in FIA_UAU.5.1.

17	Ability to configure name/address of audit/logging server to which to send audit/logging records	M	X	See FAU_STG.1
18	Ability to configure name/address of network time server	M	O	
19	Ability to configure banner	M	X	See FTA_TAB.1
20	Ability to connect/disconnect removable devices to/from a VM	O	O	See FPT_RDM_EXT.1
21	Ability to start a VM	O	O	
22	Ability to stop/halt a VM	O	O	
23	Ability to checkpoint a VM	O	O	
24	Ability to suspend a VM	O	O	
25	Ability to resume a VM	O	O	
26	[**selection**: *Ability to configure action taken if unable to determine the validity of a certificate*, *Not applicable*]	O	X	This function must be selected if "allow the administrator to choose whether to accept the certificate in these cases" in FIA_X509_EXT.2.2

Application Note: The ST author is expected to update Table 10 with an indication as to whether any of the optional functions are included as part of the TOE. The ST author may also omit the 'Notes' column as it is provided in this PP as an aid to the ST author in constructing the table.

This SFR addresses the roles of the CC Part 2 SFRs FMT_MOF.1, FMT_SMF.1, and FMT_SMR.2.

Administration is considered "local" if the administrator is physically present at the machine on which the VS is installed.

Administration is considered "remote" if communications between the administrator and the Management Subsystem travel on a network.

There is no requirement to authenticate users of the virtualization system. Users that have access to VMs but not to the Management Subsystem need not authenticate to the virtualization system in order to use Guest VMs. Requirements for authentication of VM users is determined by the policies of the domains running within the Guest VMs.

For a VS where the OS is part of the platform and not part of the TOE, it is acceptable for the VS to invoke the Host OS screen lock.

Evaluation Activities

FMT_MOF_EXT.1

TSS

The evaluator shall examine the TSS and Operational Guidance to ensure that it describes which security management functions require administrator privilege and the actions associated with each management function. The evaluator shall verify that for each management function and role specified in Table 10, the defined role is able to perform all mandatory functions as well as all optional or selection-based functions claimed in the ST.

Guidance

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

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

Tests

The evaluator shall test each management function for each role listed in the FMT_MOF_EXT.1.1 Table 10 in the ST to demonstrate that the function can be performed by the roles that are authorized to do so and the result of the function is demonstrated. The evaluator shall also verify for each claimed management function in Table 10, that if the TOE claims

not to provide a particular role with access to the function, then it is not possible to access the TOE as that role and perform that function.

FMT_SMO_EXT.1 Separation of Management and Operational Networks

FMT_SMO_EXT.1.1

The TSF shall support the separation of management and operational network traffic through [**selection**:

- *separate physical networks*
- *separate logical networks*
- *trusted channels as defined in FTP_ITC_EXT.1*
- *data encryption using an algorithm specified in [**selection**: FCS_COP.1/AEAD, FCS_COP.1/SKC]*

].

Application Note: Management communications must be separate from user workload communications. Administrative network traffic—including communications between physical hosts concerning load balancing, audit data, VM startup and shutdown—must be isolated from guest operational networks. For purposes of this requirement, management traffic also includes VMs transmitted over management networks whether for backup, live migration, or deployment.

"Separate physical networks" refers to using separate physical interfaces and cables to isolate management and operational networks from each other.

"Separate logical networks" refers to using logical networking constructs, such as separate IP spaces or virtual networks to isolate traffic across general-purpose networking ports. Management and operational networks are kept separate within the hosts using separate virtualized networking components.

If the ST author selects "trusted channels..." then the protocols used for network separation must be selected in FTP_ITC_EXT.1.

The ST author selects "data encryption using an algorithm..." if, for example, the TOE encrypts VMs as data blobs for backup, storage, deployment, or live migration, and does not send the data through a tunnel. If the ST author selects "data encryption using an algorithm..." then the algorithms and key sizes must be selected in the corresponding requirement. If those algorithms include any defined in FCS_COP.1/SKC, the ST must include FCS_COP.1/SKC.

Evaluation Activities

FMT_SMO_EXT.1
TSS
The evaluator shall examine the TSS to verify that it describes how management and operational traffic is separated.

Guidance
The evaluator shall examine the operational guidance to verify that it details how to configure the VS to keep Management and Operational traffic separate.

Tests
The evaluator shall configure the TOE as documented in the guidance. If separation is logical, then the evaluator shall capture packets on the management network. If plaintext Guest network traffic is detected, the requirement is not met. If separation uses trusted channels, then the evaluator shall capture packets on the network over which traffic is tunneled. If plaintext Guest network traffic is detected, the requirement is not met. If data encryption is used, then the evaluator shall capture packets on the network over which the data is sent while a VM or other large data structure is being transmitted. If plaintext VM contents are detected, the requirement is not met.

5.1.7 Class FPT: Protection of the TSF

FPT_DVD_EXT.1 Non-Existence of Disconnected Virtual Devices

FPT_DVD_EXT.1.1

The TSF shall prevent Guest VMs from accessing virtual device interfaces that are not present in the VM's current virtual hardware configuration.

Application Note: The virtualized hardware abstraction implemented by a particular VS might include the virtualized interfaces for many different devices. Sometimes these devices are not present in a particular instantiation of a VM. The interface for devices not present must not be accessible by the VM.

Such interfaces include memory buffers, PCI Bus interfaces, and processor I/O ports.

The purpose of this requirement is to reduce the attack surface of the VMM by blocking access to unused interfaces.

Evaluation Activities

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

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

Tests
The evaluator shall connect a device to a VM, then from within the guest scan the VM's devices to ensure that the connected device is present--using a device driver or other available means to scan the VM's I/O ports or PCI Bus interfaces. (The device's interface should be documented in the TSS under FPT_VDP_EXT.1.) The evaluator shall remove the device from the VM and run the scan again. This requirement is met if the device's interfaces are no longer present.

FPT_EEM_EXT.1 Execution Environment Mitigations

FPT_EEM_EXT.1.1

The TSF shall take advantage of execution environment-based vulnerability mitigation mechanisms supported by the Platform such as: [**selection**:

- *Address space randomization*
- *Memory execution protection (e.g., DEP)*
- *Stack buffer overflow protection*
- *Heap corruption detection*
- *[**assignment**: other mechanisms]*
- *No mechanisms*

]

Application Note: Processor manufacturers, compiler developers, and operating system vendors have developed execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems. Software can often take advantage of these mechanisms by using APIs provided by the operating system or by enabling the mechanism through compiler or linker options. This requirement does not mandate that these protections be enabled throughout the virtualization system—only that they be enabled where they have likely impact. For example, code that receives and processes user input should take advantage of these mechanisms.

For the selection, the ST author selects the supported mechanisms and uses the assignment to include mechanisms not listed in the selection, if any.

Evaluation Activities

FPT_EEM_EXT.1
TSS
The evaluator shall examine the TSS to ensure that it states, for each platform listed in the ST, the execution environment-based vulnerability mitigation mechanisms used by the TOE on that platform. The evaluator shall ensure that the lists correspond to what is specified in FPT_EEM_EXT.1.1.

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

Tests
There are no test activities for this component.

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_HAS_EXT.1 Hardware Assists

FPT_HAS_EXT.1.1

The VMM shall use [**assignment**: *list of hardware-based virtualization assists*] to reduce or eliminate the need for binary translation.

FPT_HAS_EXT.1.2

The VMM shall use [**assignment**: *list of hardware-based virtualization memory-handling assists*] to reduce or eliminate the need for shadow page tables.

Application Note: These hardware-assists help reduce the size and complexity of the VMM, and thus, of the trusted computing base, by eliminating or reducing the need for paravirtualization or binary translation. Paravirtualization involves modifying guest software so that instructions that cannot be properly virtualized are never executed on the physical processor.

For the assignment in FPT_HAS_EXT.1, the ST author lists the hardware-based virtualization assists on all platforms included in the ST that are used by the VMM to reduce or eliminate the need for software-based binary translation. Examples for the x86 platform are Intel VT-x and AMD-V. “None” is an acceptable assignment for platforms that do not require virtualization assists in order to eliminate the need for binary translation. This must be documented in the TSS.

For the assignment in FPT_HAS_EXT.1.2, the ST author lists the set of hardware-based virtualization memory-handling extensions for all platforms listed in the ST that are used by the VMM to reduce or eliminate the need for shadow page tables. Examples for the x86 platform are Intel EPT and AMD RVI. “None” is an acceptable assignment for platforms that do not require memory-handling assists in order to eliminate the need for shadow page tables. This must be documented in the TSS.

Evaluation Activities

FPT_HAS_EXT.1
TSS
The evaluator shall examine the TSS to ensure that it states, for each platform listed in the ST, the hardware assists and memory-handling extensions used by the TOE on that platform. The evaluator shall ensure that these lists correspond to what is specified in the applicable FPT_HAS_EXT component.

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

Tests
There are no test activities for this component.

FPT_HCL_EXT.1 Hypercall Controls

FPT_HCL_EXT.1.1

The TSF shall validate the parameters passed to hypercall interfaces prior to execution of the VMM functionality exposed by each interface.

Application Note: The purpose of this requirement is to help ensure the integrity of the VMM by protecting the attack surface exposed to untrusted Guest VMs through hypercalls.

A hypercall interface allows VMM functionality to be invoked by VM-aware guest software. For example, a hypercall interface could be used to get information about the real world, such as the time of day or the underlying hardware of the host system. A hypercall could also be used to transfer data between VMs through a copy-paste mechanism. Because hypercall interfaces expose the VMM to Guest software, these interfaces constitute attack surface.

There is no expectation that The evaluator shall need to review source code in order to accomplish the evaluation activity.

Evaluation Activities

FPT_HCL_EXT.1
TSS
The evaluator shall examine the TSS (or proprietary TSS Annex) to ensure that all hypercall functions are documented at the level necessary for the evaluator to run the below test. Documentation for each hypercall interface must include: how to invoke the interface, parameters and legal values, and any conditions under which the interface can be invoked (e.g., from guest user mode, guest privileged mode, during guest boot only).

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

Tests
The evaluator shall perform the following test: For each hypercall interface documented in the TSS or proprietary TSS Annex, the evaluator shall attempt to invoke the function from within the VM using an invalid parameter (if any). If the VMM or VS crashes or generates an exception, or if no error is returned to the guest, then the test fails. If an error is returned to the guest, then the test succeeds.

FPT_RDM_EXT.1 Removable Devices and Media

FPT_RDM_EXT.1.1

The TSF shall implement controls for handling the transfer of virtual and physical removable media and virtual and physical removable media devices between information domains.

FPT_RDM_EXT.1.2

The TSF shall enforce the following rules when [**assignment**: *virtual or physical removable media and virtual or physical removable media devices*] are switched between information domains: [**selection**:

- *the administrator has granted explicit access for the media or device to be connected to the receiving domain*
- *the media in a device that is being transferred is ejected prior to the receiving domain being allowed access to the device*
- *the user of the receiving domain expressly authorizes the connection*
- *the device or media that is being transferred is prevented from being accessed by the receiving domain*

]

Application Note: The purpose of these requirements is to ensure that VMs are not given inadvertent access to information from different domains because of media or removable media devices left connected to physical machines. Removable media is media that can be ejected from a device, such as a compact disc, floppy disk, SD, or compact flash memory card.

Removable media devices are removable devices that include media, such as USB flash drives and USB hard drives. Removable media devices can themselves contain removable media (e.g., USB CDROM drives).

For purposes of this requirement, an Information Domain is:

a. A VM or collection of VMs
b. The virtualization system
c. Host OS
d. Management Subsystem

These requirements also apply to virtualized removable media—such as virtual CD drives that connect

to ISO images—as well as physical media—such as CDROMs and USB flash drives. In the case of virtual CDROMs, virtual ejection of the virtual media is sufficient.

In the first assignment, the ST author lists all removable media and removable media devices (both virtual and real) that are supported by the TOE. The ST author then selects actions that are appropriate for all removable media and removable media devices (both virtual and real) that are being claimed in the assignment.

For clarity, the ST author may iterate this requirement so that like actions are grouped with the removable media or devices to which they apply (e.g., the first iteration could contain all devices for which media is ejected on a switch; the second iteration could contain all devices for which access is prevented on a switch, etc.).

Evaluation Activities

FPT_RDM_EXT.1
TSS
The evaluator shall examine the TSS to ensure it describes the association between the media or devices supported by the TOE and the actions that can occur when switching information domains.

Guidance
The evaluator shall examine the operational guidance to ensure it documents how an administrator or user configures the behavior of each media or device.

Tests
The evaluator shall perform the following test for each listed media or device: The evaluator shall configure two VMs that are members of different information domains, with the media or device connected to one of the VMs. The evaluator shall disconnect the media or device from the VM and connect it to the other VM. The evaluator shall verify that the action performed is consistent with the action assigned in the TSS.

FPT_TST.1 TSF Self-Testing

FPT_TST.1.1

The TSF shall run a suite of the following self-tests [*during initial start-up, [**selection**: periodically during normal operation, at the request of the authorized user, at the conditions [**assignment**: conditions under which self-test should occur], at no other time]*] to demonstrate the correct operation of [*TSF DRBG specified in FCS_RBG.1*]: [**assignment**: *DRBG health tests*].

FPT_TST.1.2

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

FPT_TST.1.3

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

Application Note: This SFR is a required dependency of FCS_RBG.1. It is intended to require that any DRBG implemented by the TOE undergo health testing at startup (and optionally in other circumstances) 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. The TSF's response to a DRBG health test failure is addressed by in FPT_FLS.1.

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_TUD_EXT.1 Trusted Updates to the Virtualization System

FPT_TUD_EXT.1.1

The TSF shall provide administrators the ability to query the currently executed version of the TOE firmware and software as well as the most recently installed version of the TOE firmware and software.

Application Note: The version currently running (being executed) may not be the version most recently installed. For instance, maybe the update was installed but the system requires a reboot before this update will run. Therefore, it needs to be clear that the query should indicate both the most recently executed version as well as the most recently installed update.

FPT_TUD_EXT.1.2

The TSF shall provide administrators the ability to manually initiate updates to TOE firmware and software and [**selection**: *automatic updates*, *no other update mechanism*].

FPT_TUD_EXT.1.3

The TSF shall provide means to authenticate firmware and software updates to the TOE using a [**selection**: *digital signature mechanism using certificates*, *digital signature mechanism not using certificates*] prior to installing those updates.

Application Note: The digital signature mechanism referenced in FPT_TUD_EXT.1.3 is one of the algorithms specified in FCS_COP.1/SigVer.

If certificates are used by the update verification mechanism, then FIA_X509_EXT.1 and FIA_X509_EXT.2 in the Functional Package for X.509, version 1.0 must be included in the ST. Certificates are validated in accordance with FIA_X509_EXT.1 in the Functional Package for X.509, version 1.0 and the appropriate selections should be made in FIA_X509_EXT.2.1 in Functional Package for X.509, version 1.0. Additionally, FPT_TUD_EXT.2 must be included in the ST.

“Update” in the context of this SFR refers to the process of replacing a non-volatile, system resident software component with another. The former is referred to as the NV image, and the latter is the update image. While the update image is typically newer than the NV image, this is not a requirement. There are legitimate cases where the system owner may want to rollback a component to an older version (e.g., when the component manufacturer releases a faulty update, or when the system relies on an undocumented feature no longer present in the update). Likewise, the owner may want to update with the same version as the NV image to recover from faulty storage.

All discrete software components (e.g., applications, drivers, kernel, firmware) of the TSF, should be digitally signed by the corresponding manufacturer and subsequently verified by the mechanism performing the update. Since it is recognized that components may be signed by different manufacturers, it is essential that the update process verify that both the update and NV images were produced by the same manufacturer (e.g., by comparing public keys) or signed by legitimate signing keys (e.g., successful verification of certificates when using X.509 certificates).

The Digital Signature option is the preferred mechanism for authenticating updates. The Published Hash option will be removed from a future version of this PP.

Evaluation Activities

FPT_TUD_EXT.1
TSS
The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system software. Updates to the TOE either have a hash associated with them, or are signed by an authorized source. The evaluator shall verify that the description includes either a digital signature or published hash verification of the software before installation and that installation fails if the verification fails. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the update, and the actions that take place for both successful and unsuccessful verification. If digital signatures are used, the evaluator shall also ensure the definition of an authorized source is contained in the TSS.

If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or administrator guidance) describes how the certificates are installed/updated/selected, if necessary.

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

Tests
The evaluator shall perform the following tests:

- *Test FPT_TUD_EXT.1:1: The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that it is successfully installed on the TOE. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update.*
- *Test FPT_TUD_EXT.1:2: The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:*
    1. *A modified version (e.g., using a hex editor) of a legitimately signed or hashed update*
    2. *An image that has not been signed/hashed*
    3. *An image signed with an invalid hash or invalid signature (e.g., by using a different key as expected for creating the signature or by manual modification of a legitimate hash/signature)*

FPT_VDP_EXT.1 Virtual Device Parameters

FPT_VDP_EXT.1.1

The TSF shall provide interfaces for virtual devices implemented by the VMM as part of the virtual hardware abstraction.

FPT_VDP_EXT.1.2

The TSF shall validate the parameters passed to the virtual device interface prior to execution of the VMM functionality exposed by those interfaces.

Application Note: The purpose of this requirement is to ensure that the VMM is not vulnerable to compromise through the processing of malformed data passed to the virtual device interface from a guest OS. The VMM cannot assume that any data coming from a VM is well-formed—even if the virtual device interface is unique to the VS and the data comes from a virtual device driver supplied by the Virtualization Vendor.

Evaluation Activities

FPT_VDP_EXT.1
TSS
The evaluator shall examine the TSS to ensure it lists all virtual devices accessible by the guest OS. The TSS, or a separate proprietary document, must also document all virtual device interfaces at the level of I/O ports or PCI Bus interfaces - including port numbers (absolute or relative to a base), port name, address range, and a description of legal input values.

The TSS must also describe the expected behavior of the interface when presented with illegal input values. This behavior must be deterministic and indicative of parameter checking by the TSF.

The evaluator shall ensure that there are no obvious or publicly known virtual I/O ports missing from the TSS.

There is no expectation that evaluators will examine source code to verify the "all" part of the evaluation activity.

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

Tests
For each virtual device interface, the evaluator shall attempt to access the interface using at least one parameter value that is out of range or illegal. The test is passed if the interface behaves in the manner documented in the TSS. Interfaces that do not have input parameters need not be tested. This test can be performed in conjunction with the tests for FPT_DVD_EXT.1.

FPT_VIV_EXT.1 VMM Isolation from VMs

FPT_VIV_EXT.1.1

The TSF must ensure that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the platform.

FPT_VIV_EXT.1.2

The TSF must ensure that a Guest VM is unable to invoke platform code that runs at a privilege level equal to or exceeding that of the VMM without involvement of the VMM.

Application Note: This requirement is intended to ensure that software running within a Guest VM cannot compromise other VMs, the VMM, or the platform. This requirement is not met if Guest VM software—whatever its privilege level—can crash the VS or the Platform, or breakout of its virtual hardware abstraction to gain execution on the platform, within or outside of the context of the VMM.

This requirement is not violated if software running within a VM can crash the guest OS and there is no way for an attacker to gain execution in the VMM or outside of the virtualized domain.

FPT_VIV_EXT.1.2 addresses several specific mechanisms that must not be permitted to bypass the VMM and invoke privileged code on the Platform.

At a minimum, the TSF should enforce the following:

- On the x86 platform, a virtual System Management Interrupt (SMI) cannot invoke platform System Management Mode (SMM).
- An attempt to update virtual firmware or virtual BIOS cannot cause physical platform firmware or physical platform BIOS to be modified.
- An attempt to update virtual firmware or virtual BIOS cannot cause the VMM to be modified.

Of the above, the first bullet does not apply to platforms that do not support SMM. The rationale behind the third bullet is that a firmware update of a single VM must not affect other VMs. So if multiple VMs share the same firmware image as part of a common hardware abstraction, then the update of a single machine's BIOS must not be allowed to change the common abstraction. The virtual hardware abstraction is part of the VMM.

Evaluation Activities

FPT_VIV_EXT.1
TSS
The evaluator shall verify that the TSS (or a proprietary annex to the TSS) describes how the TSF ensures that guest software cannot degrade or disrupt the functioning of other VMs, the VMM or the platform. And how the TSF prevents guests from invoking higher-privilege platform code, such as the examples in the note.

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

Tests
There are no test activities for this component.

5.1.8 Class FTA: TOE Access

FTA_TAB.1 TOE Access Banner

FTA_TAB.1.1

Before establishing a user session, the [**selection**: *TSF*, *TOE platform*] shall display an [**assignment**: *advisory notice and consent warning regarding use of the TOE*] message.

