# Protection Profile for Server Virtualization


*29 October 2014*

Version 1.0

# 0 Preface

## 0.1 Objectives of Document

This document presents the Common Criteria (CC) Protection Profile (PP) to express the fundamental security and evaluation requirements for Server Virtualization.

## 0.2 Scope of Document

The scope of the Protection Profile within the development and evaluation process is described in the Common Criteria for Information Technology Security Evaluation [CC]. In particular, a PP defines the IT security requirements of a generic type of TOE and specifies the functional and assurance security measures to be offered by that TOE to meet stated requirements [CC1, Section C.1].

## 0.3 Intended Readership

The target audiences of this PP are Server Virtualization developers, CC consumers, evaluators and schemes.

## 0.4 Related Documents

**Common Criteria**[1]

[CC1] Common Criteria for Information Technology Security Evaluation,
Part 1: Introduction and General Model,
CCMB-2012-09-001, Version 3.1 Revision 4, September 2012.

[CC2] Common Criteria for Information Technology Security Evaluation,
Part 2: Security Functional Components,
CCMB-2012-09-002, Version 3.1 Revision 4, September 2012.

[CC3] Common Criteria for Information Technology Security Evaluation,
Part 3: Security Assurance Components,
CCMB-2012-09-003, Version 3.1 Revision 4, September 2012.

[CEM] Common Methodology for Information Technology Security Evaluation,
Evaluation Methodology,
CCMB-2012-09-004, Version 3.1, Revision 4, September 2012.

[1]For details see http://www.commoncriteriaportal.org/

## 0.5 Revision History

| Version | Date | Description |
|---|---|---|
| 1.0 | 29 October 2014 | Initial Release |

# Table of Contents

# List of Figures



# List of Tables

# 1 PP Introduction

## 1.1 PP Reference Identification

PP Reference: Protection Profile for Server Virtualization

PP Version: 1.0

PP Date: 29 October 2014

1 This Protection Profile (PP) describes the security requirements for Server Virtualization and provides a minimal baseline set of requirements targeted at mitigating well-defined threats.

2 This PP is based on requirements identified in the Distributed Management Task Force (DMTF) white paper dated September 5, 2012. The DMTF document defines design goals and requirements for the secure implementation of server virtualization solutions. This protection profile has distilled over 100 requirements from the DMTF paper down to a core of approximately 50 requirements considered by the authors to be fundamental to the security of Virtualization Systems.

## 1.2 Glossary

3 **Administrator/Virtualization System Administrator.** See **User**.

4 **Auditor.** See **User**.

5 **Domain/Information 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 Virtualization System itself can be considered an Information Domain, as can its Management Subsystem.

6 **Guest Operating System (Guest OS).** An operating system that runs within a Guest or Helper VM.

7 **Guest Network.** See **Operational Network**.

8 **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.

9 **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.

10 **Host Operating System (Host OS).** An operating system onto which a Virtualization System is installed. Relative to the VS, the Host OS is part of the Platform.

11 **Hypervisor.** The Hypervisor is part of the VMM. It is the software executive of the physical platform of a Virtualization System. 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.

12 **Hypercall.** An API function that allows VM-aware software running within a VM to invoke Hypervisor or VMM functionality.

13 **Introspection.** A capability that allows a specially designated and privileged domain to have visibility into another domain for purposes of anomaly detection or monitoring.

14 **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 for purposes of VM transfer or coordinate, and backend storage networks are considered management networks.

15 **Management Subsystem.** Components of the VS that allow VS Administrators to configure and manage the VMM, as well as configure Guest or Helper VMs. VMM management functions include VM configuration, virtualized network configuration, and allocation of physical resources.

16 **Operational Network.** An Operational Network is a network, which may have both physical and virtualized components, used to connect Guest and Helper 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."

17 **Physical Platform.** The hardware environment on which a VS executes. Physical platform resources include processors, memory, devices, and associated firmware.

18 **Platform.** The hardware, firmware, and software environment into which a VS is installed and executes.

19 **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 and Helper VMs (defined above). 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 a physical devices, and name-service VMs that help establish communication paths between VMs.

20 **System Security Policy (SSP).** The overall policy enforced by the Virtualization System defining constraints on the behavior of VMs and users.

21 **User.** For purposes of this document, there are three user roles in Virtualization Systems defined by their privileges:

- **Unprivileged Users.** Generally referred to as "Users." Users operate Guest VMs. For some VS instantiations, Users may function only within Guest VMs (e.g. remote thin client environments). For other instantiations, policy may permit Users to, for example, start and stop Guest VMs, connect and disconnect Guest VMs to physical devices, and

log in to multiple Guests at the same time. Users are not permitted to configure VMs, virtual networking, or policy.

- **VS Administrators.** Generally referred to as "Administrators." Administrators perform management functions on the VS, including creation and configuration of Guest VMs, assignment of Users to VMs, definition of Domains, creation and management of virtualized network infrastructure, and maintenance of the VS. VS Administration does not include administration of software running within Guest VMs, such as the Guest OS.
- **Auditors.** An Auditor is responsible for managing the audit capabilities of the TOE. An Auditor may also be an Administrator. It is not a threshold requirement that the TOE be capable of supporting an Auditor role that is separate from that of an Administrator.

22 **Virtual Machine (VM).** A Virtual Machine is a virtualized hardware environment in which an operating system may execute.

23 **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.

24 **Virtualization System (VS).** A software product that enables multiple independent computing systems to execute on the same physical hardware platform without interference from one other. For purposes of this document, the VS consists of a Virtual Machine Manager (VMM), Virtual Machine (VM) abstractions, a management subsystem, and other components.

25

## 1.3 Compliant Targets of Evaluation

### 1.3.1 Components of a Virtualization System

26 Server Virtualization in the context of this PP relates 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 the appearance of isolation and 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.

27 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 other. For purposes of this document, the VS consist of a Virtual Machine Manager (VMM), Virtual Machine (VM) abstractions, and other components.

28 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.

29 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.

30 A Service VM is a VM whose purpose is to support the Hypervisor in providing the resources or services necessary to support Guest and Helper VMs (defined below). 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 a physical devices, and name-service VMs that help establish communication paths between VMs.

31 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 purposes of this document, Helper VMs are subject to all requirements that apply to Guest VMs, unless specifically stated otherwise.

### 1.3.2 TOE Definition

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. The Management Subsystem is part of the TOE, but is not part of the VMM.

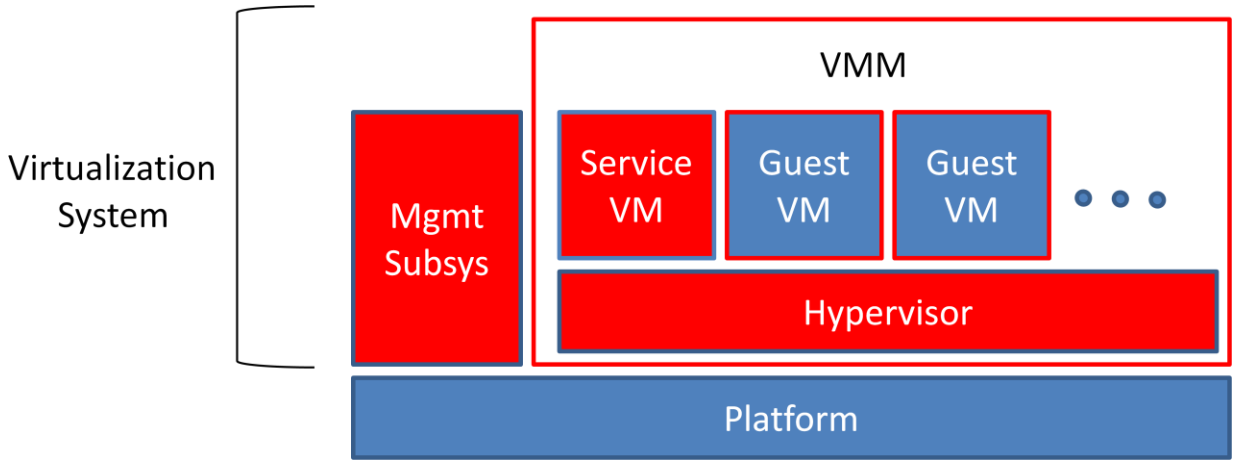


**Figure 1. Virtualization System and Platform**

32 For purposes of this Protection Profile, the VS is the TOE, subject to some caveats. The Platform onto which the VS is installed is not part of the TOE, to include hardware, platform firmware, and Host Operating System. 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.