Application Note: This requirement is intended to apply to interactive sessions between a human user and the TOE. IT entities establishing connections or programmatic connections (e.g., remote procedure calls over a network) are not required to be covered by this requirement.

Evaluation Activities

FTA_TAB.1
TSS
The evaluator shall examine the TSS to verify that it identifies the mechanism that is responsible for displaying the warning banner.

Guidance

The evaluator shall examine the operational guidance to verify that it describes how to configure the warning banner if the TSF provides the mechanism to do this.

Tests
The evaluator shall follow the operational guidance to configure the TOE or its operational environment (depending on selection made in FTA_TAB.1) to display the advisory warning message "TEST TEST Warning Message TEST TEST". The evaluator shall then log out and confirm that the configured advisory message is displayed before login can occur.

5.1.9 Class FTP: Trusted Path/Channels

FTP_ITC_EXT.1 Trusted Channel Communications

FTP_ITC_EXT.1.1

The TSF shall use [**selection**:

- *TLS as conforming to the Functional Package for Transport Layer Security (TLS), version 2.1*
- *TLS/HTTPS as conforming to FCS_HTTPS_EXT.1*
- *IPsec as conforming to FCS_IPSEC_EXT.1*
- *SSH as conforming to the Functional Package for Secure Shell (SSH), version 2.0*

] and [**selection**:

- *certificate-based authentication of the remote peer*
- *non-certificate-based authentication of the remote peer*
- *no authentication of the remote peer*

] to provide a trusted communication channel between itself and

- audit servers (as required by FAU_STG.1) and

[**selection**:

- *remote administrators (as required by FTP_TRP.1.1 (if selected in FMT_MOF_EXT.1)*
- *for separation of management and operational networks (if selected in FMT_SMO_EXT.1)*
- *[**assignment**: other remote IT entities]*
- *no other remote IT entities*

] that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from disclosure and detection of modification of the communicated data.

Application Note: If the ST author selects either TLS or HTTPS, the TOE will claim the relevant functionality defined in the Functional Package for Transport Layer Security (TLS), version 2.1. This PP does not mandate that a product implement TLS with mutual authentication, but if the product includes the capability to perform TLS with mutual authentication, then mutual authentication must be included within the TOE boundary.
If the ST author selects TLS/HTTPS, the TOE will claim FCS_HTTPS_EXT.1

If the ST author selects IPsec, the TOE will claim FCS_IPSEC_EXT.1.

If the ST author selects SSH, the TOE will claim the relevant functionality defined in the Functional Package for Secure Shell (SSH), version 2.0.

If the ST author selects "certificate-based authentication of the remote peer," then the TOE will claim the relevant functionality defined in the Functional Package for X.509, version 1.0. "No authentication of the remote peer" should be selected only if the TOE is acting as a server in a non-mutual authentication configuration.

Evaluation Activities

FTP_ITC_EXT.1
TSS
The evaluator shall review the TSS to determine that it lists all trusted channels the TOE uses for remote communications, including both the external entities and remote users used for the channel as well as the protocol that is used for each.

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

Tests
The evaluator shall configure the TOE to communicate with each external IT entity and type of remote user identified in the TSS. The evaluator shall monitor network traffic while the VS performs communication with each of these destinations. The

evaluator shall ensure that for each session a trusted channel was established in conformance with the protocols identified in the selection.

FTP_UIF_EXT.1 User Interface: I/O Focus

FTP_UIF_EXT.1.1

The TSF shall indicate to users which VM, if any, has the current input focus.

Application Note: This requirement applies to all users—whether user or administrator. In environments where multiple VMs run at the same time, the user must have a way of knowing which VM user input is directed to at any given moment. This is especially important in multiple-domain environments.

In the case of a human user, this is usually a visual indicator. In the case of headless VMs, the user is considered to be a program, but this program still needs to know which VM it is sending input to; this would typically be accomplished through programmatic means.

Evaluation Activities

FTP_UIF_EXT.1
TSS
The evaluator shall ensure that the TSS lists the supported user input devices.

Guidance
The evaluator shall ensure that the operational guidance specifies how the current input focus is indicated to the user.

Tests
For each supported input device, the evaluator shall demonstrate that the input from each device listed in the TSS is directed to the VM that is indicated to have the input focus.

FTP_UIF_EXT.2 User Interface: Identification of VM

FTP_UIF_EXT.2.1

The TSF shall support the unique identification of a VM's output display to users.

Application Note: In environments where a user has access to more than one VM at the same time, the user must be able to determine the identity of each VM displayed in order to avoid inadvertent cross-domain data entry.

There must be a mechanism for associating an identifier with a VM so that an application or program displaying the VM can identify the VM to users. This is generally indicated visually for human users (e.g., VM identity in the window title bar) and programmatically for headless VMs (e.g., an API function). The identification must be unique to the VS, but does not need to be universally unique.

Evaluation Activities

FTP_UIF_EXT.2
TSS
The evaluator shall ensure that the TSS describes the mechanism for identifying VMs to the user, how identities are assigned to VMs, and how conflicts are prevented.

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

Tests
The evaluator shall perform the following test: The evaluator shall attempt to create and start at least three Guest VMs on a single display device where the evaluator attempts to assign two of the VMs the same identifier. If the user interface displays different identifiers for each VM, then the requirement is met. Likewise, the requirement is met if the system refuses to create or start a VM when there is already a VM with the same identifier.

5.1.10 TOE Security Functional Requirements Rationale

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

Table 11: SFR Rationale

Threat	Addressed by	Rationale
T.3P_SOFTWARE	FAU_GEN.1	This threat is mitigated by this SFR because the audit events can report potential integrity breaches and attempts.
	FCS_CKM.1/AKG	This threat is mitigated by this SFR by generating strong cryptographic asymmetric keys to protect stored data.
	FCS_CKM.1/SKG	This threat is mitigated by this SFR by generating strong cryptographic symmetric keys to protect stored data.
	FCS_CKM.2 (Implementation-based)	This threat is mitigated by this SFR by using key encapsulation to distribute cryptographic keys.
	FCS_COP.1/AEAD	This threat is mitigated by this SFR by implementing authenticated encryption measures.
	FCS_COP.1/Hash	This threat is mitigated by this SFR by implementing cryptographic integrity verification measures.
	FCS_COP.1/KeyedHash	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/KeyEncap (Selection-based)	This threat is mitigated by this SFR by using a secure key encapsulation method to transmit a symmetric key to a third party as part of key establishment for trusted communications.
	FCS_COP.1/KeyWrap (Selection-based)	This threat is mitigated by this SFR by using a secure key wrap method to transmit a symmetric key to a third party as part of key establishment for trusted communications.
	FCS_COP.1/SigGen	This threat is mitigated by this SFR by ensuring that secure digital signature algorithms are used for trusted communications.
	FCS_COP.1/SigVer	This threat is mitigated by this SFR by implementing signature verification functions used for protected storage.
	FCS_COP.1/XOF (Selection-based)	This threat is mitigated by this SFR by supporting extendable output function implementations that are dependencies of some secure key generation and signature verification algorithms.
	FDP_PPR_EXT.1	This threat is mitigated by this SFR by requiring support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	This threat is mitigated by this SFR by ensuring that network traffic is visible only to VMs configured to be that network.
	FPT_DDI_EXT.1 (Optional)	This threat is mitigated by this SFR by requiring that physical device drivers be isolated other parts of the TOE and from one another.
	FPT_EEM_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_HCL_EXT.1	This threat is mitigated by this SFR by requiring that hypercall parameters be validated.
	FPT_ML_EXT.1 (Objective)	This threat is mitigated by this SFR by requiring measured launch of the platform and VMM (objective).

	FPT_VDP_EXT.1	This threat is mitigated by this SFR by requiring validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	Ensures that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
T.DATA_LEAKAGE	FAU_GEN.1	This threat is mitigated by this SFR because the audit events can report attempts to breach isolation.
	FCS_CKM.6	This threat is mitigated by this SFR by requiring cryptographic key destruction to protect domain data in shared storage.
	FCS_COP.1/AEAD	This threat is mitigated by this SFR by implementing authenticated encryption measures.
	FCS_COP.1/Hash	This threat is mitigated by this SFR by implementing cryptographic integrity verification measures.
	FCS_COP.1/KeyedHash	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigGen	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigVer	This threat is mitigated by this SFR by implementing signature verification functions used for protected storage.
	FCS_COP.1/XOF (Selection-based)	This threat is mitigated by this SFR by supporting extendable output function implementations that is a dependency of signature verification algorithms.
	FCS_ENT_EXT.1	This threat is mitigated by this SFR by requiring that domains have access to high-quality entropy for cryptographic purposes.
	FDP_PPR_EXT.1	This threat is mitigated by this SFR by requiring support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_RIP_EXT.1	This threat is mitigated by this SFR by ensuring that domain data is cleared from memory before memory is re-allocated.
	FDP_RIP_EXT.2	This threat is mitigated by this SFR by ensuring that domain data is cleared from physical storage upon re-allocation of the storage.
	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	This threat is mitigated by this SFR by ensuring that network traffic is visible only to VMs configured to be that network.
	FPT_DVD_EXT.1	This threat is mitigated by this SFR by ensuring that VMs can access only those virtual devices that they are configured to access.
	FPT_EEM_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use security mechanisms supported by the physical platform.
	FPT_GVI_EXT.1 (Optional)	This threat is mitigated by this SFR by requiring that the TOE support Guest VM measurements and integrity checks (optional).
	FPT_HAS_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_INT_EXT.1 (Objective)	This threat is mitigated by this SFR by requiring that the TOE support introspection into Guest VMs (optional).
	FPT_RDM_EXT.1	This threat is mitigated by this SFR by requiring support for rules for switching removable media between domains to reduce the chance of data spillage.

	FPT_VDP_EXT.1	This threat is mitigated by this SFR by requiring validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	This threat is mitigated by this SFR by ensuring that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
	FTP_UIF_EXT.1	This threat is mitigated by this SFR by ensuring that users are able to determine the domain with the current input focus.
	FTP_UIF_EXT.2	This threat is mitigated by this SFR by ensuring that users can know the identity of any VM that they can access.
T.DENIAL_OF_SERVICE	FCS_CKM.6	This threat is mitigated by this SFR by requiring cryptographic key destruction to ensure residual data in shared storage is unrecoverable.
	FDP_RIP_EXT.1	This threat is mitigated by this SFR by ensuring that domain data is cleared from memory before memory is re-allocated.
	FDP_RIP_EXT.2	This threat is mitigated by this SFR by ensuring that domain data is cleared from storage upon re-allocation of the storage.
T.MISCONFIGURATION	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that data sharing between VMs is turned off by default.
T.PLATFORM_ COMPROMISE	FDP_HBI_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported mechanisms for access to physical devices.
	FDP_PPR_EXT.1	This threat is mitigated by this SFR by requiring support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	This threat is mitigated by this SFR by ensuring that network traffic is visible only to VMs configured to be that network.
	FPT_DVD_EXT.1	This threat is mitigated by this SFR by ensuring that VMs cannot access virtual devices that they are not configured to access.
	FPT_EEM_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_HCL_EXT.1	This threat is mitigated by this SFR by requiring that hypercall parameters be validated.
	FPT_ML_EXT.1 (Objective)	This threat is mitigated by this SFR by requiring measured launch of the platform and VMM (objective).
	FPT_VDP_EXT.1	This threat is mitigated by this SFR by requiring validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	This threat is mitigated by this SFR by ensuring that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
T.UNAUTHORIZED_ ACCESS	FAU_GEN.1	This threat is mitigated by this SFR because the audit events can report potential attempts to access the management subsystem.
	FCS_CKM_EXT.7 (Implementation-based)	This threat is mitigated by this SFR by requiring use of strong cryptographic key establishment algorithms.
	FCS_CKM.1/AKG	This threat is mitigated by this SFR by requiring generation of strong asymmetric cryptographic keys.
	FCS_CKM.1/SKG	This threat is mitigated by this SFR by requiring generation of strong symmetric cryptographic keys.

FCS_COP.1/AEAD	This threat is mitigated by this SFR by implementing authenticated encryption measures.
FCS_COP.1/Hash	This threat is mitigated by this SFR by implementing cryptographic integrity verification measures.
FCS_COP.1/KeyedHash	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
FCS_COP.1/SigGen	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
FCS_COP.1/SigVer	This threat is mitigated by this SFR by implementing signature verification functions used for protected storage.
FCS_COP.1/SKC (Selection-based)	This threat is mitigated by this SFR by implementing cryptographic confidentiality measures.
FCS_COP.1/XOF (Selection-based)	This threat is mitigated by this SFR by supporting extendable output function implementations preventing access from outside entities.
FCS_HTTPS_EXT.1 (Selection-based)	This threat is mitigated by this SFR by ensuring that HTTPS trusted communications channels are implemented properly.
FCS_IPSEC_EXT.1 (Selection-based)	This threat is mitigated by this SFR by ensuring that IPsec trusted communications channels are implemented properly.
FCS_RBG.1	This threat is mitigated by this SFR by requiring that the TOE uses a cryptographically strong DRBG.
FCS_RBG.2 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FCS_RBG.3 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FCS_RBG.4 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FCS_RBG.5 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FIA_AFL_EXT.1 (Selection-based)	This threat is mitigated by this SFR by requiring that the TOE detect failed authentication attempts for administrator access.
FIA_PMG_EXT.1 (Selection-based)	This threat is mitigated by this SFR by ensuring that password-based administrator login is properly implemented.
FIA_PSK_EXT.1 (Selection-based)	This SFR mitigates the threat by optionally requiring support for pre-shared keys as an alternate authentication method for IPsec.
FIA_UAU.5	This threat is mitigated by this SFR by ensuring that strong mechanisms are used for administrator authentication.
FIA_UIA_EXT.1	This threat is mitigated by this SFR by requiring that administrators be successfully authenticated before performing management functions.
FIA_X509_EXT.4 (Selection-based)	This threat is mitigated by this SFR by requiring an exception to be met without the need to validate the revocation of the certificate.
FMT_MOF_EXT.1	This threat is mitigated by this SFR by limiting the management functions to be limited to authorized administrators as managed through controls required by FMT_MOF_EXT.1.
FMT_SMO_EXT.1	This threat is mitigated by this SFR by requiring that the TOE support having separate management and operational networks.

	FPT_FLS.1	This threat is mitigated by this SFR by ensuring that the TOE enters a failure state if it is unable to implement its DRBG function to adequately protect data in transit.
	FPT_TST.1	This threat is mitigated by this SFR by requiring self-testing for DRBG functionality to ensure that keys used for protection of data in transit are adequately strong.
	FTA_TAB.1	This threat is mitigated by this SFR by displays advisory notice and consent warning message regarding use of the TOE to administrators.
	FTP_ITC_EXT.1	This threat is mitigated by this SFR by ensuring that trusted communications channels are implemented using good cryptography.
	FTP_TRP.1 (Selection-based)	This threat is mitigated by this SFR by ensuring that certain communications use a trusted path.
T.UNAUTHORIZED_ MODIFICATION	FAU_ARP.1 (Optional)	This threat is mitigated by this SFR by requiring support for automatic responses to audit events (optional).
	FAU_GEN.1	This threat is mitigated by this SFR because the audit events can report potential integrity breaches and attempts.
	FAU_SAA.1 (Optional)	This threat is mitigated by this SFR by requiring support for rules for indicating security violations based on audit events (optional).
	FAU_SAR.1	This threat is mitigated by this SFR by requiring support for administrator review of audit records.
	FAU_STG.1	This threat is mitigated by this SFR by requiring support for protected transmission of audit records off the TOE.
	FAU_STG.2	This threat is mitigated by this SFR by requiring protection of stored audit records.
	FAU_STG.5	This threat is mitigated by this SFR by taking some action to ensure that exhaustion of audit storage does not allow for activities to be performed without auditing.
	FCS_CKM.1/AKG	This threat is mitigated by this SFR by requiring generation of asymmetric keys for protection of integrity measures.
	FCS_CKM.1/SKG	This threat is mitigated by this SFR by requiring generation of symmetric keys for protection of integrity measures.
	FCS_COP.1/AEAD	This threat is mitigated by this SFR by implementing authenticated encryption measures.
	FCS_COP.1/Hash	This threat is mitigated by this SFR by implementing cryptographic integrity verification measures.
	FCS_COP.1/KeyedHash	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigGen	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigVer	This threat is mitigated by this SFR by implementing signature verification functions used for protected storage.
	FCS_COP.1/XOF (Selection-based)	This threat is mitigated by this SFR by supporting extendable output function implementations to prevent the execution of modified code.
	FDP_PPR_EXT.1	This threat is mitigated by this SFR by requiring support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that authorized data transfers between VMs are done securely.

	FDP_VNC_EXT.1	This threat is mitigated by this SFR by ensuring that network traffic is visible only to VMs configured to be that network.
	FPT_DDI_EXT.1 (Optional)	This threat is mitigated by this SFR by requiring that physical device drivers be isolated other parts of the TOE and from one another.
	FPT_EEM_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_HCL_EXT.1	This threat is mitigated by this SFR by requiring that hypercall parameters be validated.
	FPT_ML_EXT.1 (Objective)	This threat is mitigated by this SFR by requiring measured launch of the platform and VMM (objective).
	FPT_VDP_EXT.1	This threat is mitigated by this SFR by requiring validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	This threat is mitigated by this SFR by ensuring that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
T.UNAUTHORIZED_UPDATE	FAU_GEN.1	This threat is mitigated by this SFR because the audit events can report potential integrity breaches and attempts.
	FCS_CKM.1/AKG	This threat is mitigated by this SFR by requiring generation of asymmetric keys for protection of integrity measures.
	FCS_CKM.1/SKG	This threat is mitigated by this SFR by requiring generation of symmetric keys for protection of integrity measures.
	FCS_COP.1/AEAD	This threat is mitigated by this SFR by implementing authenticated encryption measures.
	FCS_COP.1/Hash	This threat is mitigated by this SFR by implementing cryptographic integrity verification measures.
	FCS_COP.1/KeyedHash	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigGen	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigVer	This threat is mitigated by this SFR by implementing signature verification functions used for protected storage.
	FCS_COP.1/XOF (Selection-based)	This threat is mitigated by this SFR by supporting extendable output function implementations to prevent modification of the virtualization system's behavior.
	FDP_PPR_EXT.1	This threat is mitigated by this SFR by requiring support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	This threat is mitigated by this SFR by ensuring that network traffic is visible only to VMs configured to be that network.
	FMT_MOF_EXT.1	This SFR mitigates this threat by ensuring that the only way to modify the VS is through a trusted update process initiated by an authorized administrator as required by FMT_MOF_EXT.1.
	FPT_DDI_EXT.1 (Optional)	This threat is mitigated by this SFR by requiring that physical device drivers be isolated other parts of the TOE and from one another.

	FPT_EEM_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_HCL_EXT.1	This threat is mitigated by this SFR by requiring that hypercall parameters be validated.
	FPT_ML_EXT.1 (Objective)	This threat is mitigated by this SFR by requiring measured launch of the platform and VMM (objective).
	FPT_VDP_EXT.1	This threat is mitigated by this SFR by requiring validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	This threat is mitigated by this SFR by ensuring that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
T.UNPATCHED_ SOFTWARE	FPT_IDV_EXT.1 (Objective)	This threat is mitigated by this SFR by requiring support for software identification labels (optional).
	FPT_TUD_EXT.1	This threat is mitigated by this SFR by requiring support for product updates.
	FPT_TUD_EXT.2 (Selection-based)	This threat is mitigated by this SFR by requiring specific requirements for certificate-based code signing for update.
T.USER_ERROR	FAU_GEN.1	This threat is mitigated by this SFR because the audit events can report attempts to breach isolation.
	FCS_CKM.6	This threat is mitigated by this SFR by requiring cryptographic key destruction to protect domain data in shared storage.
	FDP_PPR_EXT.1	This threat is mitigated by this SFR by requiring support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_RIP_EXT.1	This threat is mitigated by this SFR by ensuring that domain data is cleared from memory before memory is re-allocated.
	FDP_RIP_EXT.2	This threat is mitigated by this SFR by ensuring that domain data is cleared from physical storage upon re-allocation of the storage.
	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	This threat is mitigated by this SFR by ensuring that network traffic is visible only to VMs configured to be that network.
	FPT_DVD_EXT.1	This threat is mitigated by this SFR by ensuring that VMs can access only those virtual devices that they are configured to access.
	FPT_EEM_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_VDP_EXT.1	This threat is mitigated by this SFR by requiring validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	This threat is mitigated by this SFR by ensuring that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
T.VMM_COMPROMISE	FAU_GEN.1	This threat is mitigated by this SFR because the audit events can report potential integrity breaches and attempts.
	FCS_CKM.6	This threat is mitigated by this SFR by requiring cryptographic key destruction to protect domain data in shared storage.

	FCS_COP.1/AEAD	This threat is mitigated by this SFR by implementing authenticated encryption measures.
	FCS_COP.1/Hash	This threat is mitigated by this SFR by implementing cryptographic integrity verification measures.
	FCS_COP.1/KeyedHash	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigGen	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
	FCS_COP.1/SigVer	This threat is mitigated by this SFR by implementing signature verification functions used for protected storage.
	FDP_PPR_EXT.1	This threat is mitigated by this SFR by requiring support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_RIP_EXT.1	This threat is mitigated by this SFR by ensuring that domain data is cleared from memory before memory is re-allocated.
	FDP_RIP_EXT.2	This threat is mitigated by this SFR by ensuring that domain data is cleared from physical storage upon re-allocation of the storage.
	FDP_VMS_EXT.1	This threat is mitigated by this SFR by ensuring that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	This threat is mitigated by this SFR by ensuring that network traffic is visible only to VMs configured to be that network.
	FPT_DDI_EXT.1 (Optional)	This threat is mitigated by this SFR by requiring that physical device drivers be isolated other parts of the TOE and from one another.
	FPT_DVD_EXT.1	This threat is mitigated by this SFR by ensuring that VMs can access only those virtual devices that they are configured to access.
	FPT_EEM_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	This threat is mitigated by this SFR by requiring that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_HCL_EXT.1	This threat is mitigated by this SFR by requiring that hypercall parameters be validated.
	FPT_ML_EXT.1 (Objective)	This threat is mitigated by this SFR by requiring measured launch of the platform and VMM (objective).
	FPT_VDP_EXT.1	This threat is mitigated by this SFR by requiring validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	This threat is mitigated by this SFR by ensuring that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
T.WEAK_CRYPTO	FCS_CKM_EXT.7 (Implementation-based)	This threat is mitigated by this SFR by requiring use of strong cryptographic key establishment algorithms.
	FCS_CKM.1/AKG	This threat is mitigated by this SFR by requiring generation of strong asymmetric cryptographic keys.
	FCS_CKM.1/SKG	This threat is mitigated by this SFR by requiring generation of symmetric keys for protection of integrity measures.
	FCS_COP.1/AEAD	This threat is mitigated by this SFR by implementing authenticated encryption measures.
	FCS_COP.1/Hash	This threat is mitigated by this SFR by implementing cryptographic integrity verification measures.

FCS_COP.1/KeyedHash	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
FCS_COP.1/SigGen	This threat is mitigated by this SFR by implementing cryptographic authenticity verification measures.
FCS_COP.1/SigVer	This threat is mitigated by this SFR by implementing signature verification functions used for protected storage.
FCS_COP.1/SKC (Selection-based)	This threat is mitigated by this SFR by implementing cryptographic confidentiality measures.
FCS_COP.1/XOF (Selection-based)	This threat is mitigated by this SFR by supporting extendable output function implementations to prevent data at rest or in transit to be disclosed.
FCS_ENT_EXT.1	This threat is mitigated by this SFR by requiring that domains have access to high-quality entropy for cryptographic purposes.
FCS_RBG.1	This threat is mitigated by this SFR by requiring that the TOE has access to high-quality entropy for cryptographic purposes.
FCS_RBG.2 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FCS_RBG.3 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FCS_RBG.4 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FCS_RBG.5 (Selection-based)	This threat is mitigated by this SFR by ensuring that entropy data used for DRBG seeding is sufficiently unpredictable to result in the generation of strong keys.
FPT_FLS.1	This threat is mitigated by this SFR by ensuring that the TOE enters a failure state if it is unable to implement its DRBG function to adequately protect data in transit.
FPT_TST.1	This threat is mitigated by this SFR by requiring self-testing for DRBG functionality to ensure that keys used for protection of data in transit are adequately strong.

5.2 Security Assurance Requirements

The Security Objectives for the TOE in Section 4 were constructed to address threats identified in Section 3.1. The Security Functional Requirements (SFRs) in Section 5.1 are a formal instantiation of the Security Objectives. The PP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of Security Assurance Requirements (SARs) from Part 3 of the Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5 that are required in evaluations against this PP. Individual evaluation activities to be performed are specified in both Section 5.1 as well as in this section.

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

The TOE Security Assurance Requirements are identified in Table 12.

Table 12: Security Assurance Requirements

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

5.2.1 Class ASE: Security Target Evaluation

As per ASE activities defined in [CEM] plus the TSS evaluation activities defined for any SFRs claimed by the TOE.

5.2.2 Class ADV: Development

The information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (TSS) portion of the ST. The TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The evaluation activities contained in Section 5.2 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

ADV_FSP.1 Basic functional specification

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.

Developer Note: As indicated in the introduction to this section, the functional specification is composed of the information contained in the AGD_OPR and AGD_PRE documentation, coupled with the information provided in the TSS of the ST. 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.

Application Note: There are no specific evaluation activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Section 5.2, 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 Documents

The guidance documents will be provided with the developer's security target. Guidance must include a description of how the authorized user verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by an authorized user.

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 TOE in that environment; and
- instructions to manage the security of the TOE as a product and as a component of the larger operational environment.

Guidance pertaining to particular security functionality is also provided; specific requirements on such guidance are contained in the evaluation activities specified with individual SFRs where applicable.

AGD_OPE.1 Operational User Guidance

Developer action elements:

AGD_OPE.1.1D

The developer shall provide operational user guidance.

Developer Note: Rather than repeat information here, the developer should review the evaluation activities for this component to ascertain the specifics of the guidance that the evaluators will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.1C

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

AGD_OPE.1.2C

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

AGD_OPE.1.3C

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

AGD_OPE.1.4C

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

AGD_OPE.1.5C

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

AGD_OPE.1.6C

The operational user guidance shall, for ~~each user role~~ **the authorized user**, 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 will be verified by the evaluation activities in Section 5.2 and evaluation of the TOE according to the CEM. The following additional information is also required.

The operational guidance shall contain instructions for configuring the password characteristics, number of allowed authentication attempt failures, the lockout period times for inactivity, and the notice and consent warning that is to be provided when authenticating.

The operational guidance shall contain step-by-step instructions suitable for use by an end-user of the VS to configure a new, out-of-the-box system into the configuration evaluated under this Protection Profile.

The documentation shall describe the process for verifying updates to the TOE, either by checking the hash or by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

- *Instructions for querying the current version of the TOE software.*
- *For hashes, a description of where the hash for a given update can be obtained. For digital signatures, instructions for obtaining the certificate that will be used by the FCS_COP.1/SigVer mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means.*
- *Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).*
- *Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.*

AGD_PRE.1 Preparative procedures

Developer action elements:

AGD_PRE.1.1D

The developer shall provide the TOE including its preparative procedures.

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

Content and presentation elements:

AGD_PRE.1.1C

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

AGD_PRE.1.2C

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

Evaluator action elements:

AGD_PRE.1.1E

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

AGD_PRE.1.2E

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

Evaluation Activities

AGD_PRE.1
As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms (that is, combination of hardware and operating system) claimed for the TOE in the ST.

The operational guidance shall contain step-by-step instructions suitable for use by an end-user of the VS to configure a new, out-of-the-box system into the configuration evaluated under this Protection Profile.

5.2.4 Class ALC: Life-Cycle Support

At the assurance level specified for TOEs conformant to this PP, life-cycle support is limited to an examination of the TOE vendor's development and configuration management process in order to provide a baseline level of assurance that the TOE itself is developed in a secure manner and that the developer has a well-defined process in place to deliver updates to mitigate known security flaws. This is a result of the critical role that a developer's practices play in contributing to the overall trustworthiness of a product.

ALC_CMC.1 Labeling of the TOE

Developer action elements:

ALC_CMC.1.1D
The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.1.1C
The TOE shall be labeled with its unique reference.

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.

The evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST.

If the vendor maintains a website advertising the TOE, the evaluator shall examine the information on the website to ensure that the information in the ST is sufficient to distinguish the product.

ALC_CMS.1 TOE CM coverage

Developer action elements:

ALC_CMS.1.1D
The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.1.1C
The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.2C
The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.1E

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

Evaluation Activities

ALC_CMS.1
The evaluator shall ensure that the developer has identified (in public-facing development guidance for their platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment 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 TOE developer, in conjunction with any other necessary parties, to provide information as to how the VS is updated to address security issues in a timely manner. The documentation describes the process of providing updates to the public from the time a security flaw is reported/discovered, to the time an update is released. This description includes the parties involved (e.g., the developer, hardware vendors) and the steps that are performed (e.g., developer testing), including worst case time periods, before an update is made available to the public.

Developer action elements:

ALC_TSU_EXT.1.1D

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

Content and presentation elements:

ALC_TSU_EXT.1.1C

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

ALC_TSU_EXT.1.2C

The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.

Application Note: The total length of time may be presented as a summation of the periods of time that each party (e.g., TOE developer, hardware vendor) on the critical path consumes. The time period until public availability per deployment mechanism may differ; each is described.

ALC_TSU_EXT.1.3C

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

Application Note: The reporting mechanism could include websites, email addresses, and 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 for TOE software. The evaluator will also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator 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 TOE patching this

vulnerability. 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 TOE. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

5.2.5 Class ATE: Tests

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

ATE_IND.1 Independent Testing - Conformance

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operation) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 5.1 are being met, although some additional testing is specified for SARs in Section 5.2. 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/TOE combinations that are claiming conformance to this PP.

Developer action elements:

ATE_IND.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements:

ATE_IND.1.1C

The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.1.1E

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

ATE_IND.1.2E

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

Evaluation Activities

ATE_IND.1
The evaluator shall prepare a test plan and report documenting the testing aspects of the system. While it is not necessary to have one test case per test listed in an evaluation activity, the evaluators 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 evaluators are 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) is provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of cryptographic engines to be used. The cryptographic algorithms implemented by these engines are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS/HTTPS, SSH).

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

5.2.6 Class AVA: Vulnerability Assessment

For the first generation of this Protection Profile, the evaluation lab is expected to survey open sources to learn 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, evaluators will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future PPs.

AVA_VAN.1 Vulnerability survey

Developer action elements:

AVA_VAN.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN.1.1C

The TOE shall be suitable for testing.

Evaluator action elements:

AVA_VAN.1.1E

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

AVA_VAN.1.2E

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

AVA_VAN.1.3E

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

Evaluation Activities

AVA_VAN.1

As with ATE_IND 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 determine the vulnerabilities that have been found in virtualization in general, as well as those that pertain to the particular TOE. 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. For example, if the vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this PP. 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 13: Auditable Events for Strictly Optional Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ARP.1	Actions taken due to potential security violations.	No additional information
FAU_SAA.1	Enabling and disabling of any of the analysis mechanisms.	No additional information
	Automated responses performed by the TSF.	No additional information
FPT_DDI_EXT.1	No events specified	N/A
FPT_GVI_EXT.1	Actions taken due to failed integrity check.	No additional information

A.1.2 Class ALC: Life-Cycle Support

ALC_FLR.1 Basic Flaw Remediation (ALC_FLR.1)

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

Developer action elements:

ALC_FLR.1.1D

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

Content and presentation elements:

ALC_FLR.1.1C

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

ALC_FLR.1.2C

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

ALC_FLR.1.3C

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

ALC_FLR.1.4C

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

Evaluator action elements:

ALC_FLR.1.1E

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

Evaluation Activities

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

ALC_FLR.2 Flaw Reporting Procedures (ALC_FLR.2)

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

Developer action elements:

ALC_FLR.2.1D

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

ALC_FLR.2.2D

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

ALC_FLR.2.3D

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

Content and presentation elements:

ALC_FLR.2.1C

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

ALC_FLR.2.2C

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

ALC_FLR.2.3C

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

ALC_FLR.2.4C

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

ALC_FLR.2.5C

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

ALC_FLR.2.6C

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

ALC_FLR.2.7C

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

ALC_FLR.2.8C

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

Evaluator action elements:

ALC_FLR.2.1E

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

Evaluation Activities

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

ALC_FLR.3 Systematic Flaw Remediation (ALC_FLR.3)

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

Developer action elements:

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

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

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

Content and presentation elements:

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

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

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

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

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

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

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

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

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

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

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

Evaluator action elements:

ALC_FLR.3.1E

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

Evaluation Activities

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

A.1.3 Class FAU: Security Audit

FAU_ARP.1 Security Audit Automatic Response

FAU_ARP.1.1

The TSF shall take [**assignment**: *list of actions*] upon detection of a potential security violation.

Application Note: In certain cases, it may be useful for virtualization systems to perform automated responses to certain security events. An example may include halting a VM which has taken some action to violate a key system security policy. This may be especially useful with headless endpoints when there is no human user in the loop.

The potential security violation mentioned in FAU_ARP.1.1 refers to FAU_SAA.1.

Evaluation Activities

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

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

Tests
The evaluator shall generate a potential security violation as defined in FAU_SAA.1 and verify that each action in the assignment in FAU_ARP.1.1 is performed by the TSF as a result. The evaluator shall perform this action for each security violation that is defined in FAU_SAA.1.

FAU_SAA.1 Potential Violation Analysis

FAU_SAA.1.1

The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the enforcement of the SFRs.

FAU_SAA.1.2

The TSF shall enforce the following rules for monitoring audited events:

a. Accumulation or combination of [**assignment**: *subset of defined auditable events*] known to indicate a potential security violation;
b. [**assignment**: *any other rules*].

Application Note: The potential security violation described in FAU_SAA.1 can be used as a trigger for automated responses as defined in FAU_ARP.1.

Evaluation Activities

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

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

Tests

The evaluator shall cause each combination of auditable events defined in FAU_SAA.1.2 to occur, and verify that a potential security violation is indicated by the TSF.

A.1.4 Class FPT: Protection of the TSF

FPT_DDI_EXT.1 Device Driver Isolation

FPT_DDI_EXT.1.1

The TSF shall ensure that device drivers for physical devices are isolated from the VMM and all other domains.

Application Note: In order to function on physical hardware, the VMM must have access to the device drivers for the physical platform on which it runs. These drivers are often written by third parties, and yet are effectively a part of the VMM. Thus the integrity of the VMM in part depends on the quality of third party code that the virtualization vendor has no control over. By encapsulating these drivers within one or more dedicated driver domains (e.g., Service VM or VMs) the damage of a driver failure or vulnerability can be contained within the domain, and would not compromise the VMM. When driver domains have exclusive access to a physical device, hardware isolation mechanisms, such as Intel's VT-d, AMD's Input/Output Memory Management Unit (IOMMU), or ARM's System Memory Management Unit (MMU) should be used to ensure that operations performed by Direct Memory Access (DMA) hardware are properly constrained.

Evaluation Activities

FPT_DDI_EXT.1
TSS
The evaluator shall examine the TSS documentation to verify that it describes the mechanism used for device driver isolation. If the TSS document indicates that a hardware isolation mechanism is used, the evaluator shall verify that the TSS documentation enumerates the hardware-isolated DMA-capable devices, and that it also provides a complete list of the accessible targets for memory transactions for each of those DMA-capable devices. (An example of information that might be included in the TSS documentation: a listing of all pages belonging to the driver domain, the identification of a subset of the driver domain's pages that the driver domain has permitted the device access to, or the identification of a dedicated area of memory reserved for the device or driver domain).

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

Tests
There are no test activities for this component.

FPT_GVI_EXT.1 Guest VM Integrity

FPT_GVI_EXT.1.1

The TSF shall verify the integrity of Guest VMs through the following mechanisms: [**assignment**: *list of Guest VM integrity mechanisms*].

Application Note: The primary purpose of this requirement is to identify and describe the mechanisms used to verify the integrity of Guest VMs that have been 'imported' in some fashion, though these mechanisms could also be applied to all Guest VMs, depending on the mechanism used. Importation for this requirement could include VM migration (live or otherwise), the importation of virtual disk files that were previously exported, VMs in shared storage, etc. It is possible that a trusted VM could have been modified during the migration or import/export process, or VMs could have been obtained from untrusted sources in the first place, so integrity checks on these VMs can be a prudent measure to take. These integrity checks could be as thorough as making sure the entire VM exactly matches a previously known VM (by hash for example), or by simply checking certain configuration settings to ensure that the VM's configuration will not violate the security model of the VS.

Evaluation Activities

FPT_GVI_EXT.1
TSS
For each mechanism listed in the assignment, the evaluator shall ensure that the TSS documents the mechanism, including how it verifies VM integrity, which set of Guest VMs it will check (all Guest VMs, only migrated VM s, etc.), when such

checks occur (before VM startup, immediately following importation/migration, on demand, etc.), and which actions are taken if a VM fails the integrity check (or which range of actions are possible if the action is configurable).

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

Tests
There are no test activities for this component.

A.2 Objective Requirements

A.2.1 Auditable Events for Objective Requirements

Table 14: Auditable Events for Objective Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FPT_IDV_EXT.1	No events specified	N/A
FPT_INT_EXT.1	Introspection initiated/enabled.	The VM introspected.
FPT_ML_EXT.1	Integrity initiated/enabled.	Integrity measurement values.

A.2.2 Class FPT: Protection of the TSF

FPT_IDV_EXT.1 Software Identification and Versions

FPT_IDV_EXT.1.1

The TSF shall include software identification (SWID) tags that contain a SoftwareIdentity element and an Entity element as defined in ISO/IEC 19770-2:2009.

FPT_IDV_EXT.1.2

The TSF shall store SWIDs in a.swidtag file as defined in ISO/IEC 19770-2:2009.

Application Note: SWID tags are XML files embedded within software that provide a standard method for IT departments to track and manage the software. The presence of SWIDs can greatly simplify the software management process and improve security by enhancing the ability of IT departments to manage updates.

Evaluation Activities

FPT_IDV_EXT.1
TSS
The evaluator shall examine the TSS to ensure it describes how SWID tags are implemented and the format of the tags. The evaluator shall verify that the format complies with FPT_IDV_EXT.1.1 and that SWIDs are stored in accordance with FPT_IDV_EXT.1.2.

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

Tests
The evaluator shall perform the following test: The evaluator shall check for the existence of SWID tags in a.swidtag file. The evaluator shall open the file and verify that each SWID contains at least a SoftwareIdentity element and an Entity element.

FPT_INT_EXT.1 Support for Introspection

FPT_INT_EXT.1.1

The TSF shall support a mechanism for permitting the VMM or privileged VMs to access the internals of another VM for purposes of introspection.

Application Note: Introspection can be used to support malware and anomaly detection from outside of the guest environment. This not only helps protect the guest OS, it also protects the VS by providing an opportunity for the VS to detect threats to itself that originate within VMs, and that may attempt to

break out of the VM and compromise the VMM or other VMs.

The hosting of malware detection software outside of the Guest VM helps protect the guest and helps ensure the integrity of the malware detection/antivirus software. This capability can be implemented in the VMM itself, but ideally it should be hosted by a Service VM so that it can be better contained and does not introduce bugs into the VMM.

Evaluation Activities

FPT_INT_EXT.1
TSS
The evaluator shall examine the TSS documentation to verify that it describes the interface for VM introspection and whether the introspection is performed by the VMM or another VM.

Guidance
The evaluator shall examine the operational guidance to ensure that it contains instructions for configuration of the introspection mechanism.

Tests
There are no test activities for this component.

FPT_ML_EXT.1 Measured Launch of Platform and VMM

FPT_ML_EXT.1.1

The TSF shall support a measured launch of the virtualization system. Measured components of the VS shall include the static executable image of the hypervisor and: [**selection**:

- *Static executable images of the Management Subsystem*
- ***[assignment**: list of (static images of) Service VMs]*
- ***[assignment**: list of configuration files]*
- *no other components*

]

FPT_ML_EXT.1.2

The TSF shall make the measurements selected in FPT_ML_EXT.1.1 available to the Management Subsystem.

Application Note: A measured launch of the platform and VS demonstrates that the proper TOE software was loaded. A measured launch process employs verifiable integrity measurement mechanisms. For example, a VS may hash components such as the hypervisor, service VMs, or the Management Subsystem. A measured launch process only allows components to be executed after the measurement has been recorded. An example process may add each component's hash before it is executed so that the final hash reflects the evidence of a component's state prior to execution. The measurement may be verified as the system boots, but this is not required.