33 In general, for Server Virtualization products that are installed onto "bare metal," the entire set of installed components constitute the TOE, and the hardware constitute 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.

### 1.3.3 Requirements Met by the Platform

34 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 assurance 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 the assurance level does not change between instances of

equivalent products on equivalent platforms. The guidelines will also ensure that the appropriate testing is done when the distinction is significant.

### 1.3.4 Scope of Certification

35 Successful evaluation of a VS 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.

### 1.3.5 Vendor Attestation

36 This PP includes three SFRs that include elements that are met via vendor attestation. Attestation assurance activities are reserved for SFRs that define a property that is critical to the system's security functionality, but that is impossible or impractical to test in a repeatable and consistent manner. These Attestation assurance activities require the vendor to make an assertion in the ST that their product meets the specified SFR—no further testing or assessment of the product regarding these SFRs or elements is performed by the CCTL. The CCTL will simply verify that the ST includes a pre-determined Attestation statement. By including this statement in the ST, the vendor is accepting responsibility for the assurance of their product in these particular areas. If at any time evidence is produced that indicates that these statements are false (and the product does not meet the specified security functionality), the CC certificate may be revoked. See FDP_VMS_EXT.1, FDP_VNC_EXT.1, and FPT_VIV_EXT.1 for the associated Attestation statements.

### 1.3.6 Product and Platform Equivalence

37 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 the Profile's equivalency guidelines [to be published].

# 2 CC Conformance

As defined by the references [CC1], [CC2] and [CC3], this PP:

- conforms to the requirements of Common Criteria v3.1, Revision 4
- is Part 2 extended, Part 3 conformant
- does not claim conformance to any other PP.

In order to be conformant to this PP, a TOE must demonstrate Exact Compliance. Exact Compliance, as a subset of Strict Compliance as defined by the CC, is defined as the ST containing all of the requirements in section 5 of the this PP, and potentially requirements from Annex B, C, and D of this PP. While iteration is allowed, no additional requirements (from the CC parts 2 or 3) are allowed to be included in the ST. Further, no requirements in section 5 of this PP are allowed to be omitted.

# 3 SECURITY PROBLEM DESCRIPTION

38 The security problem to be addressed by compliant TOEs is described by threats and policies that are common to a Server Virtualization System in general, rather than those that might be targeted at a specific Virtualization System function or at a specific type of Virtualization System. Annex A presents the Security Problem Description (SPD) in a more "traditional" form. The following sections detail the problems that compliant TOEs will address; references to the "traditional" statements in Annex A are included.

39 Assumptions are described in Table 4 (Annex A).

## 3.1 Cross-Domain Data Leakage

40 It is a fundamental property of VMs that the domains encapsulated by different VMs remain separate unless data sharing is permitted by policy. For this reason, all VSs shall support a policy that prohibits information transfer between VMs.

41 It shall be possible to configure VMs such that data cannot be moved between domains from VM to VM, or through virtual or physical network components under the control of the VS. When VMs are configured as such, it shall not be possible for data to leak between domains, neither by the express efforts of software or users of a VM, nor because of vulnerabilities or errors in the implementation of the VMM or other VS components.

42 If it is possible for data to leak between domains when prohibited by policy, then an adversary on one domain or network can obtain data from another domain. Such cross-domain data leakage can, for example, cause classified information, corporate proprietary information, or medical data to be made accessible to unauthorized entities.

43 [T.DATA_LEAKAGE]

## 3.2 Unauthorized Update

44 It is common for attackers to target unpatched versions of software containing known flaws. This means it is extremely important to update Virtualization System software as soon as possible when updates are available. But the source of the updates and the updates themselves must be trusted. If an attacker can write their own update containing malicious code they can take control of the VS.

45 Methods of countering this threat typically involve digitally signing updates and verifying the signature before installation of the update, or computing a hash on updates and verifying the result against a value provided by the vendor.

46 [T.UNAUTHORIZED_UPDATE]

## 3.3 Unauthorized Modification

47 System integrity is a core security objective for Virtualization Systems. To achieve system integrity the integrity of each VMM component must be established and maintained. Malware running on the platform must not be able to undetectably modify VS components while the system is running or at rest. Likewise, malicious code running within a virtual machine must not be able to modify VS components. Integrity measurements and secure launch of the VS are ways of ensuring integrity.

48 [T.UNAUTHORIZED_MODIFICATON]

## 3.4 User Error

49 If a VS is capable of simultaneously displaying VMs of different domains to the same user at the same time, there is always the chance that the user will become confused and unintentionally leak information between domains. This is especially likely if VMs belonging to different domains are indistinguishable. The VS must take measures to minimize the likelihood of such confusion.

50 [T.USER_ERROR]

## 3.5 Vulnerability In Third-Party Software

51 In some VS implementations, critical functions are by necessity performed by software not produced by the Virtualization Vendor. Such software includes Host OSs and physical device drivers. Vulnerabilities in this software can be exploited by an adversary and result in VMM compromise. Where possible, the VS should mitigate the results of potential vulnerabilities or malicious content in third-party code.

[T.3P_SOFTWARE]

## 3.6 VMM Compromise

52 The VS is designed to provide the appearance of exclusivity to the VMs and is designed to separate or isolate their functions except where specifically shared. Failure of security mechanisms could lead to unauthorized intrusion into or modification of the VMM, or bypass of the VMM altogether. This must be prevented to avoid compromising the VS.

53 [T.VMM_COMPROMISE]

## 3.7 Platform Compromise

54 The VS must be capable of protecting the platform from threats that originate within VMs and operational networks connected to the VS. The hosting of untrusted—even malicious—domains by the VS cannot be permitted to compromise the security and integrity of the platform on which the VS executes. If an attacker can access the underlying platform in a manner not controlled by the VMM, the attacker might be able to modify system firmware or software—compromising both the VS and the underlying platform.

55 [T.PLATFORM_COMPROMISE]

## 3.8 Unauthorized Access to Management Functions

56 Server VSs generally include a comprehensive management infrastructure. Functions performed by the management layer include VM configuration, virtualized network configuration, allocation of physical resources, and reporting. Only certain authorized system users (administrators) are allowed to exercise management functions.

57 VS servers communicate with other servers and clients, as well as administrators, over communication networks. The endpoints can be both geographically and logically separated from the server, and pass through a variety of other systems which may be under the control of an adversary, and offer the opportunity for communications with the server to be compromised. An adversary with access to an open management network could inject commands into the management infrastructure. This would provide an adversary with administrator privilege on the server platform, and administrative control

over the VMs and virtual network connections. The adversary could also gain access to the management network by hijacking the management network channel.

58 [T.UNAUTHORIZED_ACCESS]

## 3.9 Weak Cryptography Due to Insufficient Entropy

59 To the extent that VMs appear isolated within the VS, a threat of weak cryptography may arise if the VMM does not provide sufficient entropy to support security-related features that depend on entropy to implement cryptographic algorithms. For example, a random number generator keeps an estimate of the number of bits of noise in the entropy pool. From this entropy pool random numbers are created. Good random numbers are essential to implementing strong cryptography. Cryptography implemented using poor random numbers can be defeated by a sophisticated adversary.

60 [T.WEAK_CRYPTO]

## 3.10 Unmanageable Enterprise Network

61 VSs can host VMs that are part of one or more larger physical networks. The VS itself is infrastructure network entity and must be managed as such. Unmanageable networks make it difficult to keep software and firmware of servers, clients, and network devices updated and patched. Known vulnerabilities in unpatched software often can easily be exploited by adversaries to compromise the VS or platform.

62 [T.UNMANAGEABLE_NW]

# 4 SECURITY OBJECTIVES

64 Compliant TOEs provide security functionality to address threats and to implement policies that are imposed by law or regulation. The following sections provide a description of this functionality.