The Platform is outside of the TOE. However, this requirement specifies that the VS must be capable of receiving Platform measurements if the Platform provides them. This requirement is requiring TOE support for Platform measurements if provided; it is not placing a requirement on the Platform to take such measurements.

If available, hardware should be used to store measurements in such a manner that they cannot be modified in any manner except to be extended. These measurements should be produced in a repeatable manner so that a third party can verify the measurements if given the inputs. Hardware devices, like Trusted Platform Modules (TPM), TrustZone, and MMU are some examples that may serve as foundations for storing and reporting measurements.

Platforms with a root of trust for measurement (RTM) should initiate the measured launch process. This may include core BIOS or the chipset. The chipset is the preferred RTM, but core BIOS or other firmware is acceptable. In a system without a traditional RTM, the first component that boots would be considered the RTM, this is not preferred.

Evaluation Activities

FPT_ML_EXT.1
TSS
The evaluator shall verify that the TSS or Operational Guidance describes how integrity measurements are performed and

made available to the Management Subsystem.

Guidance
The evaluator shall examine the operational guidance to verify that it documents how to access the measurements in the Management Subsystem.

Tests
The evaluator shall perform the following test: The evaluator shall start the VS, login as an administrator, and verify that the measurements for the specified components are viewable in the Management Subsystem.

A.3 Implementation-dependent Requirements

A.3.1 Auditable Events for Implementation-dependent Requirements

Table 15: Auditable Events for Implementation-dependent Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_CKM.2	No events specified	N/A
FCS_CKM_EXT.7	No events specified	N/A

A.3.2 Class FCS: Cryptographic Support

FCS_CKM.2 Cryptographic Key Distribution

This component must be included in the ST if the TOE implements any of the following features:
- ***Key Encapsulation Support***
- ***Key Wrap Support***

FCS_CKM.2.1

The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method [**selection**: *key encapsulation as specified in FCS_COP.1/KeyEncap*, *key wrapping as specified in FCS_COP.1/KeyWrap*] that meets the following: [*none*].

Application Note: If key encapsulation is selected, FCS_COP.1/KeyEncap is claimed. If key wrapping is selected, FCS_COP.1/KeyWrap is claimed.

Evaluation Activities

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

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

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

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

FCS_CKM_EXT.7 Cryptographic Key Agreement

This component must be included in the ST if the TOE implements any of the following features:

- ***Key Agreement Support***

FCS_CKM_EXT.7.1

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

Table 16 provides the allowable choices for completion of the selection operations of FCS_CKM_EXT.7.

Table 16: Allowable Choices for FCS_CKM_EXT.7

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

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

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

Evaluation Activities

FCS_CKM_EXT.7

TSS

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

KAS2

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

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

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

The evaluator shall generate a test group (i.e., set of tests) for each parameter value of the above parameter type with the largest number of supported values. For example, if the TOE supports all five Modulo values, then the evaluator shall generate five test groups. Each of the above supported parameter values must be included in at least one test group. Regardless of how many parameter values are supported, there must be at least two test groups. Half of the test groups are designated as Algorithm Functional Tests (AFT) and the remainder are designated as Validation Tests (VAT). If there is an odd number of groups, then the extra group is designated randomly as either AFT or VAT.

Algorithm Functional Test

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

Validation Test

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

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

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

FFC Diffie-Hellman Key Agreement

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

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

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

Algorithm Functional Test

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

Validation Test

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

Elliptic Curve Diffie-Hellman Key Agreement

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

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

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

Algorithm Functional Test

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

Validation Test

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

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 17: Auditable Events for Selection-based Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_COP.1/KeyEncap	No events specified	N/A
FCS_COP.1/KeyWrap	No events specified	N/A
FCS_COP.1/SKC	No events specified	N/A
FCS_COP.1/XOF	No events specified	N/A
FCS_HTTPS_EXT.1	Establishment/Termination of a HTTPS session.	Non-TOE endpoint of connection (IP address).
	Failure to establish a HTTPS Session.	• Reason for failure. • Non-TOE endpoint of connection (IP address) for failures.
FCS_IPSEC_EXT.1	Decisions to DISCARD or BYPASS network packets processed by the TOE.	• Presumed identity of source subject. • The entry in the SPD that applied to the decision.
	Failure to establish an IPsec SA.	• Source IP address, if applicable. • Identity of destination subject. • Reason for failure.
	Establishment/Termination of an IPsec SA.	• Identity of destination subject. • Transport layer protocol, if applicable. • Source subject service identifier, if applicable. • Non-TOE endpoint of connection (IP address) for both successes and failures.
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
FIA_AFL_EXT.1	Unsuccessful login attempts limit is met or exceeded.	Origin of attempt (e.g., IP address).
FIA_PMG_EXT.1	No events specified	N/A
FIA_PSK_EXT.1	No events specified	N/A
FIA_X509_EXT.4	No events specified	N/A
FPT_TUD_EXT.2	No events specified	N/A
FTP_TRP.1	Initiation of the trusted channel.	User ID and remote source (IP Address) if feasible.

	Termination of the trusted channel.	User ID and remote source (IP Address) if feasible.
	Failures of the trusted path functions.	User ID and remote source (IP Address) if feasible.

B.2 Class FCS: Cryptographic Support

FCS_COP.1/KeyEncap Cryptographic Operation - Key Encapsulation

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

- ***FCS_CKM.2.1***

FCS_COP.1.1/KeyEncap

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

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

Table 18: Allowable Choices for FCS_COP.1/KeyEncap

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

Application Note: This SFR is claimed when "key encapsulation" is selected in FCS_CKM.2.1. For this PP, the only anticipated use of key encapsulation is the use of ML-KEM as part of key establishment for trusted communications.

Evaluation Activities

FCS_COP.1/KeyEncap

TSS

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

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

ML-KEM Key Encapsulation

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

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

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

- *Previously generated ciphertext (c) [32 bytes]*

Encapsulation Test *For each supported parameter set the evaluator shall generate 25 test cases consisting of an encapsulation key ek and random value m. For each test case the valuator shall require the implementation under test to generate the corresponding shared secret k and ciphertext c. To determine correctness, the evaluator shall compare the resulting values with those generated using a known-good implementation using the same inputs.*
Encapsulation Key Check (if supported) *The evaluator shall generate 10 encapsulation keys such that:*

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

The evaluator shall invoke the TOE's Encapsulation Key Check functionality to determine the validity of the 10 keys. The unmodified keys should be determined valid, and the modified keys should be determined invalid.
Decapsulation Key Check (if supported) *The evaluator shall generate 10 decapsulation keys such that:*

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

The evaluator shall invoke the TOE's Decapsulation Key Check functionality to determine the validity of the 10 keys. The unmodified keys should be determined valid, and the modified keys should be determined invalid.
Decapsulation Test *For each supported parameter set the evaluator shall use a single previously generated decapsulation key dk and generate 10 test cases consisting of valid and invalid ciphertexts c. For each test case the evaluator shall require the implementation under test to generate the corresponding shared secret k whether or not the ciphertext is valid. To determine correctness, the evaluator shall compare the resulting values with those generated using a known-good implementation using the same inputs.*

FCS_COP.1/KeyWrap Cryptographic Operation - Key Wrapping

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

- ***FCS_CKM.2.1***

FCS_COP.1.1/KeyWrap

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

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

Table 19: Allowable choices for FCS_COP.1/KeyWrap

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

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

Application Note: This SFR is claimed when "key wrapping" is selected in FCS_CKM.2.1. NIST 800-57p1rev5 sec. 5.6.2 specifies that the size of key used to protect the key being transported should be at least the security strength of the key it is protecting.

Evaluation Activities

FCS_COP.1/KeyWrap

TSS

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

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

For tests of AES-GCM and AES-CCM, see testing for FCS_COP.1/AEAD. The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer. The tests must be executed on a platform that is as close as practically possible to the operational platform (but which may be instrumented in terms of, for example, use of a debug mode). Where the test is not carried out on the TOE itself, the test platform shall be identified and the differences between test environment and TOE execution environment shall be described.

AES-KW

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

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

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

Algorithm Functional Test *The evaluator shall generate 100 encryption test cases using random data for each combination of claimed key size, keyword cipher type, and six supported payload sizes such that the payload sizes include the minimum, the maximum, two that are divisible by 128, and two that are not divisible by 128. The results shall be compared with those generated by a known-good implementation using the same inputs. The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.*

AES-KWP

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

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

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

***Algorithm Functional Test** The evaluator shall generate 100 encryption test cases using random data for each combination of claimed key size, keyword cipher type, and six supported payload sizes such that the payload sizes include the minimum, the maximum, two that are divisible by 128, and two that are not divisible by 128. The results shall be compared with those generated by a known-good implementation using the same inputs. The evaluator shall generate 100 decryption test cases using the same parameters as above, but with 20 of each 100 test cases having modified ciphertext to produce an incorrect result. To determine correctness, the evaluator shall confirm that the results correspond as expected for both the modified and unmodified values.*

FCS_COP.1/SKC Cryptographic Operation (Encryption/Decryption)

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

- ***FMT_SMO_EXT.1.1***
- ***FCS_DTLSC_EXT.1.2, FCS_TLSC_EXT.1.2 from Functional Package for Transport Layer Security (TLS), version 2.1***

FCS_COP.1.1/SKC

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

Table 20 provides the allowable choices for completion of the selection operations of FCS_COP.1/SKC.

Table 20: Allowable choices for FCS_COP.1/SKC

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

[**selection**: *: ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A*] [CTR]

Application Note: This SFR is claimed if the TSF uses an algorithm defined by this SFR to enforce separation of management and operational networks as claimed in FMT_SMO_EXT.1.

Evaluation Activities

FCS_COP.1/SKC

TSS

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

Guidance

There are no additional Guidance evaluation activities for this component.

Tests

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

AES-CBC

Identifier	***Cryptographic Algorithm***	***Cryptographic Key Sizes***	***List of Standards***
AES-CBC	*AES in CBC mode with non-repeating and unpredictable IVs*	*256 bits*	*[**selection:** ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES]* *[**selection:** ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]*

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

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

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

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

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

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

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

```
}
```

where

`AES_KEY_SHUFFLE`

is defined as:

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

The above pseudocode is for encryption. For decryption, swap all instances of CT and PT. The initial IV, key, and plaintext/ciphertext should be random. The evaluator shall test the decrypt functionality using the same test as above, exchanging CT and PT, and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

XTS-AES

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

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

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

Single Data Unit Test *For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. The payload size and data unit size must be equal.* ***Multiple Data Unit Test*** *For each combination of claimed key size, direction, and supported tweak value format, the evaluator shall generate 50 test cases consisting of random payload data. The payload data size is determined randomly for each test case from supported values within the range [128-65536] bits. Likewise, the data unit size is determined randomly for each test case from supported values within the range [128-65535] bits. The payload size and data unit size must not be equal. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.*

AES-CTR

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

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

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

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

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

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

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

For each combination of direction and claimed key, the evaluator shall generate five test cases such that the data size is not a multiple of 128 bits. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated by a known good implementation using the same input parameters.
Counter Test *The evaluator shall generate a single message of 1000 blocks (128000 bits) and either encrypt or decrypt it. Back-compute the IVs used. Verify that they are unique and increasing (encryption) or decreasing (decryption).*

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

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

- ***FCS_COP.1.1/SigVer***

FCS_COP.1.1/XOF

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

Table 21 provides the allowable choices for completion of the selection operations of FCS_COP.1/XOF.

Table 21: Allowable choices for FCS_COP.1/XOF

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

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

Evaluation Activities

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

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

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

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

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

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

Algorithm Functional Test *For each supported function, generate test cases consisting of random data for every message length from 0 bits (if supported) to rate -1 bits, where rate equals 1088. Additionally, generate tests cases of random data for messages of every multiple of (rate+1) bits starting at length rate, and continuing until 65535 is exceeded.*
Monte Carlo Test *The Monte Carlo test takes in a single 128-bit message (SEED) and desired output length in bits, and runs 100 iterations of the chained computation. MaxOutBytes and MinOutBytes are the largest and smallest supported input and output sizes in bytes, respectively.*

```
Range = maxOutBytes - minOutBytes + 1
OutputLen = maxOutBytes
```

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

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

Variable Output Test *This test measures the ability of the TOE to generate output digests of varying sizes. The evaluator shall generate 512 test cases such that the input for each test case consists of 128- bits of random data, and the output length includes the minimum supported value, the maximum supported value, and 510 random values between the minimum and maximum digest sizes supported by the implementation.*

FCS_HTTPS_EXT.1 HTTPS Protocol

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

- ***FTP_ITC_EXT.1.1***

FCS_HTTPS_EXT.1.1

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

Application Note: This SFR is included in the ST if the ST author selects "TLS/HTTPS" in FTP_ITC_EXT.1.1.

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

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS.

Evaluation Activities

FCS_HTTPS_EXT.1
TSS
The evaluator shall check the TSS to ensure that it is clear on how HTTPS uses TLS to establish an administrative session, focusing on any client authentication required by the TLS protocol vs. security administrator authentication which may be done at a different level of the processing stack.

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

Tests
Testing for this activity is done as part of the TLS testing; this may result in additional testing if the TLS tests are done at the TLS protocol level.

FCS_IPSEC_EXT.1 IPsec Protocol

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

- ***FTP_ITC_EXT.1.1***

FCS_IPSEC_EXT.1.1

The TSF shall implement the IPsec architecture as specified in RFC 4301.

Application Note: In the following elements of the FCS_IPSEC_EXT.1 component, it is allowable for some or all of the individual elements to be implemented by the platform on which the VPN client operates. However, this is only the case when the platform is within the TOE boundary, as is the case

where this PP-Module is being claimed on top of a general-purpose OS or a mobile device.

When the TOE is a standalone software application, the IPsec functionality must be implemented by the TSF, though it is permissible for the TSF to invoke cryptographic algorithm services from the TOE platform to support the TOE's implementation of IPsec. The TOE may also rely on the TOE platform for X.509 certificate validation services, though it is the responsibility of the TSF to take the proper action based on the validation response that is returned.

It is also permissible for the TSF to rely on low-level capabilities of the platform to perform enforcement and routing functions as a result of the policies the TSF maintains. For example, while the TSF must provide the capability to implement the Security Policy Database (SPD) abstraction, it is permissible for the TSF to depend on the platform-provided network stack to perform the low-level packet filtering and routing actions once the TSF has set up those rules as defined by the SPD.

While enforcement of the IPsec requirements must be implemented by the TSF, it is permissible for the TSF to receive configuration of the IPsec behavior from an environmental source, most notably a VPN gateway.

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

While there must be a means to order the rules, a general approach to ordering is not mandated, as long as the TOE can distinguish the IP packets and apply the rules accordingly. There may be multiple SPDs (one for each network interface), but this is not required.

A VPN gateway fully implements the IPsec capability and provides an administrative interface to establish and populate an SPD. A VPN client is not required to provide an administrative interface to create or maintain an SPD.

As an alternative, a client may provide an interface that can be used by another application or network entity, such as a VPN gateway, as a means to establish and populate the SPD. In either of these cases (the client provides an administrative interface, or an API), while the client is expected to maintain the SPD abstraction, it is permitted for the low-level enforcement and routing activities to be implemented by platform capabilities (e.g., a network driver) as configured by the client.

FCS_IPSEC_EXT.1.2

The TSF shall implement [**selection**: *tunnel mode*, *transport mode*].

Application Note: If the TOE is used to connect to a VPN gateway for the purposes of establishing a secure connection to a private network, the ST author is expected to select tunnel mode. If the TOE uses IPsec to establish an end-to-end connection to another IPsec VPN client, the ST author is expected to select transport mode. If the TOE uses IPsec to establish a connection to a specific endpoint device for the purpose of secure remote administration, the ST author is expected to select transport mode.

FCS_IPSEC_EXT.1.3

The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [*AES-GCM-256 as specified in RFC 4106*].

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

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol: [*IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8247, and RFC 4868 for hash functions*].

FCS_IPSEC_EXT.1.6

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

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

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [*IKEv2 SA*] lifetimes can be configured by [**selection**: *an administrator*, *a VPN gateway*] based on [**selection**: *number of packets/number of bytes*, *length of time*]]. If length of time is used, it must include at least one option that is 24 hours or less for IKE SAs and eight hours or less for Child SAs.

Application Note: There is a selection that allows the ST author to specify which entity is responsible for "configuring" the life of the security association (SA). An implementation that allows an administrator to configure the client or a VPN gateway that pushes the SA lifetime down to the client are both acceptable.

As far as SA lifetimes are concerned, the TOE can limit the lifetime based on the number of bytes transmitted, or the number of packets transmitted. Either packet-based or volume-based SA lifetimes are acceptable; the ST author makes the appropriate selection to indicate which type of lifetime limits are supported.

For IKEv2, there are no hard-coded limits, therefore it is required that an administrator be able to configure the values. In general, instructions for setting the parameters of the implementation, including lifetime of the SAs, should be included in the operational guidance generated for AGD_OPE. It is appropriate to refine the requirement in terms of number of MB or KB instead of number of packets, as long as the TOE is capable of setting a limit on the amount of traffic that is protected by the same key (the total volume of all IPsec traffic protected by that key).

FCS_IPSEC_EXT.1.8

The TSF shall ensure that IKE implements DH Groups

- 20 (384-bit Random ECP) according to RFC 5114 and

[**selection**:

- *15 (3072-bit MODP) according to RFC 3526*
- *16 (4096-bit MODP) according to RFC 3526*
- *17 (6144-bit MODP) according to RFC 3526*
- *18 (8192-bit MODP) according to RFC 3526*
- *21 (521-bit Random ECP) according to RFC 5114*

].

Application Note: The selection is used to specify additional DH groups supported. This applies to IKEv2 exchanges. It should be noted that if any additional DH groups are specified, they must comply with the requirements (in terms of the ephemeral keys that are established) listed in FCS_CKM.1.

Since the implementation may allow different DH groups to be negotiated for use in forming the SAs, the assignments in FCS_IPSEC_EXT.1.9 and FCS_IPSEC_EXT.1.10 may contain multiple values. For each DH group supported, the ST author consults Table 2 in 800-57 to determine the "bits of security" associated with the DH group. Each unique value is then used to fill in the assignment (for 1.9 they are doubled; for they are inserted directly into the assignment). For example, suppose the implementation supports DH group 15 (3072-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 15 is 128, and for group 20 it is 192. For FCS_IPSEC_EXT.1.9, then, the assignment would read "[256, 384]" and for FCS_IPSEC_EXT.1.10 it would read "[128, 192]"

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE DH key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG.1 and having a length of at least [**assignment**: *(one or more) numbers of bits that is at least twice the "bits of security" value associated with the negotiated DH group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*] bits.

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**assignment**: *(one or more) "bits of security" values associated with the negotiated DH group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*].

FCS_IPSEC_EXT.1.11

The TSF shall ensure that [*IKEv2*] performs peer authentication using [*[**selection**: RSA, ECDSA] that use X.509v3 certificates that conform to RFC 4945 and [**selection**: Pre-shared Keys that conform to RFC 8784, no other method]*].

Application Note: At least one public-key-based peer authentication method is required in order to conform to this PP-Module; one or more of the public key schemes is chosen by the ST author to reflect what is implemented. The ST author also ensures that appropriate FCS requirements reflecting

the algorithms used (and key generation capabilities, if provided) are listed to support those methods. Note that the TSS will elaborate on the way in which these algorithms are to be used. X.509 certificates will be validated against FIA_X509_EXT.1 from Functional Package for X.509, version 1.0.

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

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the [**selection**: *IP address, Fully Qualified Domain Name (FQDN), user FQDN, Distinguished Name (DN)*] and [**selection**: *no other reference identifier type, [**assignment**: other supported reference identifier types]*] contained in a certificate does not match the expected values for the entity attempting to establish a connection.

Application Note: The TOE must support at least one of the following identifier types: IP address, FQDN, user FQDN, or DN. In the future, the TOE will be required to support all of these identifier types. The TOE is expected to support as many IP address formats (IPv4 and IPv6) as IP versions supported by the TOE in general. The ST author may assign additional supported identifier types in the second selection.

FCS_IPSEC_EXT.1.13

The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

Application Note: At this time, only the comparison between the presented identifier in the peer's certificate and the peer's reference identifier is mandated by the testing below. However, in the future, this requirement will address two aspects of the peer certificate validation: 1) comparison of the peer's ID payload to the peer's certificate, which are both presented identifiers, as required by RFC 4945 and 2) verification that the peer identified by the ID payload and the certificate is the peer expected by the TOE (per the reference identifier). At that time, the TOE will be required to demonstrate both aspects (i.e., that the TOE enforces that the peer's ID payload matches the peer's certificate, which both match configured peer reference identifiers).

Excluding the DN identifier type (which is necessarily the Subject DN in the peer certificate), the TOE may support the identifier in either the Common Name or Subject Alternative Name (SAN) or both. If both are supported, the preferred logic is to compare the reference identifier to a presented SAN, and only if the peer's certificate does not contain a SAN, to fall back to a comparison against the Common Name. In the future, the TOE will be required to compare the reference identifier to the presented identifier in the SAN only, ignoring the Common Name.

FCS_IPSEC_EXT.1.14

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

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

While it is acceptable for this capability to be configurable, the default configuration in the evaluated configuration (either "out of the box" or by configuration guidance in the AGD documentation) must enable this functionality.

Evaluation Activities

FCS_IPSEC_EXT.1
TSS
In addition to the TSS EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the TOE boundary includes a general-purpose OS or mobile device, the evaluator shall examine the TSS to ensure that it describes whether the VPN client capability is architecturally integrated with the platform itself or it is a separate executable that is bundled with the platform.

Guidance

In addition to the guidance EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the configuration of the IPsec behavior is from an environmental source, most notably a VPN gateway (e.g through receipt of required connection parameters from a VPN gateway), the evaluator shall ensure that the operational guidance contains any appropriate information for ensuring that this configuration can be properly applied.

Note, in this case, that the implementation of the IPsec protocol must be enforced entirely within the TOE boundary; i.e., it is not permissible for a software application TOE to be a graphical front-end for IPsec functionality implemented totally or in part by the underlying OS platform. The behavior referenced here is for the possibility that the configuration of the IPsec connection is initiated from outside the TOE, which is permissible so long as the TSF is solely responsible for enforcing the configured behavior. However, it is allowable for the TSF to rely on low-level, platform-provided networking functions to implement the SPD from the client (e.g., enforcement of packet routing decisions).

Tests

As a prerequisite for performing the Test EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall do the following:

The evaluator shall minimally create a test environment equivalent to the test environment illustrated below. It is expected that the traffic generator is used to construct network packets and will provide the evaluator with the ability to manipulate fields in the ICMP, IPv4, IPv6, UDP, and TCP packet headers. The evaluator shall provide justification for any differences in the test environment.

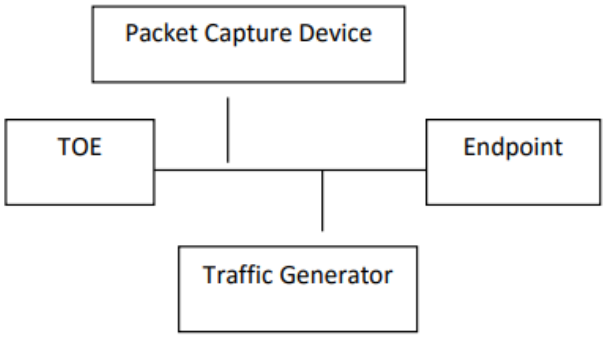


Figure 2: Test Environment

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

FCS_IPSEC_EXT.1.1

TSS

The evaluator shall examine the TSS and determine that it describes how the IPsec capabilities are implemented.

If the TOE is a standalone software application, the evaluator shall ensure that the TSS asserts that all IPsec functionality is implemented by the TSF. The evaluator shall also ensure that the TSS identifies what platform functionality the TSF relies on to support its IPsec implementation, if any (e.g., does it invoke cryptographic primitive functions from the platform's cryptographic library, enforcement of packet routing decisions by low-level network drivers).

If the TOE is part of a general-purpose desktop or mobile OS, the evaluator shall ensure that the TSS describes at a high level the architectural relationship between the VPN client portion of the TOE and the rest of the TOE (e.g., is the VPN client an integrated part of the OS or is it a standalone executable that is bundled into the OS package). If the SPD is implemented by the underlying platform in this case, then the TSS describes how the client interacts with the platform to establish and populate the SPD, including the identification of the platform's interfaces that are used by the client.

In all cases, the evaluator shall also ensure that the TSS describes how the client interacts with the network stack of the platforms on which it can run (e.g., does the client insert itself within the stack via kernel mods, does the client simply invoke APIs to gain access to network services).

The evaluator shall ensure that the TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how the available rules and actions form the SPD using terms defined in RFC 4301, such as BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet).

As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial, and the evaluator shall

determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Guidance
The evaluator shall examine the operational guidance to verify it describes how the SPD is created and configured. If there is an administrative interface to the client, then the guidance describes how the administrator specifies rules for processing a packet. The description includes all three cases - a rule that ensures packets are encrypted/decrypted, dropped, and allowing a packet to flow in plaintext. The evaluator shall determine that the description in the operational guidance is consistent with the description in the TSS, and that the level of detail in the operational guidance is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

If the client is configured by an external application, such as the VPN gateway, then the operational guidance should indicate this and provide a description of how the client is configured by the external application. The description should contain information as to how the SPD is established and set up in an unambiguous fashion. The description should also include what is configurable via the external application, how ordering of entries may be expressed, as well as the impacts that ordering of entries may have on the packet processing.

In either case, the evaluator shall ensure the description provided in the TSS is consistent with the capabilities and description provided in the operational guidance.

Tests
Depending on the implementation, the evaluator may be required to use a VPN gateway or some form of application to configure the client. For Test 2, the evaluator is required to choose an application that allows for the configuration of the full set of capabilities of the VPN client (in conjunction with the platform). For example, if the client provides a robust interface that allows for specification of wildcards, subnets, etc., it is unacceptable for the evaluator to choose a VPN gateway that only allows for specifying a single fully qualified IP addresses in the rule.

The evaluator shall perform the following tests:

- *Test FCS_IPSEC_EXT.1.1:1: The evaluator shall configure an SPD on the client that is capable of the following: dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the client with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator shall perform both positive and negative test cases for each type of rule. The evaluator shall observe via the audit trail and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed through without modification, was encrypted by the IPsec implementation.*
- *Test FCS_IPSEC_EXT.1.1:2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and operational guidance. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.*

FCS_IPSEC_EXT.1.2
TSS
The evaluator shall check the TSS to ensure it states that the VPN can be established to operate in tunnel mode, transport mode, or both (as selected).

Guidance
The evaluator shall confirm that the operational guidance contains instructions on how to configure the connection in each mode selected.

If both transport and tunnel modes are implemented, the evaluator shall review the operational guidance to determine how the use of a given mode is specified.

Tests
The evaluator shall perform the following tests based on the selections chosen:

- *Test FCS_IPSEC_EXT.1.2:1: [conditional]: If tunnel mode is selected, the evaluator shall use the operational guidance to configure the TOE and a VPN gateway to operate in tunnel mode. The evaluator shall configure the TOE and the VPN gateway to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure*

an allowable SA can be negotiated. The evaluator shall then initiate a connection from the client to connect to the VPN gateway peer. The evaluator shall observe (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

- *Test FCS_IPSEC_EXT.1.2:2: [conditional]: If transport mode is selected, the evaluator shall use the operational guidance to configure the TOE to operate in transport mode and also configures an IPsec peer to accept IPsec connections using transport mode. The evaluator shall configure the TOE and the endpoint device to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the remote endpoint. The evaluator shall observe (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.*
- *Test FCS_IPSEC_EXT.1.2:3: [conditional]: If both tunnel and transport modes are selected, the evaluator shall perform both Test 1 and Test 2 above, demonstrating that the TOE can be configured to support both modes.*
- *Test FCS_IPSEC_EXT.1.2:4: [conditional]: If both tunnel and transport modes are selected, the evaluator shall modify the testing for FCS_IPSEC_EXT.1 to include the supported mode for SPD PROTECT entries to show that they only apply to traffic that is transmitted or received using the indicated mode.*

FCS_IPSEC_EXT.1.3

TSS

The evaluator shall examine the TSS to verify that the TSS provides a description of how a packet is processed against the SPD and that if no "rules" are found to match, that a final rule exists, either implicitly or explicitly, that causes the network packet to be discarded.

Guidance

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

Tests

The evaluator shall perform the following test:

- *Test FCS_IPSEC_EXT.1.3:1: The evaluator shall configure the SPD such that it has entries that contain operations that DISCARD, PROTECT, and (if applicable) BYPASS network packets. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches a BYPASS entry and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE-created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was not permitted to flow to any of the TOE's interfaces.*

FCS_IPSEC_EXT.1.4

TSS

The evaluator shall examine the TSS to verify that the algorithm AES-GCM-256 is implemented. In addition, the evaluator shall ensure that the SHA-based HMAC algorithm conforms to the algorithms specified in the relevant iteration of FCS_COP.1/KeyedHash.

Guidance

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

Tests

- *Test FCS_IPSEC_EXT.1.4:1: The evaluator shall configure the TOE as indicated in the operational guidance, configuring the TOE to use the AES-GCM-256 algorithm, and attempt to establish a connection using ESP.*

FCS_IPSEC_EXT.1.5

TSS

The evaluator shall examine the TSS to verify that IKEv2 is implemented.

Guidance

The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE to support only IKEv2 (if necessary), and uses the guidance to configure the TOE to perform NAT traversal for the tests below.

Tests

- *Test FCS_IPSEC_EXT.1.5:1: The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 7296, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.*

- *Test FCS_IPSEC_EXT.1.5:2: The evaluator shall configure a remote peer to support IKEv1 only. If the TOE's supported versions of IKE is configurable, the evaluator shall follow the instructions specified in the operational guidance to ensure that only IKEv2 is supported. The evaluator shall then attempt to establish a connection between the TOE and that peer and verify the TSF rejects the connection attempt based on its lack of support for IKEv1.*

FCS_IPSEC_EXT.1.6

TSS

The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv2 payload and that the algorithm AES-GCM-256 is specified.

Guidance

The evaluator shall check the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through the TOE's default configuration (i.e., no configuration is necessary), direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Tests

The evaluator shall use the operational guidance to configure the TOE (or to configure the OE to have the TOE receive configuration) to perform the following test for the selected ciphersuite:

- *Test FCS_IPSEC_EXT.1.6:1: The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator shall confirm the algorithm was that used in the negotiation. The evaluator shall confirm that the connection is successful by confirming that data can be passed through the connection once it is established. For example, the evaluator may connect to a webpage on the remote network and verify that it can be reached.*

FCS_IPSEC_EXT.1.7

TSS

There are no TSS EAs for this requirement.

Guidance

The evaluator shall check the operational guidance to ensure it provides instructions on how the TOE configures the values for SA lifetimes. In addition, the evaluator shall check that the guidance has the option for either the administrator or VPN gateway to configure IKE SAs if time-based limits are supported. Currently, there are no values mandated for the number of packets or number of bytes; the evaluator shall simply check the operational guidance to ensure that this can be configured if selected in the requirement.

Tests

When testing this functionality, the evaluator shall ensure that both sides are configured appropriately. From the RFC "In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

Each of the following tests shall be performed:

- *Test FCS_IPSEC_EXT.1.7:1: [conditional]: The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.*
- *Test FCS_IPSEC_EXT.1.7:2: [conditional]: The evaluator shall construct a test where an IKEv2 IKE_SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.*
- *Test FCS_IPSEC_EXT.1.7:3: [conditional]: The evaluator shall perform a test similar to Test 2 for Child SAs, except that the lifetime will be 8 hours or less instead of 24 hours or less.*

FCS_IPSEC_EXT.1.8

TSS

The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator shall check to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Guidance

There are no guidance EAs for this requirement.

Tests
The evaluator shall perform the following test:

- *Test FCS_IPSEC_EXT.1.8:1: For each supported DH group, the evaluator shall test to ensure that IKEv2 can be successfully completed using that particular DH group.*

FCS_IPSEC_EXT.1.9
TSS
The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x" (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this EP is used, and that the length of "x" and the nonces meet the stipulations in the requirement.

Guidance
There are no guidance EAs for this requirement.

Tests
There are no test EAs for this requirement.

FCS_IPSEC_EXT.1.10
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.9.

FCS_IPSEC_EXT.1.11
TSS
The evaluator shall ensure that the TSS describes whether peer authentication is performed using RSA, ECDSA, or both.

If any selection with pre-shared keys is chosen in the selection, the evaluator shall check to ensure that the TSS describes how those selections work in conjunction with authentication of IPsec connections.

The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which fields of the certificate are used as the presented identifier (DN, Common Name, or SAN), and if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate.

Guidance
If any selection with "Pre-shared Keys" is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

If any method other than no other method is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

The evaluator shall ensure that the operational guidance describes how to set up the TOE to use the cryptographic algorithms RSA, ECDSA, or either, depending on which is claimed in the ST.

In order to construct the environment and configure the TOE for the following tests, the evaluator shall ensure that the operational guidance also describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE as a trusted CA.

The evaluator shall also ensure that the operational guidance includes the configuration of the reference identifiers for the peer.

Tests
For efficiency's sake, the testing that is performed here has been combined with the testing for FIA_X509_EXT.2 in Functional Package for X.509, version 1.0 and FCS_IPSEC_EXT.1.12, and FCS_IPSEC_EXT.1.13. The following tests shall be repeated for each peer authentication protocol selected in the FCS_IPSEC_EXT.1.11 selection above:

- *Test FCS_IPSEC_EXT.1.11:1: The evaluator shall have the TOE generate a public-private key pair and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE and the peer VPN used to establish a connection) for its signature. The values for the DN (Common Name, Organization, Organizational Unit, and Country) will also be passed in the request. Alternatively, the evaluator may import to the TOE a previously generated private key and corresponding certificate.*

- *Test FCS_IPSEC_EXT.1.11:2: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.*
- *Test FCS_IPSEC_EXT.1.11:3: The evaluator shall test that the TOE can properly handle revoked certificates – conditional on whether CRL or OCSP is selected; if both are selected, then a test is performed for each method. For this current version of the PP-Module, the evaluator has to only test one up in the trust chain (future drafts may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used and that the SA is established. The evaluator shall then attempt the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the TOE will not establish an SA.*
- *Test FCS_IPSEC_EXT.1.11:4: [conditional]: For each selection made, the evaluator shall verify factors are required, as indicated in the operational guidance, to establish an IPsec connection with the server.*

For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:

- *Test FCS_IPSEC_EXT.1.11:5: For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKEv2 authentication succeeds.*
- *Test FCS_IPSEC_EXT.1.11:6: For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKEv2 authentication fails.*
- *Test FCS_IPSEC_EXT.1.11:7: [conditional]: If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented certificate, but to match the Common Name in the peer's presented certificate and verify that the IKEv2 authentication fails.*
- *Test FCS_IPSEC_EXT.1.11:8: [conditional]: If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKEv2 authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKEv2 authentication fails. To demonstrate a comparison of DN values, the evaluator shall change any one of the four DN values and verify that the IKEv2 authentication fails.*
- *Test FCS_IPSEC_EXT.1.11:9: [conditional]: If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator shall repeat tests 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKEv2 authentication fails.*
- *Test FCS_IPSEC_EXT.1.11:10: [conditional]: If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKEv2 peer.*

FCS_IPSEC_EXT.1.12
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.11.

FCS_IPSEC_EXT.1.13
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.11.

FCS_IPSEC_EXT.1.14
TSS
The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA that is protecting the negotiation.

Guidance
There are no guidance EAs for this requirement.

Tests
The evaluator shall follow the guidance to configure the TOE to perform the following tests:

- *Test FCS_IPSEC_EXT.1.14:1: The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.*

- *Test FCS_IPSEC_EXT.1.14:2: [conditional]: The evaluator shall attempt to establish a IKEv2 Child SA that selects an encryption algorithm with more strength than that being used for the IKEv2 IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.*
- *Test FCS_IPSEC_EXT.1.14:3: The evaluator shall attempt to establish an IKEv2 IKE_SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.*
- *Test FCS_IPSEC_EXT.1.14:4: The evaluator shall attempt to establish an IKEv2 Child SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.*

FCS_RBG.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 [*384 bits of min-entropy*] from a TSF interface for obtaining entropy.

Application Note: This SFR is claimed if "TSF interface for obtaining entropy" is selected in FCS_RBG.1.2, i.e., the TOE's entropy source is outside the TOE boundary.

If the environmental entropy source produces full entropy, the amount of entropy data collected is equal to the amount specified in the selection. If the entropy source does not assure full entropy (i.e., the ratio of min-entropy to collected data is less than 1:1), then the TSF must collect sufficient additional entropy data and perform a derivation function on it to reduce the collected data to a seed with the claimed min-entropy. In this case, the entropy documentation is expected to identify the min-entropy rate of the source data, how much data the TSF collects, and what derivation function is performed on this collected data to ensure that it has the claimed min-entropy.

One should not apply NIST SP 800-90B (or AIS-31) statistical tests against an external entropy source since the TSF is unable to enforce entropy requirements or conditioning requirements against something outside of its logical boundary. However, the TSS may include estimates for min-entropy from external sources that contribute to the overall entropy requirements for the DRBG.

Evaluation Activities

FCS_RBG.2
The evaluator shall examine the entropy documentation required by FCS_RBG.1 to verify that it identifies, for each DRBG function implemented by the TOE, the TSF external interface that is 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 the TSF can derive a seed from it that has the claimed amount of min-entropy. If the TSF uses an entropy source that can be assumed to have full entropy, the seed is the output data from the entropy source. If the TSF uses an entropy source where the min-entropy of the data is less than 1:1, the evaluator shall verify that the entropy documentation identifies how large the collected data sample is and how TSF derives this data into a seed that has the claimed min-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 DRBG using a [*TSF software-based entropy source*] [**assignment**: *name of entropy source*] with [*a minimum of 384*] bits of min-entropy.

Application Note: This SFR is claimed if "TSF entropy source..." is selected in FCS_RBG.1.2, i.e., the TOE has a single entropy source that is inside the TOE boundary. If the TOE obtains seed material from an environmental component, this is specified using FCS_RBG.2.

If the TSF entropy source produces full entropy, the amount of entropy data collected is equal to the amount specified in the selection. If the entropy source does not assure full entropy (i.e., the ratio of min-entropy to collected data is less than 1:1), then the TSF must collect sufficient additional entropy data and perform a derivation function on it to reduce the collected data to a seed with the claimed min-entropy. In this case, the entropy documentation is expected to identify the min-entropy rate of the source data, how much data the TSF collects, and what derivation function is performed on this collected data to ensure that it has the claimed min-entropy.

One can apply NIST SP 800-90B (or AIS-31) statistical tests against internal entropy sources (aka raw entropy) to confirm the min-entropy of the entropy source.

Evaluation Activities

FCS_RBG.3

The evaluator shall examine the entropy documentation required by FCS_RBG.1 to verify that it identifies, for each DRBG function implemented by the TOE, the TSF entropy source that is 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 the TSF can derive a seed from it that has the claimed amount of min-entropy. If the TSF uses an entropy source that can be assumed to have full entropy, the seed is the output data from the entropy source. If the TSF uses an entropy source where the min-entropy of the data is less than 1:1, the evaluator shall verify that the entropy documentation identifies how large the collected data sample is and how TSF derives this data into a seed that has the claimed min-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 DRBG using [*[**assignment**: number] **independent** TSF software-based entropy sources*].

Application Note: This SFR is claimed if "multiple independent TSF entropy sources..." is selected in FCS_RBG.1.2, i.e., the TSF performs some sort of operation to combine data from multiple distinct sources into a value that is used to derive a seed. Since this PP defines the TOE as software, it does not have any hardware components and therefore would only claim software-based entropy sources here. If the TOE obtains seed material from any hardware components, this is specified using FCS_RBG.2. FCS_RBG.5 defines the mechanism by which all 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 to verify that it identifies, for each DRBG function implemented by the TOE, each TSF entropy source used to seed the TOE's DRBG.

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 Entropy 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 entropy source(s), input from TSF interface(s) for obtaining entropy*] resulting in a minimum of [*384*] bits of min-entropy to create the entropy input into the derivation function as defined in [**assignment**: *list of standards*].

Application Note: This SFR is claimed if "multiple TSF entropy sources..." is selected in FCS_RBG.1.2, i.e., the TSF performs some sort of operation to combine data from multiple distinct sources into a value that is used to derive a seed.

If the environmental entropy source produces full entropy, the amount of entropy data collected is equal to the amount specified in the selection. If the entropy source does not assure full entropy (i.e., the ratio of min-entropy to collected data is less than 1:1), then the TSF must collect sufficient additional entropy data and perform a derivation function on it to reduce the collected data to a seed with the claimed min-entropy. In this case, the entropy documentation is expected to identify the min-entropy rate of the source data, how much data the TSF collects, and what derivation function is performed on this collected data to ensure that it has the claimed min-entropy.