## 4.1 VM Isolation

65 VMs are the fundamental subject of the system. The VMM is responsible for applying the system security policy (SSP) to the VM and all resources. As basic functionality, the VMM must support a security policy that mandates no information transfer between VMs.

66 The VMM must support the necessary mechanisms to isolate the resources of all VMs. The VMM partitions a platform's physical resources for use by the supported virtual environments. Depending on the use case, a VM may require a completely isolated environment with exclusive access to system resources, or share some of its resources with other VMs. It must be possible to enforce a security policy that prohibits the transfer of data between VMs through shared devices. When the platform security policy allows the sharing of resources across VM boundaries, the VMM must ensure that all access to those resources is consistent with the policy. The VMM may delegate the responsibility for the mediation of sharing of particular resources to select Service VMs; however in doing so, it remains responsible for mediating access to the Service VMs, and each service VM must mediate all access to any shared resource that has been delegated to it in accordance with the SSP.

67 Devices, whether virtual or physical, are resources requiring access control. The VMM must enforce access control in accordance to system security policy. Physical devices are platform devices with access mediated via the VMM per the non-bypassability objective. Virtual devices may include virtual storage devices and virtual network devices. Some of the access control restrictions must be enforced internal to Service VMs, as may be the case for isolating virtual networks. VMMs may also expose purely virtual interfaces. These are VMM specific, and while they are not analogous to a physical device, they are also subject to access control.

68 The VMM must support the mechanisms to isolate all resources associated with virtual networks and to limit a VM's access to only those virtual networks for which it has been configured. The VMM must also support the mechanisms to control the configurations of virtual networks according to the SSP.

69 [O.VM_ISOLATION]

## 4.2 VMM Integrity

70 Integrity is a core security objective for Virtualization Systems. To achieve system integrity the integrity of each VMM component must be established and maintained. This objective concerns only the integrity of the VS—not the integrity of software running inside of VMs or of the physical platform. The overall objective is to ensure the integrity of critical components of a VS.

71 Initial integrity of a VS is established through a digitally signed installation or update package, or through integrity measurements made at launch. Integrity is maintained in a running system by careful protection of the VMM from untrusted users and software. For example, it must not be possible for software running within a Guest VM to exploit a vulnerability in a device or hypercall interface and gain

control of the VMM. The vendor must release patches for vulnerabilities as soon as practicable after discovery.

72 [O.VMM_INTEGRITY]

## 4.3 Maintain Platform Integrity

73 The integrity of the VMM depends on the integrity of the hardware and software on which the VMM relies. Although the VS does not have complete control over the integrity of the platform, the VS must ensure that no VS-hosted software or VS users are capable of undermining the integrity of the platform.

74 [O.PLATFORM_INTEGRITY]

## 4.4 Maintain Domain Integrity

75 While the VS is not responsible for the contents or correct functioning of software that runs within Guest VMs, it is responsible for ensuring that such software is not interfered with by VMs from other domains.

76 [O.DOMAIN_INTEGRITY]

## 4.5 Management Functions Access Control

77 Server VSs generally include a comprehensive management infrastructure. VMM management functions include VM configuration, virtualized network configuration, allocation of physical resources, and reporting. Only certain authorized system users (administrators) are allowed to exercise management functions.

78 Because of the privileges exercised by the VMM management functions, it must not be possible for the VMM's management components to be compromised without administrator notification. This means that unauthorized users cannot be permitted access to the management functions, and the management components must not be interfered with by Guest VMs or unprivileged users on other networks—including operational networks connected to the TOE.

79 VMMs include a set of management functions that collectively allow administrators to configure and manage the VMM, as well as configure Guest or Helper VMs. These management functions are specific to the virtualization system, distinct from any other management functions that might exist for the internal management of any given Guest VM. These VMM management functions are privileged, with the security of the entire system relying on their proper use. The VMM management functions can be classified into different categories and the policy for their use and the impact to security may vary accordingly.

80 The management functions might be distributed throughout the VMM (within the VMM and Service VMs). The VMM must support the necessary mechanisms to enable the control of all platform management functions according to the system security policy. When a management function is distributed among multiple Service VMs, the VMs must be protected using the security mechanisms of the Hypervisor and any Service VMs involved to ensure that the intent of the system security policy is not compromised. Additionally, since hypercalls permit Guest VMs to invoke the Hypervisor, and often allow the passing of data to the Hypervisor, it is important that the hypercall interface is well-guarded and that all parameters be validated.

81 The VMM maintains configuration data for every VM on the system. This configuration data, whether of Service or Guest VMs, must be protected. The mechanisms used to establish, modify and verify configuration data are part of the platform management functions and must be protected as such. The proper internal configuration of Service VMs that provide critical security functions can also greatly impact platform security. These configurations must also be protected. Internal configuration of Guest VMs should not impact overall platform security. The overall goal is to ensure that the VMM, including the environments internal to Service VMs, is properly configured and that all Guest VM configurations are maintained consistent with the system security policy throughout their lifecycle.

82 Server virtualization platforms are usually managed remotely. An administrator must, for example, remotely update virtualization software, start and shut down VMs, and manage virtualized network connections. The console could be run on a separate machine or it could itself run in a VM. In most cases, an administrator must communicate with a privileged management agent over a network. Communications with the management infrastructure must be protected from Guest VMs and operational networks.

83 [O.MANAGEMENT_ACCESS]

## 4.6 Manageable Enterprise Network

84 VSs can host VMs that are part of one or more larger physical networks. The VS itself is part of the network infrastructure and must be managed as such. This means it must be updated and patched as a normal part of enterprise network operations in order to prevent potential compromise of the VMM and all the networks and VMs that it hosts. The VS must support standards and protocols that help enhance manageability of the VS as an IT product.

85 [O.MANAGEABLE_NETWORK]

## 4.7 Entropy for VMs

86 In order to function as members of operational networks, VMs must be able to communicate securely with other network entities—whether virtual or physical. They must therefore have access to sources of entropy sufficient for them to communicate securely.

87 [O.VM_ENTROPY]

## 4.8 Audit

88 The purpose of audit is to capture and protect data about what happens on a system so that it can later be examined to determine what has happened in the past. Security-relevant events related to virtualization must be logged by the VMM and made available to an auditor. The system must also have a way to prevent the audit data from using all the CPU or resources available preventing a denial of service within the VMM.

89 [O.AUDIT]

# 5 SECURITY FUNCTIONAL REQUIREMENTS

The individual security functional requirements are specified in the sections below.

## 5.1 Conventions

The CC defines operations on Security Functional Requirements: assignments, selections, assignments within selections and refinements. This document uses the following font conventions to identify the operations defined by the CC:

- Assignment: Indicated with *italicized* text;
- Refinement made by PP author: Indicated by the word "Refinement" in **bold text** after the element number with additional text in **bold text** and deletions with ~~strikethroughs~~, if necessary;
- Selection: Indicated with underlined text;
- Assignment within a Selection: Indicated with *italicized and underlined* text;
- Iteration: Indicated by appending the iteration number in parenthesis, e.g., (1), (2), (3).

Extended SFRs are identified by having a label 'EXT' after the requirement name for TOE SFRs.

## 5.2 FAU_GEN.1 Audit Data Generation

90 FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a. Start-up and shutdown of audit functions;
b. [Specifically defined auditable events listed in Table 1].

91 FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a. Date and time of the event;
b. Type of event;
c. Subject and object identity (if applicable);
d. The outcome (success or failure) of the event; and
e. Additional information defined in Table 1.

*Application Note:*

92 The ST author can include other auditable events directly in Table 1; they are not limited to the list presented. The ST author should update the table in FAU_GEN.1.2 with any additional information generated. "Subject identity" in FAU_GEN.1.2 could be an administrator's user id or an identifier specifying a VM, for example.

***Assurance Activity:***

93 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 is described in the TSS.

94 The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP. The evaluator shall examine the 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.

95 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.