One can apply NIST SP 800-90B (or AIS-31) statistical tests against internal entropy sources (aka raw entropy) to confirm the min-entropy of the entropy sources either in aggregate or individually.

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 to verify that it describes the method by which data from these sources are combined into a single seed. This should include an estimation of the rate at which each entropy 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 amount of sampled data, the min-entropy rate of the sampled data, and the method by which the multiple sampled data sources are combined can be used to determine that the TSF can derive a seed from it that has the claimed amount of min-entropy. This includes any description of how the TSF derives data larger than the seed value into a seed that has the claimed min-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 Class FIA: Identification and Authentication

FIA_AFL_EXT.1 Authentication Failure Handling

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

- ***FIA_UAU.5.1***

FIA_AFL_EXT.1.1

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

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

] unsuccessful authentication attempts occur related to administrators attempting to authenticate remotely using [**selection**: *username and password, username and PIN*].

FIA_AFL_EXT.1.2

When the defined number of unsuccessful authentication attempts has been met, the TSF shall: [**selection**: *prevent the offending administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until [**assignment**: action to unlock] is taken by an administrator, prevent the offending administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until an administrator-defined time period has elapsed*].

Application Note: This requirement is claimed if the selection for "[local] authentication based on username and password" or "authentication based on username and a PIN..." is selected in FIA_UAU.5.1. Because this PP assumes physical access to the TOE is secured (A.PHYSICAL), this SFR only applies to interfaces that are used to access the TOE remotely; a local physical console, for example, does not require a lockout mechanism.

When the TOE uses directory-based password authentication, it is expected that account lockout is enforced by the directory, and this behavior is defined in FIA_UAU.5.2.

Evaluation Activities

FIA_AFL_EXT.1

TSS

The evaluator shall examine the TSS to verify that it describes the authentication mechanisms that authentication failure lockout applies to and whether the failure threshold is fixed or configurable. If configurable, the evaluator shall verify that athe TSS identifies the range of supported values. The evaluator shall also verify that the TSS identifies whether the failure state is resolved through administrative action or whether it persists for a configurable amount of time (and what the supported time range is if this applies).

Guidance

The evaluator shall verify that if the range of authentication failures that trigger a lockout is configurable, the operational guidance describes how to configure this parameter and what the range of acceptable values is for it. This is also covered by FMT_MOF_EXT.1, function 15.

If a locked account can be unlocked through administrative action, the evaluator shall verify that the operational guidance includes instructions for how to unlock this account.

If a locked account can be unlocked automatically through a configurable time period, the evaluator shall verify that the operational guidance includes instructions for how to configure this time period, as well as the minimum and maximum values for it.

Tests

The evaluator shall perform the following tests for each credential selected in FIA_AFL_EXT.1.1:

The evaluator shall set an administrator-configurable threshold n for failed attempts, or note the ST-specified assignment.

- *Test FIA_AFL_EXT.1:1: The evaluator shall attempt to authenticate remotely with the credential n-1 times. The evaluator will then attempt to authenticate using a good credential and verify that authentication is successful.*
- *Test FIA_AFL_EXT.1:2: The evaluator shall make n attempts to authenticate using a bad credential. The evaluator will then attempt to authenticate using a good credential and verify that the attempt is unsuccessful. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.*

After reaching the limit for unsuccessful authentication attempts The evaluator shall proceed as follows:

- *Test FIA_AFL_EXT.1:3: If the administrator action selection in FIA_AFL_EXT.1.2 is selected, then the evaluator will confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote administrator's access results in successful access (when using valid credentials for that administrator).*
- *Test FIA_AFL_EXT.1:4: If the time period selection in FIA_AFL_EXT.1.2 is selected, The evaluator shall wait for just less than the time period configured and show that an authentication attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured and show that an authentication attempt using valid credentials results in successful access.*

FIA_PMG_EXT.1 Password Management

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

- ***FIA_UAU.5.1***

FIA_PMG_EXT.1.1

The TSF shall provide the following password management capabilities for administrative passwords:

a. Passwords shall be able to be composed of any combination of upper and lower case characters, digits, and the following special characters: [**selection**: *"!"*, *"@"*, *"#"*, *"$"*, *"%"*, *"^"*, *"& "*, *"*"*, *"("*, *")"*, ***[assignment**: other characters]*]
b. Minimum password length shall be configurable
c. Passwords of at least 15 characters in length shall be supported

Application Note: This SFR is included in the ST if the ST author selects "[local] authentication based on username and password" in FIA_UAU.5.1. If the TOE uses directory-based authentication, it is expected that password composition requirements are enforced by the directory server.

The ST author selects the special characters that are supported by the TOE; they may optionally list additional special characters supported using the assignment. "Administrative passwords" refers to passwords used by administrators to gain access to the Management Subsystem.

Evaluation Activities

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

Guidance
The evaluator shall examine the operational guidance to determine that it provides guidance to security administrators in the composition of strong passwords, and that it provides instructions on setting the minimum password length.

Tests
The evaluator shall also perform the following test: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible combinations of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

FIA_PSK_EXT.1 Pre-Shared Key Composition

The inclusion of this selection-based component depends upon selection in:
- ***FCS_IPSEC_EXT.1.11***

FIA_PSK_EXT.1.1

The TSF shall be able to use pre-shared keys for [*IKEv2*].

FIA_PSK_EXT.1.2

The TSF shall be able to accept the following as pre-shared keys: [*generated bit-based*] keys.

Evaluation Activities

FIA_PSK_EXT.1
TSS
The evaluator shall examine the TSS to ensure that it identifies all protocols that allow pre-shared keys. For each protocol identified by the requirement, the evaluator shall confirm that the TSS states which pre-shared key selections are supported.

Guidance
The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure pre-shared keys if any configuration is required.

The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure the mandatory_or_not flag per RFC 8784.

Tests
The evaluator shall attempt to establish a connection and confirm that the connection requires the selected factors in the PSK to establish the connection in alignment with table 1 from RFC 8784.

FIA_X509_EXT.4 Exceptions to X.509 Certificate Revocation Checking

The inclusion of this selection-based component depends upon selection in:
- ***FIA_X509_EXT.1.4 from Functional Package for X.509, version 1.0***

FIA_X509_EXT.4.1

The TSF shall provide alternate functionality to standard X.509 certificate revocation checking for the following exceptions: [**selection**: *firmware checking of updates: invalidate automatic updates if the firmware certificate is compromised, [**assignment**: other exceptions and corresponding functionality]*].

Application Note: This selection-based requirement is claimed when FIA_X509_EXT.1.4 in Functional Package for X.509, version 1.0 selects "not obtain revocation status information by the TSF due to..." This SFR identifies the specific exceptions where revocation status is not obtained.

Evaluation Activities

FIA_X509_EXT.4
TSS
The evaluator shall ensure the TSS describes, for each exception, the alternate functionality the TOE implements to handle the lack of certificate revocation. The description must be consistent with the selection in the requirement.

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

Tests
The evaluator shall perform the following test: For each exception, the evaluator shall configure the TSF as necessary to meet the exceptional condition described, and exercise the function using a certificate chain without revocation information. The evaluator shall attempt to examine TSF logging or behavior as described in the TSS to confirm the alternative action described is performed. For example, in the case of firmware updates that invalidate automatic updates, the evaluator shall invoke an automatic update and observe that the update is not performed. In other cases, the TSS describes the alternative action.

B.4 Class FPT: Protection of the TSF

FPT_TUD_EXT.2 Trusted Update Based on Certificates

The inclusion of this selection-based component depends upon selection in:
- ***FPT_TUD_EXT.1.3***

FPT_TUD_EXT.2.1

The TSF shall not install an update if the code signing certificate is deemed invalid.

Application Note: Certificates may optionally be used for code signing of system software updates (FPT_TUD_EXT.1.3). This element must be included in the ST if certificates are used for validating updates. If “code signing for system software updates” is selected in FIA_X509_EXT.2.1 from the Functional Package for X.509, version 1.0, FPT_TUD_EXT.2 must be included in the ST.

Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with FIA_X509_EXT.1 in the Functional Package for X.509, version 1.0.

Evaluation Activities

FPT_TUD_EXT.2
TSS
There are no additional TSS evaluation activities for this component.

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

Tests
The evaluation activity for this requirement is performed in conjunction with the evaluation activity for FIA_X509_EXT.1 and FIA_X509_EXT.2 in the Functional Package for X.509, version 1.0.

B.5 Class FTP: Trusted Path/Channels

FTP_TRP.1 Trusted Path

The inclusion of this selection-based component depends upon selection in:
- ***FMT_MOF_EXT.1.1,***
- ***FTP_ITC_EXT.1.1***

FTP_TRP.1.1

The TSF shall **use a trusted channel as specified in FTP_ITC_EXT.1 to** provide a **trusted** communication path between itself and [*remote*] **administrators** that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from [*modification, disclosure*].

FTP_TRP.1.2

The TSF shall permit [*remote **administrators***] to initiate communication via the trusted path.

FTP_TRP.1.3

The TSF shall require the use of the trusted path for [*[all remote administration actions]*].

Application Note: This SFR is included in the ST if "remote" is selected in FMT_MOF_EXT.1.1.

Protocols used to implement the remote administration trusted channel must be selected in FTP_ITC_EXT.1.

This requirement ensures that authorized remote administrators initiate all communication with the TOE via a trusted path, and that all communications with the TOE by remote administrators is performed over this path. The data passed in this trusted communication channel are encrypted as defined the protocol chosen in the first selection in FTP_ITC_EXT.1. The ST author chooses the mechanism or mechanisms supported by the TOE, and then ensures that the detailed requirements in Appendix B corresponding to their selection are copied to the ST if not already present.

Evaluation Activities

FTP_TRP.1
TSS
The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Guidance
The evaluator shall confirm that the operational guidance contains instructions for establishing the remote administrative sessions for each supported method.

Tests
The evaluator shall also perform the following tests:

- *Test FTP_TRP.1:1: The evaluators shall ensure that communications using each specified (in the operational guidance) remote administration method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.*

- *Test FTP_TRP.1:2: For each method of remote administration supported, the evaluator shall follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish remote administrative sessions without invoking the trusted path.*
- *Test FTP_TRP.1:3: The evaluator shall ensure, for each method of remote administration, the channel data is not sent in plaintext.*
- *Test FTP_TRP.1:4: The evaluator shall ensure, for each method of remote administration, modification of the channel data is detected by the TOE.*

Additional evaluation activities are associated with the specific protocols.

Appendix C - Extended Component Definitions

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

C.1 Extended Components Table

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

Table 22: Extended Component Definitions

Functional Class	Functional Components
Class FCS: Cryptographic Support	FCS_CKM_EXT Cryptographic Key Management FCS_ENT_EXT Entropy for Virtual Machines FCS_HTTPS_EXT HTTPS Protocol FCS_IPSEC_EXT IPsec Protocol
Class FDP: User Data Protection	FDP_HBI_EXT Hardware-Based Isolation Mechanisms FDP_PPR_EXT Physical Platform Resource Controls FDP_RIP_EXT Residual Information in Memory FDP_VMS_EXT VM Separation FDP_VNC_EXT Virtual Networking Components
Class FIA: Identification and Authentication	FIA_AFL_EXT Authentication Failure Handling FIA_PMG_EXT Password Management FIA_PSK_EXT Pre-Shared Key Composition FIA_UIA_EXT Administrator Identification and Authentication FIA_X509_EXT X.509 Certificate
Class FMT: Security Management	FMT_MOF_EXT Management of Security Functions Behavior FMT_SMO_EXT Separation of Management and Operational Networks
Class FPT: Protection of the TSF	FPT_DDI_EXT Device Driver Isolation FPT_DVD_EXT Non-Existence of Disconnected Virtual Devices FPT_EEM_EXT Execution Environment Mitigations FPT_GVI_EXT Guest VM Integrity FPT_HAS_EXT Hardware Assists FPT_HCL_EXT Hypercall Controls FPT_IDV_EXT Software Identification and Versions FPT_INT_EXT Support for Introspection FPT_ML_EXT Measured Launch of Platform and VMM FPT_RDM_EXT Removable Devices and Media FPT_TUD_EXT Trusted Updates FPT_VDP_EXT Virtual Device Parameters FPT_VIV_EXT VMM Isolation from VMs
Class FTP: Trusted Path/Channels	FTP_ITC_EXT Trusted Channel Communications FTP_UIF_EXT User Interface

C.2 Extended Component Definitions

C.2.1 Class FCS: Cryptographic Support

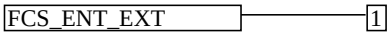
This PP defines the following extended components as part of the class originally defined by CC Part 2:

C.2.1.1 FCS_ENT_EXT Entropy for Virtual Machines

Family Behavior

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

Component Leveling



FCS_ENT_EXT.1, Entropy for Virtual Machines, requires the TSF to provide entropy data to VMs in a specified manner.

Management: FCS_ENT_EXT.1

There are no management functions foreseen.

Audit: FCS_ENT_EXT.1

There are no audit events foreseen.

FCS_ENT_EXT.1 Entropy for Virtual Machines

Hierarchical to: No other components.

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

FCS_ENT_EXT.1.1

The TSF shall provide a mechanism to make entropy available to VMs that is generated obtained via mechanisms specified in FCS_RBG.1.2 through [**selection**: *hypercall interface*, *virtual device interface*, *pass-through access to hardware entropy source*].

FCS_ENT_EXT.1.2

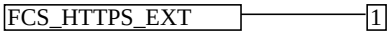
The TSF shall provide independent entropy across multiple VMs.

C.2.1.2 FCS_HTTPS_EXT HTTPS Protocol

Family Behavior

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

Component Leveling



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

Management: FCS_HTTPS_EXT.1

There are no management functions foreseen.

Audit: FCS_HTTPS_EXT.1

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

a. Failure to establish an HTTPS session.
b. Establishment/termination of an HTTPS session.

FCS_HTTPS_EXT.1 HTTPS Protocol

Hierarchical to: No other components.

Dependencies to: [FCS_TLSC_EXT.1 TLS Client Protocol, or
FCS_TLSC_EXT.2 TLS Client Protocol with Mutual Authentication, or
FCS_TLSS_EXT.1 TLS Server Protocol, or
FCS_TLSS_EXT.2 TLS Server Protocol with Mutual Authentication]

FCS_HTTPS_EXT.1.1

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

FCS_HTTPS_EXT.1.2

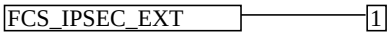
The TSF shall implement HTTPS using TLS.

C.2.1.3 FCS_IPSEC_EXT IPsec Protocol

Family Behavior

This family defines requirements for protecting communications using IPsec.

Component Leveling



FCS_IPSEC_EXT.1, IPsec Protocol, requires the TSF to securely implement the IPsec protocol.

Management: FCS_IPSEC_EXT.1

The following actions could be considered for the management functions in FMT:

- Specify VPN gateways to use for connections
- Specify IPsec VPN clients to use for connections
- Specify IPsec-capable network devices to use for connections
- Specify client credentials to be used for connections

Audit: FCS_IPSEC_EXT.1

The following actions should be auditable if FAU_GEN Security Audit Data Generation is included in the PP/ST:

- Decisions to DISCARD or BYPASS network packets processed by the TOE
- Failure to establish an IPsec SA
- Establishment/Termination of an IPsec SA

FCS_IPSEC_EXT.1 IPsec Protocol

Hierarchical to:	No other components.
Dependencies to:	FCS_CKM.1 Cryptographic Key Generation
	FCS_CKM.2 Cryptographic Key Distribution
	FCS_COP.1 Cryptographic Operation

FCS_IPSEC_EXT.1.1

The TSF shall implement the IPsec architecture as specified in RFC 4301.

FCS_IPSEC_EXT.1.2

The TSF shall implement [**selection**: *tunnel mode*, *transport mode*].

FCS_IPSEC_EXT.1.3

The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [**assignment**: *supported cryptographic algorithms*].

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol: [**assignment**: *key exchange protocol*].

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the [**assignment**: *key exchange protocol*] protocol uses the cryptographic algorithms [**assignment**: *supported cryptographic algorithms*] and no other algorithm.

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [**assignment**: *key exchange protocol*] lifetimes can be configured by [**assignment**: *authorized subjects*] based on [**assignment**: *values or metrics for maximum validity of negotiated keys*]. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and eight hours or less for Phase 2 SAs.

FCS_IPSEC_EXT.1.8

The TSF shall ensure that IKE implements DH Groups

- 20 (384-bit Random ECP) according to RFC 5114 and

[**selection**:

- *15 (3072-bit MODP) according to RFC 3526*
- *16 (4096-bit MODP) according to RFC 3526*
- *17 (6144-bit MODP) according to RFC 3526*
- *18 (8192-bit MODP) according to RFC 3526*
- *21 (521-bit Random ECP) according to RFC 5114*

].

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE DH key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG.1, and having a length of at least [**assignment**: *number of bits*] bits.

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**assignment**: *(one or more) "bits of security" values associated with the negotiated DH group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*].

FCS_IPSEC_EXT.1.11

The TSF shall ensure that [**assignment**: *key exchange protocol*] performs peer authentication using [**assignment**: *supported authentication mechanisms*].

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the [**selection**: *IP address, Fully Qualified Domain Name (FQDN), user FQDN, Distinguished Name (DN)*] and [**selection**: *no other reference identifier type, [**assignment**: other supported reference identifier types]*] contained in a certificate does not match the expected values for the entity attempting to establish a connection.

FCS_IPSEC_EXT.1.13

The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

FCS_IPSEC_EXT.1.14

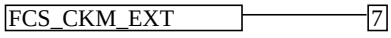
The [**selection**: *TSF, VPN gateway*] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [**selection**: *IKEv1 Phase 1, IKEv2 IKE_SA*] connection is greater than or equal to the strength of the symmetric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

C.2.1.4 FCS_CKM_EXT Cryptographic Key Management

Family Behavior

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

Component Leveling



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

Management: FCS_CKM_EXT.7

There are no management functions foreseen.

Audit: FCS_CKM_EXT.7

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

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

FCS_CKM_EXT.7 Cryptographic Key Agreement

Hierarchical to: No other components.

Dependencies to:

[FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
[FCS_CKM.1/AKG Cryptographic key generation, or
FCS_CKM.5 Cryptographic key derivation, or
FCS_CKM_EXT.8 Password-based key derivation],
[FCS_CKM_EXT.7 Cryptographic key agreement, or
FCS_COP.1 Cryptographic operation]
FCS_CKM.6 Timing and event of cryptographic key destruction
[FCS_COP.1/CMAC CMAC, or
FCS_COP.1/Hash Hashing, or
FCS_COP.1/KeyedHash Keyed Hashing, or
FCS_COP.1/SKC Symmetric Key Cryptography, or
FCS_COP.1/AEAD Authenticated Encryption with Associated Data]

FCS_CKM_EXT.7.1

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

C.2.2 Class FDP: User Data Protection

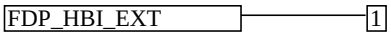
This PP defines the following extended components as part of the class originally defined by CC Part 2:

C.2.2.1 FDP_HBI_EXT Hardware-Based Isolation Mechanisms

Family Behavior

This family defines requirements for isolation of Guest VMs from the hardware resources of the physical device on which the Guest VMs are deployed.

Component Leveling



FDP_HBI_EXT.1, Hardware-Based Isolation Mechanisms, requires the TSF to identify the mechanisms used to isolate Guest VMs from platform hardware resources.

Management: FDP_HBI_EXT.1

There are no management functions foreseen.

Audit: FDP_HBI_EXT.1

There are no audit events foreseen.

FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

Hierarchical to: No other components.

Dependencies to: FDP_VMS_EXT.1 VM Separation

FDP_HBI_EXT.1.1

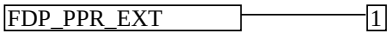
The TSF shall use [**assignment**: *list of platform-provided, hardware-based mechanisms*] to constrain a Guest VM's direct access to the following physical devices: [**selection**: *all devices, all devices except for [**assignment**: list of devices to which Guest VMs may access]*].

C.2.2.2 FDP_PPR_EXT Physical Platform Resource Controls

Family Behavior

This family defines requirements for the physical resources that the TOE will allow or prohibit Guest VMs to access.

Component Leveling



FDP_PPR_EXT.1, Physical Platform Resource Controls, requires the TSF to define the hardware resources that Guest VMs may always access, may never access, and may conditionally access based on administrative configuration.

Management: FDP_PPR_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure VM access to physical devices.

Audit: FDP_PPR_EXT.1

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

a. Successful and failed VM connections to physical devices where connection is governed by configurable policy.
b. Security policy violations.