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

**Table 1 Auditable Events**

| Requirement | Auditable Events | Additional Audit Record Contents |
|---|---|---|
| FAU_GEN.1 | None. | None. |
| FAU_SAR.1 | None. | None. |
| FAU_STG.1 | None. | None. |
| FAU_STG_EXT.1 | Failure of audit data capture due to lack of disk space or pre-defined limit.<br>On failure of logging function, capture record of failure and record upon restart of logging function. | None |
| FCS_CKM.1 | None. | None. |
| FCS_CKM.2 | None. | None. |
| FCS_CKM.4 | None. | None. |
| FCS_COP.1(1) | None. | None. |
| FCS_COP.1(2) | None. | None. |
| FCS_COP.1(3) | None. | None. |
| FCS_COP.1(4) | None. | None. |
| FCS_RBG_EXT.1 | Failure of the randomization process. | No additional information. |
| FCS_ENT_EXT.1 | None. | None. |

| | | |
|---|---|---|
| FDP_VMS_EXT.1 | None. | None. |
| FDP_PPR_EXT.1 | Successful and failed attempts for VMs to connect to physical devices where connection is governed by configurable policy.<br>Security policy violations. | VM and physical device identifiers.<br>Identifier for the security policy that was violated. |
| FDP_VNC_EXT.1 | Successful and failed attempts to connect VMs to virtual and physical networking components.<br>Security policy violations.<br>Administrator configuration of inter-VM communications channels between VMs. | VM and virtual or physical networking component identifiers.<br>Identifier for the security policy that was violated. |
| FDP_RIP_EXT.1 | None. | None. |
| FDP_RIP_EXT.2 | None. | None. |
| FDP_HBI_EXT.1 | None. | None. |
| FTP_TRP.1 | Initiation of the trusted channel.<br>Termination of the trusted channel.<br>Failures of the trusted path functions. | Identification of the claimed user identity. |
| FTP_UIF_EXT.1 | None. | None. |
| FTP_UIF_EXT.2 | None. | None. |
| FIA_PMG_EXT.1 | None. | None. |
| FIA_UIA_EXT.1 | Administrator authentication attempts<br>All use of the identification and authentication mechanism. | Provided user identity, origin of the attempt (e.g. console, remote IP address). |
| FIA_X509_EXT.1 | Failure to validate a certificate. | Reason for failure. |
| FIA_X509_EXT.2 | None. | None. |

| | | |
|---|---|---|
| FMT_SMF.1 | Updates to the TOE.<br>Configuration changes (system, network, audit function, Guest VM time, etc.).<br>Start-up and shutdown of the TOE<br>VM Start/Stop/Suspend events.<br>Guest VM resource usage (memory, network, cpu, etc.). | Configuration changes. |
| FMT_SMR.2 | None. | None. |
| FMT_MSA_EXT.1 | None. | None. |
| FMT_MOF.1 | None. | None. |
| FMT_SMO_EXT.1 | None. | None. |
| FTP_TRP.1 | Start and end of remote management session. | User ID and remote source (IP Address) if feasible. |
| FPT_TUD_EXT.1 | Initiation of update.<br>Failure of signature verification. | No additional information. |
| FPT_VIV_EXT.1 | None. | None. |
| FPT_HCL_EXT.1 | Attempts to access disabled hypercall interfaces.<br>Security policy violations. | Interface for which access was attempted.<br>Identifier for the security policy that was violated. |
| FPT_VDP_EXT.1 | None. | None. |
| FPT_HAS_EXT.1 | None. | None. |
| FPT_EEM_EXT.1 | None. | None. |
| FPT_RDM_EXT.1 | Transfer of removable media or device between VMs. | None. |
| FPT_DVD_EXT.1 | None. | None. |

### 5.2.1 FAU_SAR.1 Audit Review

97 FAU_SAR.1.1 The TSF shall provide **administrators** with the capability to read **all information** from the audit records.

98 FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

***Assurance Activity:***

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 evaluator shall review the operational guidance for the procedure on how to review the audit records. The assurance activity for this requirement is performed in conjunction with the assurance activity for FAU_GEN.1.

### 5.2.2 FAU_STG.1 Protected Audit Trail Storage

99 FAU_STG.1.1 The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

100 FAU_STG.1.2 The TSF shall be able to **prevent** unauthorized modifications to the stored audit records in the audit trail.

*101* *Application Note:*

102 The below Assurance Activity 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.

***Assurance Activity:***

103 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. The evaluator shall perform the following tests:

- Test 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 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.

### 5.2.3 FAU_STG_EXT.1 Off-Loading of Audit Data

104 FAU_STG_EXT.1.1. The TSF shall be able to transmit the generated audit data to an external IT entity using a trusted channel implementing the [selection: IPsec, SSH, TLS, TLS/HTTPS] protocol.

***Assurance Activity:***

105 The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided. Testing of the trusted channel mechanism will be performed as specified in the associated assurance activities for the

particular trusted channel mechanism. The evaluator shall also 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. The evaluator shall perform the following test for this requirement:

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

106 FAU_STG_EXT.1.2 The TSF shall [selection: drop new audit data, overwrite previous audit records according to the following rule: [assignment: *rule for overwriting previous audit records*], [assignment: *other action*]] when the local storage space for audit data is full.

*Application Note:*

107 The external log server might be used as alternative storage space in case the local storage space is full. The 'other action' could in this case be defined as 'send the new audit data to an external IT entity'.

***Assurance Activity:***

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

109 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_EXT.1.2.

## 5.3 Cryptographic Support (FCS)

### 5.3.1 FCS_CKM.1 Cryptographic Key Generation

110 FCS_CKM.1.1 The TSF shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [selection:

- **RSA schemes** using cryptographic key sizes of **2048-bit or greater** that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3;

- **ECC schemes** using **"NIST curves" P-256, P-384 and [selection: P-521, no other curves]** that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;

- **FFC schemes** using cryptographic key sizes of **2048-bit or greater** that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1

].

*Application Note:*

111 The ST author selects all key generation schemes used for key establishment and device authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2.1 and selected cryptographic protocols shall match the selection. When key generation is used for device authentication, the public key is expected to be associated with an X.509v3 certificate.

112 If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

***Assurance Activity:***

113 The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

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

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

***116 Key Generation for FIPS PUB 186-4 RSA Schemes***

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

118 Key Pair generation specifies 5 ways (or methods) to generate the primes *p* and *q*. These include:

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

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

***Key Generation for Elliptic Curve Cryptography (ECC)***

*FIPS 186-4 ECC Key Generation Test*

120 For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

*FIPS 186-4 Public Key Verification (PKV) Test*

121 For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

***Key Generation for Finite-Field Cryptography (FFC)***

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

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

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

124 and two ways to generate the cryptographic group generator g:

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

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

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

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

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

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

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

129 for each FFC parameter set and key pair.

### 5.3.2 FCS_CKM.2 Cryptographic Key Establishment

130 **FCS_CKM.2.1** The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: [selection:

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

].

*Application Note*:

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

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

133 The RSA-based key establishment schemes are described in Section 9 of NIST SP 800-56B; however, Section 9 relies on implementation of other sections in SP 800-56B. If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

134 The elliptic curves used for the key establishment scheme shall correlate with the curves specified in FCS_CKM.1.1.

135 The domain parameters used for the finite field-based key establishment scheme are specified by the key generation according to FCS_CKM.1.1.

136 ***Assurance Activity***:

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

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

139 The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

140 **Key Establishment Schemes**

***SP800-56A Key Establishment Schemes***

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

*142* *Function Test*

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

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

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

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

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

*148* *Validity Test*

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

150 The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM,

the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

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

***SP800-56B Key Establishment Schemes***

152 The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

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

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

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

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

- The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE shall not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

### 5.3.3 FCS_CKM.4 Cryptographic Key Destruction

155 FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [selection:

- For volatile memory, the destruction shall be executed by a single direct overwrite [selection: consisting of a pseudo-random pattern using the TSF's RBG, consisting of zeroes] followed by a read-verify.
    - If the read-verification of the overwritten data fails, the process shall be repeated again.

- For non-volatile EEPROM, the destruction shall be executed by a single, direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1), followed by a read-verify.
    - If the read-verification of the overwritten data fails, the process shall be repeated again.

- For non-volatile flash memory, the destruction shall be executed by [selection: a single, direct overwrite consisting of zeroes, a block erase] followed by a read-verify.
    - If the read-verification of the overwritten data fails, the process shall be repeated again.

- For non-volatile memory other than EEPROM and flash, the destruction shall be executed by overwriting three or more times with a random pattern that is changed before each write.

]

that meets the following: NIST SP 800-88.

*Application Note:*

156 The clearing indicated above applies to each intermediate storage area for plaintext key/cryptographic critical security parameter (i.e., any storage, such as memory buffers, that is included in the path of such data) upon the transfer of the key/cryptographic critical security parameter to another location.

***Assurance Activity:***

157 The evaluator shall check to ensure the TSS lists each type of plaintext key material and its origin and storage location.

158 The evaluator shall verify that the TSS describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, etc.).

159 The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. 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 three times with a random pattern that is changed before each write").

160 For each software and firmware key clearing situation the evaluator shall repeat the following tests.

- Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) to test that keys are cleared correctly, including all intermediate copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

161 For each key subject to clearing, including intermediate copies of keys that are persisted encrypted by the TOE the evaluator shall:

1) Load the instrumented TOE build in a debugger.
2) Record the value of the key in the TOE subject to clearing.
3) Cause the TOE to perform a normal cryptographic processing with the key from 1).
4) Cause the TOE to clear the key.
5) Cause the TOE to stop the execution but not exit.
6) Cause the TOE to dump the entire memory footprint of the TOE into a binary file.
7) Search the content of the binary file created in 4) for instances of the known key value from 1).

162 The test succeeds if no copies of the key from 1) are found in step 7) above and fails otherwise.

163 The evaluator shall perform this test on all keys, including those persisted in encrypted form, to ensure intermediate copies are cleared.

### 5.3.4 FCS_COP.1(1) Cryptographic Operation (AES Data Encryption/ Decryption)

164 **FCS_COP.1.1(1)** The TSF shall perform [**encryption/decryption**] in accordance with a specified cryptographic algorithm

- AES-CBC (as defined in FIPS PUB 197, and NIST SP 800-38A) mode,

- AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), and

[selection*:*

- AES Key Wrap (KW) (as defined in NIST SP 800-38F),
- AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F),
- AES-GCM (as defined in NIST SP 800-38D),
- AES-CCM (as defined in NIST SP 800-38C),
- AES-XTS (as defined in NIST SP 800-38E) mode,
- AES-CCMP-256 (as defined in NIST SP800-38C and IEEE 802.11ac-2013),
- AES-GCMP-256 (as defined in NIST SP800-38D and IEEE 802.11ac-2013),
- no other modes]

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

165 *Application Note:*

166 For the first selection of FCS_COP.1.1(1), the ST author should choose the mode or modes in which AES operates. For the second selection, the ST author should choose the key sizes that are supported by this functionality. 128-bit CBC and CCMP are required in order to comply with FCS_TLSC_EXT.1 and FCS_CKM.1.1(2).

167 Note that to comply with IEEE 802.11-2012, AES CCMP (which uses AES in CCM as specified in SP 800-38C) with cryptographic key size of 128 bits shall be implemented. Optionally, AES-CCMP-256 or AES-GCMP-256 with cryptographic key size of 256 bits may be implemented for IEEE 802.11ac connections. In the future, one of these modes may be required.

168 Support for 256-bit key sizes will be required for products entering evaluation after Quarter 3, 2015.

169 ***Assurance Activity:***

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

171 **AES-CBC Tests**

172 **AES-CBC Known Answer Tests**

173 There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

174 **KAT-1.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

175

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

176 **KAT-2.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

177 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

178 **KAT-3.** To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key *i* in each set shall have the leftmost *i* bits be ones and the rightmost *N-i* bits be zeros, for *i* in [1,N].

179 To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

180 **KAT-4.** To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

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

**182 AES-CBC Multi-Block Message Test**

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

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

**185 AES-CBC Monte Carlo Tests**

186 The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

187 # Input: PT, IV, Key

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

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

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

**AES-CCM Tests**

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

**128 bit and 256 bit keys**

**Two payload lengths**. One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).

**Two or three associated data lengths.** One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of $2^{16}$ bytes, an associated data length of $2^{16}$ bytes shall be tested.

**Nonce lengths**. All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.

**Tag lengths**. All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.

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

**Test 1**. For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

**Test 2**. For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

**Test 3**. For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.

209 **Test 4**. For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

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

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

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

213 **AES-GCM Test**

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

215 **128 bit and 256 bit keys**

216 **Two plaintext lengths**. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

217 **Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

218 **Two IV lengths**. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

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

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

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

222 **XTS-AES Test**

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

224 **256 bit (for AES-128) and 512 bit (for AES-256) keys**

225 **Three data unit (i.e., plaintext) lengths**. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or $2^{16}$ bits, whichever is smaller.

226 using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

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

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

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

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

231 **128 and 256 bit key encryption keys (KEKs)**

232 **Three plaintext lengths**. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).

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

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

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

236 One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).

237 One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

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

### 5.3.5 FCS_COP.1(2) Cryptographic Operation (Hashing)

239 **FCS_COP.1.1(2)** The TSF shall perform [***cryptographic hashing***] in accordance with a specified cryptographic algorithm SHA-1 and [selection*:* SHA-256, SHA-384, SHA-512, no other algorithms] and message digest sizes 160 and [selection*:* 256, 384, 512 bits, no other message digest sizes] that meet the following: [***FIPS Pub 180-4***].

240 **Application Note:**

241 Per NIST SP 800-131A, SHA-1 for generating digital signatures is no longer allowed, and SHA-1 for verification of digital signatures is strongly discouraged as there may be risk in accepting these signatures. It is expected that vendors will implement SHA-2 algorithms in accordance with SP 800-131A, and products entering into evaluation after Quarter 3, 2015 will be required to include SHA-2 algorithms.

242 SHA-1 is currently required in order to comply with FCS_TLSC_EXT.1 and FCS_CKM.1.1(2). Vendors are strongly encouraged to implement updated protocols that support the SHA-2 family; until updated protocols are supported, this PP allows support for SHA-1 implementations in compliance with SP 800-131A.

243 The intent of this requirement is to specify the hashing function. The hash selection shall support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used (for example, SHA 256 for 128-bit keys).

**Assurance Activity:**

244 The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present. The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

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

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

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

*248* *Short Messages Test - Bit-oriented Mode*

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

*250* *Short Messages Test - Byte-oriented Mode*

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

*252* *Selected Long Messages Test - Bit-oriented Mode*

253 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 ≤ i ≤ m. The message text shall be

pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

*254* *Selected Long Messages Test - Byte-oriented Mode*

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

*256* *Pseudo-randomly Generated Messages Test*

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

### 5.3.6 FCS_COP.1(3) Cryptographic Operation (Signature Algorithms)

258 FCS_COP.1.1(3) **Refinement**: The TSF shall perform [*cryptographic signature services* (generation and verification)] in accordance with a specified cryptographic algorithm

- [RSA schemes] <u>using</u> cryptographic key sizes [of 2048-bit or greater] that meet the following: [FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 4]

and [selection:

- <u>[ECDSA schemes] using [“NIST curves” P-256, P-384 and [selection: P-521, no other curves]] that meet the following: [FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 5];</u>
- <u>No other algorithms</u>

].

*Application Note:*

259 The ST Author should choose the algorithm implemented to perform digital signatures; if more than one algorithm is available, this requirement should be iterated to specify the functionality. For the algorithm chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm. RSA signature generation and verification is currently required in order to comply with FCS_TLSC_EXT.2.

260 ECDSA schemes will be required for products entering evaluation after Quarter 3, 2015.

***Assurance Activity:***

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

**262** **ECDSA Algorithm Tests**

***ECDSA FIPS 186-4 Signature Generation Test***

263 For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

***ECDSA FIPS 186-4 Signature Verification Test***

264 For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

**265 RSA Signature Algorithm Tests**

***Signature Generation Test***

266 The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator shall generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

267 The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

***Signature Verification Test***

268 The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

269 The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

### 5.3.7 FCS_COP.1(4) Cryptographic Operation (Keyed Hash algorithms)

270 **FCS_COP.1.1(4)** The TSF shall perform [***keyed-hash message authentication***] in accordance with a specified cryptographic algorithm HMAC-SHA-1 and [selection: HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, no other algorithms] and cryptographic key sizes [assignment: *key size (in bits) used in HMAC*] and message digest sizes 160 and [selection: 256, 384, 512, no other] bits that meet the following: [***FIPS Pub 198-1, "The Keyed-Hash Message Authentication Code, and FIPS Pub 180-4, "Secure Hash Standard***].