FDP_PPR_EXT.1 Physical Platform Resource Controls

Hierarchical to:	No other components.
Dependencies to:	FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms
	FMT_SMR.1 Security Roles

FDP_PPR_EXT.1.1

The TSF shall allow an authorized administrator to control Guest VM access to the following physical platform resources: [**assignment**: *list of physical platform resources the VMM is able to control access to*].

FDP_PPR_EXT.1.2

The TSF shall explicitly deny all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources, [**assignment**: list of physical platform resources to which access is explicitly denied]*].

FDP_PPR_EXT.1.3

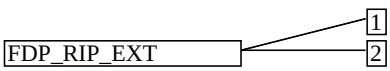
The TSF shall explicitly allow all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources, [**assignment**: list of physical platform resources to which access is always allowed]*].

C.2.2.3 FDP_RIP_EXT Residual Information in Memory

Family Behavior

This family defines requirements for ensuring that allocation of data to a Guest VM does not cause a disclosure of residual data from a previous VM.

Component Leveling



FDP_RIP_EXT.1, Residual Information in Memory, requires the TSF to ensure that physical memory is cleared to zeroes prior to its allocation to a Guest VM.

FDP_RIP_EXT.2, Residual Information on Disk, requires the TSF to ensure that physical disk storage is cleared upon allocation to a Guest VM.

Management: FDP_RIP_EXT.1

There are no management functions foreseen.

Audit: FDP_RIP_EXT.1

There are no audit events foreseen.

FDP_RIP_EXT.1 Residual Information in Memory

Hierarchical to: No other components.

Dependencies to:

FDP_RIP_EXT.1.1

The TSF shall ensure that any previous information content of physical memory is cleared prior to allocation to a Guest VM.

Management: FDP_RIP_EXT.2

There are no management functions foreseen.

Audit: FDP_RIP_EXT.2

There are no audit events foreseen.

FDP_RIP_EXT.2 Residual Information on Disk

Hierarchical to: No other components.

Dependencies to:

FDP_RIP_EXT.2.1

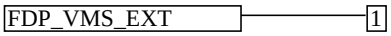
The TSF shall ensure that any previous information content of physical disk storage is cleared to zeroes upon allocation to a Guest VM.

C.2.2.4 FDP_VMS_EXT VM Separation

Family Behavior

This family defines requirements for the logical separation of multiple Guest VMs that are managed by the same virtualization system.

Component Leveling



FDP_VMS_EXT.1, VM Separation, requires the TSF to maintain logical separation between Guest VMs except through the use of specific configurable methods.

Management: FDP_VMS_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure inter-VM data sharing.

Audit: FDP_VMS_EXT.1

There are no audit events foreseen.

FDP_VMS_EXT.1 VM Separation

Hierarchical to: No other components.

Dependencies to:

FDP_VMS_EXT.1.1

The VS shall provide the following mechanisms for transferring data between Guest VMs: [**selection**:

- *no mechanism*
- *virtual networking*
- *[**assignment**: other inter-VM data sharing mechanisms]*

].

FDP_VMS_EXT.1.2

The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs.

FDP_VMS_EXT.1.3

The TSF shall allow administrators to configure the mechanisms selected in FDP_VMS_EXT.1.1 to enable and disable the transfer of data between Guest VMs.

FDP_VMS_EXT.1.4

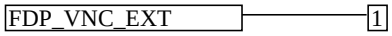
The VS shall ensure that no Guest VM is able to read or transfer data to or from another Guest VM except through the mechanisms listed in FDP_VMS_EXT.1.1.

C.2.2.5 FDP_VNC_EXT Virtual Networking Components

Family Behavior

This family defines requirements for configuration of virtual networking between Guest VMs that are managed by the virtualization system.

Component Leveling



FDP_VNC_EXT.1, Virtual Networking Components, requires the TSF to support the configuration of virtual networking between Guest VMs.

Management: FDP_VNC_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure virtual networks including VM.

Audit: FDP_VNC_EXT.1

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

a. Successful and failed attempts to connect VMs to virtual and physical networking components.
b. Security policy violations.
c. Administrator configuration of inter-VM communications channels between VMs.

FDP_VNC_EXT.1 Virtual Networking Components

Hierarchical to:	No other components.
Dependencies to:	FDP_VMS_EXT.1 VM Separation
	FMT_SMR.1 Security Roles

FDP_VNC_EXT.1.1

The TSF shall allow administrators to configure virtual networking components to connect VMs to each other and to physical networks.

FDP_VNC_EXT.1.2

The TSF shall ensure that network traffic visible to a Guest VM on a virtual network–or virtual segment of a physical network–is visible only to Guest VMs configured to be on that virtual network or segment.

C.2.3 Class FIA: Identification and Authentication

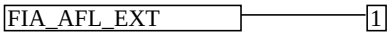
This PP defines the following extended components as part of the class originally defined by CC Part 2:

C.2.3.1 FIA_AFL_EXT Authentication Failure Handling

Family Behavior

This family defines requirements for detection and prevention of brute force authentication attempts.

Component Leveling



FIA_AFL_EXT.1, Authentication Failure Handling, requires the TSF to lock an administrator account when an excessive number of failed authentication attempts have been observed until some restorative event occurs to enable the account.

Management: FIA_AFL_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure lockout policy through unsuccessful authentication attempts.

Audit: FIA_AFL_EXT.1

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

a. Unsuccessful login attempts limit is met or exceeded.

FIA_AFL_EXT.1 Authentication Failure Handling

Hierarchical to:	No other components.
Dependencies to:	FIA_UIA_EXT.1 Administrator Identification and Authentication
	FMT_SMR.1 Security Roles

FIA_AFL_EXT.1.1

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

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

] unsuccessful authentication attempts occur related to administrators attempting to authenticate remotely using [**selection**: *username and password*, *username and PIN*].

FIA_AFL_EXT.1.2

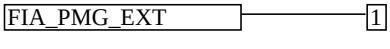
When the defined number of unsuccessful authentication attempts has been met, the TSF shall: [**selection**: *prevent the offending administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until [**assignment**: action to unlock] is taken by an administrator, prevent the offending administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until an administrator-defined time period has elapsed*].

C.2.3.2 FIA_PMG_EXT Password Management

Family Behavior

This family defines requirements for the composition of administrator passwords.

Component Leveling



FIA_PMG_EXT.1, Password Management, requires the TSF to ensure that administrator passwords meet a defined password policy.

Management: FIA_PMG_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure administrator password policy, including the ability to change default authorization factors.

Audit: FIA_PMG_EXT.1

There are no audit events foreseen.

FIA_PMG_EXT.1 Password Management

Hierarchical to: No other components.

Dependencies to: FIA_UIA_EXT.1 Administrator Identification and Authentication

FIA_PMG_EXT.1.1

The TSF shall provide the following password management capabilities for administrative passwords:

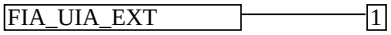
a. Passwords shall be able to be composed of any combination of upper and lower case characters, digits, and the following special characters: [**selection**: *"!", "@", "#", "$", "%", "^", "& ", "*", "(", ")", [**assignment**: other characters]*]
b. Minimum password length shall be configurable
c. Passwords of at least 15 characters in length shall be supported

C.2.3.3 FIA_UIA_EXT Administrator Identification and Authentication

Family Behavior

This family defines requirements for ensuring that access to the TSF is not granted to unauthenticated subjects.

Component Leveling



FIA_UIA_EXT.1, Administrator Identification and Authentication, requires the TSF to ensure that all subjects attempting to perform TSF-mediated actions are identified and authenticated prior to authorizing these actions to be performed.

Management: FIA_UIA_EXT.1

There are no management functions foreseen.

Audit: FIA_UIA_EXT.1

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

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

FIA_UIA_EXT.1 Administrator Identification and Authentication

Hierarchical to: No other components.

Dependencies to: FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UIA_EXT.1.1

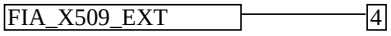
The TSF shall require administrators to be successfully identified and authenticated using one of the methods in FIA_UAU.5 before allowing any TSF-mediated management function to be performed by that administrator.

C.2.3.4 FIA_X509_EXT X.509 Certificate

Family Behavior

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

Component Leveling



FIA_X509_EXT.4, Exceptions to X.509 Certificate Revocation Checking,

Management: FIA_X509_EXT.4

There are no management functions foreseen.

Audit: FIA_X509_EXT.4

There are no audit events foreseen.

FIA_X509_EXT.4 Exceptions to X.509 Certificate Revocation Checking

Hierarchical to: No other components.

Dependencies to: FIA_X509_EXT.1 X.509 Certificate Validation

FIA_X509_EXT.4.1

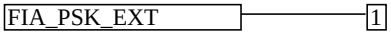
The TSF shall provide alternate functionality to standard X.509 certificate revocation checking for the following exceptions: [**selection**: *firmware checking of updates: invalidate automatic updates if the firmware certificate is compromised, [**assignment**: other exceptions and corresponding functionality]*].

C.2.3.5 FIA_PSK_EXT Pre-Shared Key Composition

Family Behavior

Components in this family describe the requirements for pre-shared keys when implementing IPsec.

Component Leveling



FIA_PSK_EXT.1, Pre-Shared Key Composition, defines the use and composition of pre-shared keys used for IPsec.

Management: FIA_PSK_EXT.1

No specific management functions are identified.

Audit: FIA_PSK_EXT.1

No specific audit functions are identified.

FIA_PSK_EXT.1 Pre-Shared Key Composition

Hierarchical to: No other components.

Dependencies to: FCS_IPSEC_EXT.1 IPsec

FIA_PSK_EXT.1.1

The TSF shall be able to use pre-shared keys for [*IKEv2*].

FIA_PSK_EXT.1.2

The TSF shall be able to accept the following as pre-shared keys: [**assignment**: *pre-shared key mechanism*] keys..

C.2.4 Class FMT: Security Management

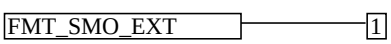
This PP defines the following extended components as part of the class originally defined by CC Part 2:

C.2.4.1 FMT_SMO_EXT Separation of Management and Operational Networks

Family Behavior

This family defines requirements for separation of management and operational networks.

Component Leveling



FMT_SMO_EXT.1, Separation of Management and Operational Networks, requires the TSF to separate its management and operational networks through a defined mechanism.

Management: FMT_SMO_EXT.1

There are no management functions foreseen.

Audit: FMT_SMO_EXT.1

There are no audit events foreseen.

FMT_SMO_EXT.1 Separation of Management and Operational Networks

Hierarchical to: No other components.

Dependencies to:

FMT_SMO_EXT.1.1

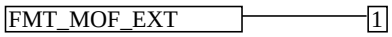
The TSF shall support the separation of management and operational network traffic through [**selection**: *separate physical networks*, *separate logical networks*, *trusted channels as defined in FTP_ITC_EXT.1*, *data encryption using an algorithm specified in [**assignment**: cryptographic requirement]*].

C.2.4.2 FMT_MOF_EXT Management of Security Functions Behavior

Family Behavior

This family defines requirements for administrative management of the TOE.

Component Leveling



FMT_MOF_EXT.1, Management of Security Functions Behavior, defines required management functions and responsibilities.

Management: FMT_MOF_EXT.1

There are no additional management functions beyond those already described in FMT_MOF_EXT.1.

Audit: FMT_MOF_EXT.1

Success or failure of management function.

FMT_MOF_EXT.1 Management of Security Functions Behavior

Hierarchical to: No other components.

Dependencies to: No other dependencies.

FMT_MOF_EXT.1.1

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

FMT_MOF_EXT.1.2

The TSF shall be capable of performing the following management functions [**assignment**: *description of management functions*].

C.2.5 Class FPT: Protection of the TSF

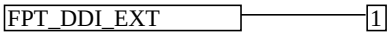
This PP defines the following extended components as part of the class originally defined by CC Part 2:

C.2.5.1 FPT_DDI_EXT Device Driver Isolation

Family Behavior

This family defines requirements for isolation of device drivers

Component Leveling



FPT_DDI_EXT.1, Device Driver Isolation, requires the TSF to isolate device drivers for physical devices from all virtual domains.

Management: FPT_DDI_EXT.1

There are no management functions foreseen.

Audit: FPT_DDI_EXT.1

There are no audit events foreseen.

FPT_DDI_EXT.1 Device Driver Isolation

Hierarchical to: No other components.

Dependencies to:

FPT_DDI_EXT.1.1

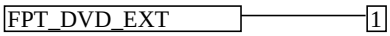
The TSF shall ensure that device drivers for physical devices are isolated from the VMM and all other domains.

C.2.5.2 FPT_DVD_EXT Non-Existence of Disconnected Virtual Devices

Family Behavior

This family defines requirements for ensuring that Guest VMs cannot access the virtual hardware interfaces disabled or disconnected virtual devices.

Component Leveling



FPT_DVD_EXT.1, Non-Existence of Disconnected Virtual Devices, requires the TSF to prevent Guest VMs from accessing virtual devices that it is not configured to have access to.

Management: FPT_DVD_EXT.1

There are no management functions foreseen.

Audit: FPT_DVD_EXT.1

There are no audit events foreseen.

FPT_DVD_EXT.1 Non-Existence of Disconnected Virtual Devices

Hierarchical to: No other components.

Dependencies to: FPT_VDP_EXT.1 Virtual Device Parameters

FPT_DVD_EXT.1.1

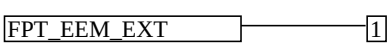
The TSF shall prevent Guest VMs from accessing virtual device interfaces that are not present in the VM's current virtual hardware configuration.

C.2.5.3 FPT_EEM_EXT Execution Environment Mitigations

Family Behavior

This family defines requirements for the TOE's compatibility with platform mechanisms that prevent vulnerabilities that allow for the execution of unauthorized code or bypass of access restrictions on memory or storage.

Component Leveling



FPT_EEM_EXT.1, Execution Environment Mitigations, requires the TSF to identify the execution environment-based protection mechanisms that it can use for self-protection.

Management: FPT_EEM_EXT.1

There are no management functions foreseen.

Audit: FPT_EEM_EXT.1

There are no audit events foreseen.

FPT_EEM_EXT.1 Execution Environment Mitigations

Hierarchical to: No other components.

Dependencies to:

FPT_EEM_EXT.1.1

The TSF shall take advantage of execution environment-based vulnerability mitigation mechanisms supported by the Platform such as: [**selection**:

- *Address space randomization*
- *Memory execution protection (e.g., DEP)*
- *Stack buffer overflow protection*
- *Heap corruption detection*
- *[**assignment**: other mechanisms]*
- *No mechanisms*

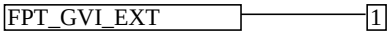
]

C.2.5.4 FPT_GVI_EXT Guest VM Integrity

Family Behavior

This family defines requirements for the TOE to assert the integrity of Guest VMs.

Component Leveling



FPT_GVI_EXT.1, Guest VM Integrity, requires the TSF to specify the mechanisms it uses to verify the integrity of Guest VMs.

Management: FPT_GVI_EXT.1

There are no management functions foreseen.

Audit: FPT_GVI_EXT.1

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

a. Actions taken due to failed integrity check.

FPT_GVI_EXT.1 Guest VM Integrity

Hierarchical to: No other components.

Dependencies to:

FPT_GVI_EXT.1.1

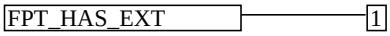
The TSF shall verify the integrity of Guest VMs through the following mechanisms: [**assignment**: *list of Guest VM integrity mechanisms*].

C.2.5.5 FPT_HAS_EXT Hardware Assists

Family Behavior

This family defines requirements for use of hardware-based virtualization assists as performance enhancements.

Component Leveling



FPT_HAS_EXT.1, Hardware Assists, requires the TSF to identify the hardware assists it uses to reduce TOE complexity.

Management: FPT_HAS_EXT.1

There are no management functions foreseen.

Audit: FPT_HAS_EXT.1

There are no audit events foreseen.

FPT_HAS_EXT.1 Hardware Assists

Hierarchical to: No other components.

Dependencies to:

FPT_HAS_EXT.1.1

The VMM shall use [**assignment**: *list of hardware-based virtualization assists*] to reduce or eliminate the need for binary translation.

FPT_HAS_EXT.1.2

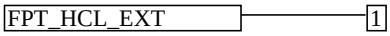
The VMM shall use [**assignment**: *list of hardware-based virtualization memory-handling assists*] to reduce or eliminate the need for shadow page tables.

C.2.5.6 FPT_HCL_EXT Hypercall Controls

Family Behavior

This family defines requirements for control of hypercall interfaces.

Component Leveling



FPT_HCL_EXT.1, Hypercall Controls, requires the TSF to implement appropriate parameter validation to protect the VMM from unauthorized access through a hypercall interface.

Management: FPT_HCL_EXT.1

There are no management functions foreseen.

Audit: FPT_HCL_EXT.1

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

a. Invalid parameter to hypercall detected.
b. Hypercall interface invoked when documented preconditions are not met.

FPT_HCL_EXT.1 Hypercall Controls

Hierarchical to: No other components.

Dependencies to: FMT_SMR.1 Security Roles

FPT_HCL_EXT.1.1

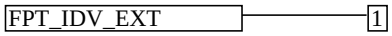
The TSF shall validate the parameters passed to hypercall interfaces prior to execution of the VMM functionality exposed by each interface.

C.2.5.7 FPT_IDV_EXT Software Identification and Versions

Family Behavior

This family defines requirements for the use of SWID tags to identify the TOE.

Component Leveling



FPT_IDV_EXT.1, Software Identification and Versions, requires the TSF to identify itself using SWID tags.

Management: FPT_IDV_EXT.1

There are no management functions foreseen.

Audit: FPT_IDV_EXT.1

There are no audit events foreseen.

FPT_IDV_EXT.1 Software Identification and Versions

Hierarchical to: No other components.

Dependencies to:

FPT_IDV_EXT.1.1

The TSF shall include software identification (SWID) tags that contain a SoftwareIdentity element and an Entity element as defined in ISO/IEC 19770-2:2009.

FPT_IDV_EXT.1.2

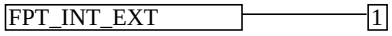
The TSF shall store SWIDs in a.swidtag file as defined in ISO/IEC 19770-2:2009.

C.2.5.8 FPT_INT_EXT Support for Introspection

Family Behavior

This family defines requirements for supporting VM introspection.

Component Leveling



FPT_INT_EXT.1, Support for Introspection, requires the TSF to support introspection.

Management: FPT_INT_EXT.1

There are no management functions foreseen.

Audit: FPT_INT_EXT.1

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

a. Introspection initiated/enabled.

FPT_INT_EXT.1 Support for Introspection

Hierarchical to: No other components.

Dependencies to:

FPT_INT_EXT.1.1

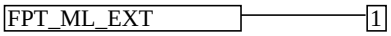
The TSF shall support a mechanism for permitting the VMM or privileged VMs to access the internals of another VM for purposes of introspection.

C.2.5.9 FPT_ML_EXT Measured Launch of Platform and VMM

Family Behavior

This family defines requirements for measured launch.

Component Leveling



FPT_ML_EXT.1, Measured Launch of Platform and VMM, requires the TSF to support a measured launch of itself.

Management: FPT_ML_EXT.1

There are no management functions foreseen.

Audit: FPT_ML_EXT.1

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

a. Integrity measurements collected.

FPT_ML_EXT.1 Measured Launch of Platform and VMM

Hierarchical to: No other components.

Dependencies to:

FPT_ML_EXT.1.1

The TSF shall support a measured launch of the virtualization system. Measured components of the VS shall include the static executable image of the hypervisor and: [**selection**:

- *Static executable images of the Management Subsystem*
- *[**assignment**: list of (static images of) Service VMs]*
- *[**assignment**: list of configuration files]*
- *no other components*

]

FPT_ML_EXT.1.2

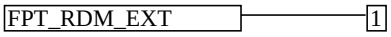
The TSF shall make the measurements selected in FPT_ML_EXT.1.1 available to the Management Subsystem.

C.2.5.10 FPT_RDM_EXT Removable Devices and Media

Family Behavior

This family defines requirements for enforcement of domain isolation when removable devices can be connected to a domain.

Component Leveling



FPT_RDM_EXT.1, Removable Devices and Media, requires the TSF to ensure that VMs are not inadvertently given access to information in different domains because removable media is simultaneously accessible from separate domains.

Management: FPT_RDM_EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to configure removable media policy.
- Ability to connect/disconnect removable devices to/from a VM.

Audit: FPT_RDM_EXT.1

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

a. Connection/disconnection of removable media or device to/from a VM.
b. Ejection/insertion of removable media or device from/to an already connected VM.