*Application Note:*

271 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. HMAC-SHA-1 is currently required in order to comply with FCS_TLSC_EXT.1 and FCS_CKM.1.1(2).

272 HMAC-SHA-256 and HMAC-SHA-384 will be required for products entering evaluation after Quarter 3, 2015 in order to support ciphersuites for FCS_TLS_EXT.2.1.

***Assurance Activity:***

273 The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

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

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

### 5.3.8 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

276 **FCS_RBG_EXT.1.1** The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)].

277 **FCS_RBG_EXT.1.2** The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: a software-based noise source, a hardware-based noise source] with a minimum of [selection: 128 bits, 192 bits, 256 bits] of entropy at least equal to the greatest security strength according to NIST SP 800-57, of the keys and hashes that it will generate.

*278* *Application Note:*

**279** NIST SP 800-90A contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used, and include the specific underlying cryptographic primitives used in the requirement. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.

280 If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1 may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2, the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

***Assurance Activity:***

281 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Annex E, Entropy Documentation and Assessment.

282 The evaluator shall also perform the following tests, depending on the standard to which the RBG conforms.

283 The evaluator shall perform 15 trials for the RBG implementation. If the RBG 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 RBG functionality.

284 If the RBG has prediction resistance enabled, each trial consists of (1) instantiate drbg, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator

verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

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

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

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

### 5.3.9 FCS_ENT_EXT.1 Extended: Entropy for Virtual Machines

287 FCS_ENT_EXT.1.1 The TSF shall provide a mechanism to make available to VMs entropy that meets FCS_RBG_EXT.1 through [selection: Hypercall interface, virtual device interface, passthrough access to hardware entropy source].

288 FCS_ENT_EXT.1.2 The TSF shall provide independent entropy across multiple VMs.

*Application Note:*

289 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.

290 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.

291 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 "passthrough 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.

292 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.

***Assurance Activity:***

293 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 VM. The evaluator shall perform the following tests:

- Test 1: The evaluator shall invoke entropy from each Guest VM. The evaluator shall verify that each VM acquires values from the interface.
- Test 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.

## 5.4 User Data Protection (FDP)

### 5.4.1 FDP_VMS_EXT.1 VM Separation

294 FDP_VMS_EXT.1.1 The TSF shall provide the following mechanisms for transferring data between Guest VMs: [selection: no mechanism, virtual networking, [assignment: *other inter-VM data sharing mechanisms*]].

295 FDP_VMS_EXT.1.2 The TSF shall allow Administrators to configure these mechanisms to [selection: enable, disable] the transfer of data between Guest VMs.

296 FDP_VMS_EXT.1.3 The TSF 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:*

297 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 Virtualization System as a whole is implemented with this fundamental requirement in mind.

298 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. This should be the same list of mechanisms supplied for FMT_MSA_EXT.1.

299 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.

300 For data transfer mechanisms implemented in terms of Hypercall functions, FDP_VMS_EXT.1.2 is met if FPT_HCL_EXT.1.2 is met for those Hypercall functions (VM access to Hypercall functions is configurable).

301 For data transfer mechanisms that use shared physical devices, FDP_VMS_EXT.1.2 is met if the device is listed in and meets FDP_PPR_EXT.1.1 (VM access to the physical device is configurable).

302 For data transfer mechanisms that use virtual networking, FDP_VMS_EXT.1.2 is met if FDP_VNC_EXT.1.1 is met (VM access to virtual networks is configurable).

303 FDP_VMS_EXT.1.3 is an attestation requirement. The vendor must attest that data cannot be transferred between Guest VMs except through the configurable mechanisms documented in FDP_VMS_EXT.1.1. The vendor must attest that there are no design or implementation flaws that permit the above mechanisms to be bypassed or defeated, or for data to be transferred through a different, undocumented mechanism.

***Assurance Activity**:*

The evaluator shall examine the TSS to verify that it documents all inter-VM communications mechanisms (as defined above), including how the mechanisms are configured, how they are invoked, and how they are disabled.

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

a. Create two VMs.

b. Test that communications cannot be passed between the VMs through the channel.

c. As an Administrator, enable inter-VM communications between two VMs.

d. Test that communications can be passed through the inter-VM channel.

e. As an Administrator again, disable inter-VM communications between the two VMs.

f. Test that communications can no longer be passed through the channel.

FDP_VMS_EXT.1.2 is met if communications is successful in step (d) and unsuccessful in step (f).

FMT_MSA_EXT.1.2 is met if communication is unsuccessful in step (b).

The evaluator must ensure that the ST includes the following statement attesting that there are no other ways for data to be transferred between VMs other than those listed in FDP_VMS_EXT.1.1:

304 A Guest VM cannot access the data of another Guest VM, or transfer data to another Guest VM other than through the mechanisms described in FDP_VMS_EXT.1.1 when expressly enabled by an authorized Administrator. There are no known design or implementation flaws that permit the above mechanisms to be bypassed or defeated, or for data to be transferred through undocumented mechanisms. This claim does not apply to covert channels or architectural side-channels.

### 5.4.2 FDP_PPR_EXT.1 Physical Platform Resource Controls

305 FDP_PPR_EXT.1.1 The TSF shall allow an authorized administrator to control VM access to the following physical platform resources: [assignment: *list of physical platform resources the VMM is able to control access to*].

306 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*]].

307 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*]].

*308* *Application Note:*

309 This requirement specifies that the VMM controls access to physical platform resources, and indicates that it must be configurable, but does not specify the means by which that is done. The ST author should list the physical platform resources that can be configured for Guest VM access by the administrator. Guest VMs may not be allowed direct access to certain physical resources; those resources are listed in the second element. 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 listed in the third element; if there are no such resources, then "no physical platform resources" is selected.

***310*** ***Assurance Activity:***

311 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 is described. This description shall cover all of the physical platforms allowed in the evaluated configuration by the ST. This description shall include 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. The evaluator shall ensure that the TSS describes how the Guest VM is associated with each physical resources, 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 VMs and physical platform resources.

312 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).

313 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.

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

- Test 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 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 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 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 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.

### 5.4.3 FDP_VNC_EXT.1 Virtual Networking Components

315 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.

316 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:*

317 Virtual networks must be isolated from one another to provide assurance 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.

318 FDP_VNC_EXT.1.2 is an attestation requirement. The vendor must attest that traffic traversing a virtual network is visible only to Guest VMs that are configured by an Administrator to be members of that virtual network, and that there are no known design or implementation flaws that permit the virtual networking configuration to be bypassed or defeated, or for data to be transferred through undocumented mechanisms.

***Assurance Activity***:

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

- Test 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 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.

320 The evaluator must ensure that the ST includes the following statement attesting that virtual network traffic is visible only to VMs configured to be on that virtual network:

321 Traffic traversing a virtual network is visible only to Guest VMs that are configured by an Administrator to be members of that virtual network. There are no known design or implementation flaws that permit the virtual networking configuration to be bypassed or defeated, or for data to be transferred through undocumented mechanisms. This claim does not apply to covert channels or architectural side-channels.

### 5.4.4 FDP_RIP_EXT.1 Residual information in memory

322 FDP.RIP_EXT.1 The TSF shall ensure that any previous information content of physical memory is cleared prior to allocation to a Guest VM.

*Application Note:*

323 Physical memory must be zeroed before it is made accessible to a VM for general use by a Guest OS.

324 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.

325 “For general use” means for use by the Guest OS in its page tables for running applications or system software.

326 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.

***Assurance Activity***:

327 1. 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.

328 2. The evaluator shall run a kernel-mode application or driver in the Guest OS that allocates a large (10MB) readable and writeable buffer. The test is to read each location in the buffer to ensure that every location contains a value of 0.