FPT_RDM_EXT.1 Removable Devices and Media

Hierarchical to: No other components.

Dependencies to: FDP_VMS_EXT.1 VM Separation

FPT_RDM_EXT.1.1

The TSF shall implement controls for handling the transfer of virtual and physical removable media and virtual and physical removable media devices between information domains.

FPT_RDM_EXT.1.2

The TSF shall enforce the following rules when [**assignment**: *virtual or physical removable media and virtual or physical removable media devices*] are switched between information domains: [**selection**:

- *the administrator has granted explicit access for the media or device to be connected to the receiving domain*
- *the media in a device that is being transferred is ejected prior to the receiving domain being allowed access to the device*
- *the user of the receiving domain expressly authorizes the connection*
- *the device or media that is being transferred is prevented from being accessed by the receiving domain*

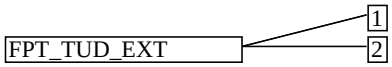
]

C.2.5.11 FPT_TUD_EXT Trusted Updates

Family Behavior

This family defines requirements for ensuring that updates to the TOE software and firmware are genuine.

Component Leveling



FPT_TUD_EXT.1, Trusted Updates to the Virtualization System, requires the TSF to define the mechanism for applying and verifying TOE updates.

FPT_TUD_EXT.2, Trusted Update Based on Certificates, requires the TSF to validate updates using a code signing certificate.

Management: FPT_TUD_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to update the virtualization system.

Audit: FPT_TUD_EXT.1

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

a. Initiation of update.
b. Failure of signature verification.

FPT_TUD_EXT.1 Trusted Updates to the Virtualization System

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FPT_TUD_EXT.1.1

The TSF shall provide administrators the ability to query the currently executed version of the TOE firmware and software as well as the most recently installed version of the TOE firmware and software.

FPT_TUD_EXT.1.2

The TSF shall provide administrators the ability to manually initiate updates to TOE firmware and software and [**selection**: *automatic updates, no other update mechanism*].

FPT_TUD_EXT.1.3

The TSF shall provide means to authenticate firmware and software updates to the TOE using a [**assignment**: *integrity action*] prior to installing those updates.

Management: FPT_TUD_EXT.2

There are no management functions foreseen.

Audit: FPT_TUD_EXT.2

There are no audit events foreseen.

FPT_TUD_EXT.2 Trusted Update Based on Certificates

Hierarchical to:	No other components.
Dependencies to:	FPT_TUD_EXT.1 Trusted Updates to the Virtualization System FIA_X509_EXT.1 X.509 Validation FIA_X509_EXT.2 X.509 Authentication

FPT_TUD_EXT.2.1

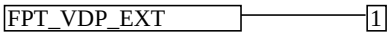
The TSF shall not install an update if the code signing certificate is deemed invalid.

C.2.5.12 FPT_VDP_EXT Virtual Device Parameters

Family Behavior

This family defines requirements for processing data transmitted to the TOE from a Guest VM.

Component Leveling



FPT_VDP_EXT.1, Virtual Device Parameters, requires the TSF to interface with Guest VMs through virtual hardware abstractions so that any data transmitted to the TOE from a Guest VM can be validated as well-formed.

Management: FPT_VDP_EXT.1

There are no management functions foreseen.

Audit: FPT_VDP_EXT.1

There are no audit events foreseen.

FPT_VDP_EXT.1 Virtual Device Parameters

Hierarchical to:	No other components.
Dependencies to:	FPT_VIV_EXT.1 VMM Isolation from VMs

FPT_VDP_EXT.1.1

The TSF shall provide interfaces for virtual devices implemented by the VMM as part of the virtual hardware abstraction.

FPT_VDP_EXT.1.2

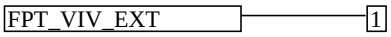
The TSF shall validate the parameters passed to the virtual device interface prior to execution of the VMM functionality exposed by those interfaces.

C.2.5.13 FPT_VIV_EXT VMM Isolation from VMs

Family Behavior

This family defines requirements for ensuring the TOE is logically isolated from its Guest VMs

Component Leveling



FPT_VIV_EXT.1, VMM Isolation from VMs, requires the TSF to ensure that there is no mechanism by which a Guest VM can interface with the TOE, other VMs, or the hardware platform without authorization.

Management: FPT_VIV_EXT.1

There are no management functions foreseen.

Audit: FPT_VIV_EXT.1

There are no audit events foreseen.

FPT_VIV_EXT.1 VMM Isolation from VMs

Hierarchical to: No other components.

Dependencies to: FDP_PPR_EXT.1 Physical Platform Resource Controls

FDP_VMS_EXT.1 VM Separation

FPT_VIV_EXT.1.1

The TSF must ensure that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the platform.

FPT_VIV_EXT.1.2

The TSF must ensure that a Guest VM is unable to invoke platform code that runs at a privilege level equal to or exceeding that of the VMM without involvement of the VMM.

C.2.6 Class FTP: Trusted Path/Channels

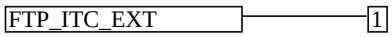
This PP defines the following extended components as part of the class originally defined by CC Part 2:

C.2.6.1 FTP_ITC_EXT Trusted Channel Communications

Family Behavior

This family defines requirements for protection of data in transit between the TOE and its operational environment.

Component Leveling



FTP_ITC_EXT.1, Trusted Channel Communications, requires the TSF to implement one or more cryptographic protocols to secure connectivity between the TSF and various external entities.

Management: FTP_ITC_EXT.1

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, PP-Module, functional package or ST:

a. Initiation of the trusted channel.
b. Termination of the trusted channel.
c. Failures of the trusted path functions.

FTP_ITC_EXT.1 Trusted Channel Communications

Hierarchical to: No other components.

Dependencies to: FAU_STG.1 External Audit Storage

FTP_ITC_EXT.1.1

The TSF shall use [**assignment**: *transport mechanism*] and [**assignment**: *authentication mechanism*] to provide a trusted communication channel between itself and

- audit servers (as required by FAU_STG.1) and

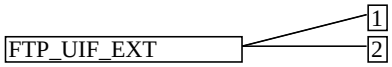
[**assignment**: *remote entities*] that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from disclosure and detection of modification of the communicated data.

C.2.6.2 FTP_UIF_EXT User Interface

Family Behavior

This family defines requirements for unambiguously identifying the specific Guest VM that a TOE user is interacting with at any given point in time.

Component Leveling



FTP_UIF_EXT.1, User Interface: I/O Focus, requires the TSF to unambiguously identify the Guest VM that has the current input focus for input peripherals.

FTP_UIF_EXT.2, User Interface: Identification of VM, requires the TOE to perform power on self-tests to verify its functionality and the integrity of its stored executable code.

Management: FTP_UIF_EXT.1

There are no management functions foreseen.

Audit: FTP_UIF_EXT.1

There are no audit events foreseen.

FTP_UIF_EXT.1 User Interface: I/O Focus

Hierarchical to: No other components.

Dependencies to:

FTP_UIF_EXT.1.1

The TSF shall indicate to users which VM, if any, has the current input focus.

Management: FTP_UIF_EXT.2

There are no management functions foreseen.

Audit: FTP_UIF_EXT.2

There are no audit events foreseen.

FTP_UIF_EXT.2 User Interface: Identification of VM

Hierarchical to: No other components.

Dependencies to:

FTP_UIF_EXT.2.1

The TSF shall support the unique identification of a VM's output display to users.

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_UID.1 - Timing of Identification	FMT_SMR.2 has a dependency on FIA_UID.1. The extended SFR FIA_UID_EXT.1 expresses this dependency by also requiring user identification for use of the TOE.
FMT_SMR.2 - Restrictions on Security Roles	Several SFRs have dependencies on FMT_SMR.2. The SFR FMT_MOF_EXT.1 implicitly satisfies this SFR because it defines the management roles that can be used to perform each function, which defines the existence of management roles and any restrictions on those roles (e.g., there are some management functions one must perform while another must not, while there are others that one must perform and the other may perform).
FPT_STM.1 - Reliable Time Stamps	FAU_GEN.1 has a dependency on FPT_STM.1. While not explicitly stated in the PP, it is assumed that this will be provided by the underlying hardware platform on which the TOE is installed. This is because the TOE is installed as a software or firmware product that runs on general-purpose computing hardware so a hardware clock is assumed to be available.
FPT_STM.1 - Reliable Time Stamps	FIA_X509_EXT.1 in the Functional Package for X.509, version 1.0 has a dependency on FPT_STM.1. While not explicitly stated in the PP, it is assumed that this will be provided by the underlying hardware platform on which the TOE is installed. This is because the TOE is installed as a software or firmware product that runs on general-purpose computing hardware so a hardware clock is assumed to be available.

Appendix E - Entropy Documentation And Assessment

The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide 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 shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the random comes from, where it is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

If implemented, the design description shall include a description of how third-party applications can add entropy to the DRBG. A description of any DRBG state saving between power-off and power-on shall 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 exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

The entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

E.3 Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

E.4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

Appendix F - Equivalency Guidelines

F.1 Introduction

The purpose of equivalence in PP-based evaluations is to find a balance between evaluation rigor and commercial practicability--to ensure that evaluations meet customer expectations while recognizing that there is little to be gained from requiring that every variation in a product or platform be fully tested. If a product is found to be compliant with a PP on one platform, then all equivalent products on equivalent platforms are also considered to be compliant with the PP.

A Vendor can make a claim of equivalence if the Vendor believes that a particular instance of their Product implements PP-specified security functionality in a way equivalent to the implementation of the same functionality on another instance of their Product on which the functionality was tested. The Product instances can differ in version number or feature level (model), or the instances may run on different platforms. Equivalency can be used to reduce the testing required across claimed evaluated configurations. It can also be used during Assurance Maintenance to reduce testing needed to add more evaluated configurations to a certification.

These equivalency guidelines do not replace Assurance Maintenance requirements or NIAP Policy #5 requirements for CAVP certificates. Nor may equivalency be used to leverage evaluations with expired certifications.

This document provides guidance for determining whether Products and Platforms are equivalent for purposes of evaluation against the Protection Profile for Virtualization (VPP).

Equivalence has two aspects:

1. ***Product Equivalence:*** Products may be considered equivalent if there are no differences between Product Models and Product Versions with respect to PP-specified security functionality.
2. ***Platform Equivalence:*** Platforms may be considered equivalent if there are no significant differences in the services they provide to the Product--or in the way the platforms provide those services--with respect to PP-specified security functionality.

The equivalency determination is made in accordance with these guidelines by the Validator and Scheme using information provided by the Evaluator/Vendor.

F.2 Approach to Equivalency Analysis

There are two scenarios for performing equivalency analysis. One is when a product has been certified and the vendor wants to show that a later product should be considered certified due to equivalence with the earlier product. The other is when multiple product variants are going though evaluation together and the vendor would like to reduce the amount of testing that must be done. The basic rules for determining equivalence are the same in both cases. But there is one additional consideration that applies to equivalence with previously certified products. That is, the product with which equivalence is being claimed must have a valid certification in accordance with scheme rules and the Assurance Maintenance process must be followed. If a product's certification has expired, then equivalence cannot be claimed with that product.

When performing equivalency analysis, the Evaluator/Vendor should first use the factors and guidelines for Product Model equivalence to determine the set of Product Models to be evaluated. In general, Product Models that do not differ in PP-specified security functionality are considered equivalent for purposes of evaluation against the VPP.

If multiple revision levels of Product Models are to be evaluated--or to determine whether a revision of an evaluated product needs re-evaluation--the Evaluator/Vendor and Validator should use the factors and guidelines for Product Version equivalence to determine whether Product Versions are equivalent.

Having determined the set of Product Models and Versions to be evaluated, the next step is to determine the set of Platforms that the Products must be tested on.

Each non-equivalent Product for which compliance is claimed must be fully tested on each non-equivalent platform for which compliance is claimed. For non-equivalent Products on equivalent platforms, only the differences that affect PP-specified security functionality must be tested for each product.

If the set of equivalent Products includes only bare-metal installations, then the equivalency analysis is complete. But if any members of the set include hosted installations or installations that integrate with an existing host operating system or control domain, then software platform equivalence must be taken into consideration. The Evaluator/Vendor and Validator should use the factors and guidance for software platform equivalence to determine whether different models or versions of host or control domain operating systems require separate testing.

"Differences in PP-Specified Security Functionality" Defined
If PP-specified security functionality is implemented by the TOE, then differences in the actual implementation between versions or product models break equivalence for that feature. Likewise, if the TOE implements the functionality in one version or model and the functionality is implemented by the platform in another version or model, then equivalence is broken. If the functionality is implemented by the platform

in multiple models or versions on equivalent platforms, then the functionality is considered different if the product invokes the platform differently to perform the function.

F.3 Specific Guidance for Determining Product Model Equivalence

Product Model equivalence attempts to determine whether different feature levels of the same product across a product line are equivalent for purposes of PP testing. For example, if a product has a “basic” edition and an “enterprise” edition, is it necessary to test both models? Or does testing one model provide sufficient confidence that both models are compliant?

Table 23, below, lists the factors for determining Product Model equivalence.

Table 23: Factors for Determining Product Model Equivalence

Factor	Same/Different	Guidance
Target Platform	Different	Product Models that virtualize different instruction sets (e.g., x86, ARM, POWER, SPARC, MIPS) are not equivalent.
Installation Types	Different	If a Product can be installed either on bare metal or onto an operating system and the vendor wants to claim that both installation types constitute a single Model, then see the guidance for “PP-Specified Functionality,” below.
Software Platform	Different	Product Models that run on substantially different software environments, such as different host operating systems, are not equivalent. Models that install on different versions of the same software environment may be equivalent depending on the below factors.
PP-Specified Functionality	Same	If the differences between Models affect only non-PP-specified functionality, then the Models are equivalent.
	Different	If PP-specified security functionality is affected by the differences between Models, then the Models are not equivalent and must be tested separately. It is necessary to test only the functionality affected by the software differences. If only differences are tested, then the differences must be enumerated, and for each difference the Vendor must provide an explanation of why each difference does or does not affect PP-specified functionality. If the Product Models are fully tested separately, then there is no need to document the differences.

F.4 Specific Guidance for Determining Product Version Equivalence

In cases of version equivalence, differences are expressed in terms of changes implemented in revisions of an evaluated Product. In general, versions are equivalent if the changes have no effect on any security-relevant claims about the TOE or evaluation evidence. Non-security-relevant changes to TOE functionality or the addition of non-security-relevant functionality does not affect equivalence.

Table 24: Factors for Determining Product Version Equivalence

Factor	Same/Different	Guidance
Product Models	Different	Versions of different Product Models are not equivalent unless the Models are equivalent as defined in Section 3.
PP-Specified Functionality	Same	If the differences affect only non-PP-specified functionality, then the Versions are equivalent.
	Different	If PP-specified security functionality is affected by the differences, then the Versions are considered to be not equivalent and must be tested separately. It is necessary only to test the functionality affected by the changes. If only the differences are tested, then for each difference the Vendor must provide an explanation of why the difference does or does not affect PP-specified functionality. If the Product Versions are fully tested separately, then there is no need to document the differences.

F.5 Specific Guidance for Determining Platform Equivalence

Platform equivalence is used to determine the platforms that a product must be tested on. These guidelines are divided into sections for determining hardware equivalence and software (host OS/control domain) equivalence. If the Product is installed onto bare metal, then only hardware equivalence is relevant. If the Product is installed onto an OS—or is integrated into an OS—then both hardware and software equivalence are required. Likewise, if the Product can be installed either on bare metal or on an operating system, both hardware and software equivalence are relevant.

F.5.1 Hardware Platform Equivalence

If a Virtualization Solution runs directly on hardware without an operating system, then platform equivalence is based primarily on processor architecture and instruction sets.

Platforms with different processor architectures and instruction sets are not equivalent. This is probably not an issue because there is likely to be a different product model for different hardware environments.

Equivalency analysis becomes important when comparing platforms with the same processor architecture. Processors with the same architecture that have instruction sets that are subsets or supersets of each other are not disqualified from being equivalent for purposes of a VPP evaluation. If the VS takes the same code paths when executing PP-specified security functionality on different processors of the same family, then the processors can be considered equivalent with respect to that application.

For example, if a VS follows one code path on platforms that support the AES-NI instruction and another on platforms that do not, then those two platforms are not equivalent with respect to that VS functionality. But if the VS follows the same code path whether or not the platform supports AES-NI, then the platforms are equivalent with respect to that functionality.

The platforms are equivalent with respect to the VS if the platforms are equivalent with respect to all PP-specified security functionality.

Table 25: Factors for Determining Hardware Platform Equivalence

Factor	Same/Different/None	Guidance
Platform Architectures	Different	Hardware platforms that implement different processor architectures and instruction sets are not equivalent.
PP-Specified Functionality	Same	For platforms with the same processor architecture, the platforms are equivalent with respect to the application if execution of all PP-specified security functionality follows the same code path on both platforms.

F.5.2 Software Platform Equivalence

If the Product installs onto or integrates with an operating system that is not installed with the product--and thus is not part of the TOE--then the Product must be tested on all non-equivalent Software Platforms.

The guidance for Product Model (Section 3) specifies that Products intended for use on substantially different operating systems (e.g., Windows vs. Linux vs. SunOS) are different Models. Therefore, platforms running substantially different operating systems are not equivalent. Likewise, operating systems with different major version numbers are not equivalent for purposes of this PP.

As a result, Software Platform equivalence is largely concerned with revisions and variations of operating systems that are substantially the same (e.g., different versions and revision levels of Windows or Linux).

Table 26: Factors for Determining Software Platform Equivalence

Factor	Same/Different/None	Guidance
Platform Type/Vendor	Different	Operating systems that are substantially different or come from different vendors are not equivalent.
Platform Versions	Different	Operating systems are not equivalent if they have different major version numbers.
PP-Specified Functionality	Same	If the differences between software platform models or versions affect only non-PP-specified functionality, then the software platforms are equivalent.
	Different	If PP-specified security functionality is affected by the differences between software platform versions or models, then the software platforms are not considered equivalent and must be tested separately. It is necessary only to test the functionality affected by the changes. If only the differences are tested, then for each difference the Vendor must provide an explanation of why the difference does or does not affect PP-specified functionality. If the Products are fully tested on each platform, then there is no need to document the differences.

F.6 Level of Specificity for Tested and Claimed Equivalent Configurations

In order to make equivalency determinations, the vendor and evaluator must agree on the equivalency claims. They must then provide the scheme with sufficient information about the TOE instances and platforms that were evaluated, and the TOE instances and platforms that are claimed to be equivalent.

The ST must describe all configurations evaluated down to processor manufacturer, model number, and microarchitecture version.

The information regarding claimed equivalent configurations depends on the platform that the VS was developed for and runs on.

Bare-Metal VS

For VSes that run without an operating system on bare-metal or virtual bare-metal, the claimed configuration must describe the platform down to the specific processor manufacturer, model number, and microarchitecture version. The Vendor must describe the differences in the TOE with respect to PP-specified security functionality and how the TOE operates differently to leverage platform differences (e.g., instruction set extensions) in the tested configuration versus the claimed equivalent configuration.

VS with OS Support

For VSes that run on an OS host or with the assistance of an OS, then the claimed configuration must describe the OS down to its specific model and version number. The Vendor must describe the differences in the TOE with respect to PP-specified security functionality and how the TOE functions differently to leverage platform differences in the tested configuration versus the claimed equivalent configuration.

Appendix G - Acronyms

Table 27: Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
Base-PP	Base Protection Profile
CC	Common Criteria
CEM	Common Evaluation Methodology
CNSA	Commercial National Security Algorithm Suite
cPP	Collaborative Protection Profile
CPU	Central Processing Unit
DEP	Data Execution Prevention
DKM	Derived Keying Material
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptography
EP	Extended Package
FFC	Finite-Field Cryptography
FIPS	Federal Information Processing Standard
FP	Functional Package
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
ITSEF	Information Technology Security Evaluation Facility
KDF	Key Derivation Function
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OE	Operational Environment
OS	Operating System
PKV	Public Key Verification
PP	Protection Profile
PP-Configuration	Protection Profile Configuration
PP-Module	Protection Profile Module
RSA	Rivest, Shamir, Adleman

SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SP	Special Publication
SPD	Security Policy Database
SSP	System Security Policy
ST	Security Target
SWID	Software Identification
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification
VM	Virtual Machine
VMM	Virtual Machine Manager
VS	Virtualization System

Appendix H - Bibliography

Table 28: 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.
[Errata]	Errata and Interpretation for CC:2022 (Release 1) and CEM:2022 (Release 1), Version 1.2
[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.