### 5.4.5 FDP_RIP_EXT.2 Residual information on disk

329 FDP.RIP_EXT.2 The TSF shall ensure that any previous information content of physical disk storage is cleared prior to allocation to a Guest VM.

*Application Note:*

330 Disk storage must be zeroed before it is made accessible to a VM for use by a Guest OS.

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

332 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.

***333*** ***Assurance Activity:***

The evaluator shall ensure that the TSS documents the conditions under which physical disk storage is not cleared prior to allocation to a Guest VM.

The evaluator shall perform the following tests:

1. The evaluator (as an unprivileged VM user) must create a new, large file (10MB) in the VM's file system. The test is to read each location in the file to ensure that every location contains a value of 0. This can be done using a custom tool or a binary file editor or viewer.
2. The evaluator (as VS Administrator) must create a virtual disk and connect it to a VM. As an unprivileged VM user, the evaluator must then create a large (10 MB) memory-mapped file on the virtual disk. The test is to read each location in the file to ensure that every location contains a value of 0. This can be done using a custom tool or a binary file editor or viewer.

### 5.4.6 FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

334 FDP_HBI_EXT.1.1 The TSF shall use [selection: no mechanism, [assignment: *list of platform-provided, hardware-based mechanisms*]] to constrain a Guest VM's direct access to the following physical devices: [selection: no devices, [assignment: *physical devices to which the VMM allows Guest VMs physical access*]].

*Application Note:*

335 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.

336 If the VMM does not allow Guest VMs direct access to any physical devices, then the “no mechanisms” and “no devices” selections can be made.

***337*** ***Assurance Activity:***

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

339 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.

## 5.5 Trusted Path/Channel (FTP)

### 5.5.1 FTP_TRP.1 Trusted Path for Remote Administration

340 FTP_TRP.1.1 **Refinement**: The TSF shall use [selection: IPsec, TLS, TLS/HTTPS] 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 endpoints and protection of the communicated data from disclosure and detection of modification of the communicated data.

341 FTP_TRP.1.2 **Refinement**: The TSF shall permit remote administrators to initiate communication via the trusted path.

342 FTP_TRP.1.3 **Refinement**: The TSF shall require the use of the trusted path for all remote administration actions.

*Application Note:*

343 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 by the protocol chosen in the first selection. The ST author chooses the mechanism or mechanisms supported by the TOE, and then ensures that the detailed requirements in Annex B corresponding to their selection are copied to the ST if not already present.

***Assurance Activity:***

344 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. The evaluator shall confirm that the operational guidance contains instructions for establishing the remote administrative sessions for each supported method. The evaluator shall also perform the following tests:

- Test 1: The evaluators shall ensure that communications using each specified (in the operational guidance) remote administration method is tested during the course of the evaluation, setting

up the connections as described in the operational guidance and ensuring that communication is successful.

- Test 2: For each method of remote administration supported, the evaluator shall follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish remote administrative sessions without invoking the trusted path.
- Test 3: The evaluator shall ensure, for each method of remote administration, the channel data is not sent in plaintext.
- Test 4: The evaluator shall ensure, for each method of remote administration, modification of the channel data is detected by the TOE.

345 Further assurance activities are associated with the specific protocols.

### 5.5.2 FTP_UIF_EXT.1 User Interface: I/O Focus

346 FTP_UIF_EXT.1.1 The TSF shall indicate to the user which VM is currently connected to [selection: keyboard, mouse [assignment: *other supported user input devices*]].

*Application Note:*

347 This requirement applies to all users—whether Unprivileged User or Administrator.

348 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.

***Assurance Activity**:*

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

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

351 3. 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.

### 5.5.3 FTP_UIF_EXT.2 User Interface: Identification of VM

352 FTP_UIF_EXT.2.1 The TSF shall uniquely identify a VM's output display to a user.

*Application Note:*

353 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.

354 The requirement would be met, for example, by a border around a VM's screen display that identifies the VM.

***Assurance Activity***:

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.

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.

## 5.6 Identification and Authentication (FIA)

### 5.6.1 FIA_PMG_EXT.1 Extended: Password Management

355 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 settable and support passwords of 15 characters or greater.

*Application Note:*

356 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.

***Assurance Activity:***

357 The evaluator shall examine the operational guidelines 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. The evaluator shall also perform the following tests. Note that one or more of these tests may be performed with a single test case.

- Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible 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.

### 5.6.2 FIA_UIA_EXT.1 Administrator Identification and Authentication

358 FIA_UIA_EXT.1.1 The TSF shall allow the following actions prior to requiring the Administrator to initiate the identification and authentication process:

- [selection: no actions, [assignment*: list of services, actions performed by the TSF in response to local or remote requests*]]

359 FIA_UIA_EXT.1.2 The TSF shall require each administrative user to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that administrative user.

*Application Note:*

360 This requirement applies to users of services available from the TOE directly, and not services available by connecting from the platform, for instance. While it should be the case that few or no services are available to external entities prior to identification and authentication, if there are some available (perhaps ICMP echo) these should be listed in the assignment statement; otherwise “no other actions” should be selected.

361 Authentication can be password-based through the local console or through a protocol that supports passwords (such as SSH), or be certificate based (SSH, TLS).

***Assurance Activity:***

362 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/used, any protocol transactions that take place, and what constitutes a “successful logon”. 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, etc.) 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.

363 The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

- Test 1: The evaluator shall use the operational guidance to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.
- Test 2: The evaluator shall configure the services allowed (if any) according to the operational guidance, and then determine the services available to a remote or local administrator. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

### 5.6.3 FIA_X509_EXT.1 X.509 Certificate Validation

364 FIA_X509_EXT.1.1 The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted certificate.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.

- The TSF shall validate the revocation status of the certificate using [selection: the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, a Certificate Revocation List (CRL) as specified in RFC 5759].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.
    - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field.

*Application Note:*

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

366 Regardless of the selection of TSF or TOE platform, the validation is expected to end in a trusted root CA certificate in a root store managed by the platform.

367 FIA_X509_EXT.1.2 The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

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

***Assurance Activity:***

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

370 The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

371 The tests described must be performed in conjunction with the other Certificate Services assurance activities, including the use cases in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

- Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function (application validation, trusted channel setup, or trusted software update) failing. The evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

- Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.
- Test 3: The evaluator shall test that the TOE can properly handle revoked certificates – conditional on whether CRL or OCSP is selected; if both are selected, then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions may required to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.
- Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.
- Test 5: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.
- Test 6: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

### 5.6.4 FIA_X509_EXT.2 X.509 Certificate Authentication

372 FIA_X509_EXT.2.1 The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [selection: IPsec, TLS, HTTPS, SSH], and [selection: code signing for system software updates, code signing for integrity verification, [assignment: *other uses*], no additional uses].

*Application Note:*

373 FIA_X509_EXT.2.2 When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [selection: allow the administrator to choose whether to accept the certificate in these cases, accept the certificate, not accept the certificate].

*Application Note:*

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

***Assurance Activity:***

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

376 The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted

channel. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

377 The evaluator shall perform Test 1 for each function listed in FIA_X509_EXT.2.1 that requires the use of certificates:

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

## 5.7 Security Management (FMT)

### 5.7.1 FMT_SMF.1 Specification of Management Functions

378 FMT_SMF.1.1 The TSF shall be capable of performing the following management functions:

1) Ability to administer the Virtualization System locally and remotely;
2) Ability to update the Virtualization System, and to verify the updates using [selection: digital signature, published hash, no other mechanism] capability prior to installing those updates;
3) Ability to configure password policy
    - Minimum password length,
    - Minimum password complexity,
    - Maximum password lifetime.
4) Ability to create, delete, and configure VMs;
5) Ability to set default initial VM configurations;
6) Ability to configure virtual networks including VMs;
7) Ability to manage the audit system and audit data;
8) Ability to configure VM access to physical devices;
9) Ability to configure inter-VM data sharing;
10) Ability to enable/disable VM access to Hypercall functions;
11) Ability to configure removable media policy;
12) [selection:
    - Ability to configure the list of TOE-provided services available before an entity is identified and authenticated, as specified in FIA_UIA_EXT.1;
    - Ability to configure the cryptographic functionality;
    - Ability to change default authorization factors,
    - No other capabilities

].

*Application Note:*

379 The TOE must provide functionality for both local and remote administration.

380 Administration is considered “local” if management communications between the Administrator and the Management Subsystem do not travel outside the data center or site that hosts the Virtualization System, and the traffic does not travel on other than the Management network.

381 Administration is considered “remote” if the Administrator is located outside the data center or site, or if traffic between the Administrator and the Management Subsystem travels on a network other than the dedicated Management network—whether encrypted or not. Note that is it possible to perform “remote” administration from a VM running on the host that is being managed if communications between the VM and the Management Subsystem run over an operational (a.k.a. Guest) network.

382 The TOE must provide the capability for the administrator to verify that updates received came from a trusted source. They must be capable of performing this action using digital signatures, and optionally a published hash. If the TOE offers the ability for the administrator to configure the services available prior to identification or authentication, or if any of the cryptographic functionality on the TOE can be configured, then the ST author makes the appropriate choice or choices in the second selection. Likewise, if the TOE can be configured to require that default authentication factors (or passwords) are changed to a non-default value that meets the requirements of FIA_PMG_EXT.1. Otherwise, "no other capabilities" should be selected.

***383*** ***Assurance Activity***

384 The evaluator shall examine the TSS to ensure that it describes each management function listed. The evaluator shall examine the operational guidance to verify that it includes detailed instructions of how to configure and use the listed management functions.

385 The evaluator must login as authorized Administrator and perform the tests listed below. If the evaluator is able to perform the required actions, then the test is passed.

386 Function 1: The evaluator must access the Management Subsystem both locally and remotely, and verify that an Administrator is capable of changing a VS configuration setting.

387 Function 2: Specified in *FPT_TUD_EXT.1.*

388 Function 3: Specified in *FIA_PMG_EXT.1.* Additionally, it may be the case that the VS Management Subsystem arrives with default authorization factors (password) in place. If it does, then the selection “Change default authorization factors” must be made so that there is a mechanism to change these authorization factors. The operational guidance shall describe the method by which the user changes these factors when they are installing and configuring the VS. The TSS shall describe the default authorization factors that exist. The evaluators must also perform the following test:

- [conditional] If the TOE provides default authorization factors, the evaluator shall change these factors in the course of installing and configuring the VS as described in the operational guidance. The evaluator shall then confirm that the (old) authorization factors are no longer valid for administrative access.

389 Function 4: The evaluator must create, configure, and delete VMs.

390 Function 5: Specified in *FMT_MSA_EXT.1.*

391 Function 6: Specified in *FDP_VNC_EXT.1*.

392 Function 7: Specified in *FAU_SAR.1.*

393 Function 8: Specified in *FDP_PPR_EXT.1.*

394 Function 9: Specified in *FDP_VMS_EXT.1.*

395 Function 10: Specified in *FPT_HCL_EXT.1.*

396 Function 11: Specified in *FPT_RDM_EXT.1.*

### 5.7.2 FMT_SMR.2 Restrictions on Security Roles

397 FMT_SMR.2.1 The TSF shall maintain the roles:

- **Administrator.**
- **User.**

398 FMT_SMR.2.2 The TSF shall be able to associate users with roles.

399 FMT_SMR.2.3 The TSF shall ensure that the conditions

- **Administrator role shall be able to administer the TOE locally;**
- **Administrator role shall be able to administer the TOE remotely;**
- **Administrator role shall be able to manage the audit capabilities of the TOE.**

are satisfied.

*Application Note:*

400 Unprivileged users are not able to administer the TOE or manage the audit capabilities of the TOE.

401 FMT_SMR.2.2 requires that user accounts be associated with only one role. However, note that multiple users may have the same role, and the TOE is not required to restrict roles to a single person.

402 FMT_SMR.2.3 requires that an authorized administrator be able to administer the TOE through the local console and through a remote mechanism (IPsec, SSH, TLS, TLS/HTTPS). For multiple component TOEs, only the TOE components providing the management control and configuration of the other TOE components require a local administration interface. The Administrator role is used for managing the Audit system.

***Assurance Activity:***

403 The evaluator shall examine the TSS to verify that they describe the administrator role and the powers granted to and limitations of the role. The evaluator shall review the operational guidance to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration. In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this PP be tested; for instance, if the TOE can be administered through

a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

### 5.7.3 FMT_MSA_EXT.1 Default data sharing configuration

404 FMT_MSA_EXT.1.1 The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs using [selection: no mechanism, virtual networking, [assignment: *other inter-VM data sharing mechanisms*]].

405 FMT_MSA_EXT.1.2 The TSF shall allow Administrators to specify alternative initial configuration values to override the default values when a Guest VM is created.

*Application Note:*

406 By default the VMM must enforce a policy prohibiting sharing of data between VMs. The default policy applies to all mechanisms for sharing data between VMs, including inter-VM communication channels, shared physical devices, shared virtual devices, and virtual networks. The default policy does not apply to covert channels and architectural side-channels.

407 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 all other mechanisms through which data can be transferred between Guest VMs. This should be the same list of mechanisms supplied in FDP_VMS_EXT.1.

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

- User interface-based mechanisms, such as copy/pate and drag-and-drop.
- Shared virtual or physical devices
- API-based mechanisms such as Hypercalls.

***409*** ***Assurance Activity:***

410 This requirement is met if FDP_VMS_EXT.1 is met.

### 5.7.4 FMT_MOF.1 Management of security functions behavior

411 FMT_MOF.1.1 **Refinement**: The TSF shall restrict the ability to **perform the functions listed in FMT_SMF.1** to **authorized Administrators**.

*412* *Application Note:*

413 Management functions must be performed only by VS Administrators. The VS Management interface must authenticate users of the Management System. A user must authenticate to the Management Subsystem to receive Administrator credentials.

414 Management functions are listed in FMT_SMF.1.

415 Functions that may be either management or non-management (depending on local policy) include connecting/disconnecting removable devices to/from a VM, starting a VM, checkpointing a VM, and suspending a VM.

416 There is no requirement to authenticate unprivileged 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.

***417*** ***Assurance Activity:***

418 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 the security management functions and actions can be executed only by authorized Administrators.

419 The evaluator must attempt to access each of the functions and policies in FMT_SMF.1 as a non-Administrative user and verify that the attempts fail. In addition, the evaluator must attempt to perform the below-listed management functions. The requirement is met if all attempts fail.

420 Test 1. The evaluator must attempt to change a VS configuration setting.

421 Test 2: The evaluator must attempt to create, delete, and reconfigure a VM.

### 5.7.5 FMT_SMO_EXT.1 Separation of Management and Operational Networks

422 FMT_SMO_EXT.1.1 The TSF shall support the configuration of separate management and operational networks through [selection: physical means, logical means, TLS, TLS/HTTPS, IPsec, SSH].

*Application Note:*

423 Management communications must be separate from user workloads. Administrative communications—including communications between physical hosts concerning load balancing, audit data, VM startup and shutdown—must be separate from guest operational networks.

424 “Physical means” refers to using separate physical networks for management and operational networks. For example, the machines in the management network are connected by separate cables plugged into separate and dedicated physical ports on each physical host.

425 “Logical means” refers to using separate network cables to connect physical hosts together using general-purpose networking ports. The management and operational networks are kept separate within the hosts using separate virtualized networking components.

***Assurance Activity***:

426 The evaluator shall examine the TSS to verify that it describes how management and operational networks may be separated.

427 The evaluator shall examine the operational guidance to verify that it details how to configure the VS with separate Management and Operational Networks.

428 The evaluator shall configure the management network as documented. If separation is cryptographic or logical, then the evaluator shall capture packets on the management network. If Guest network traffic is detected, the requirement is not met.

## 5.8 Protection of the TSF (FPT)

### 5.8.1 FPT_TUD_EXT.1 Trusted Updates to the Virtualization System

429 FPT_TUD_EXT.1.1 The TSF shall provide *Administrators* the ability to query the currently executed version of the TOE firmware/software as well as the most recently installed version of the TOE firmware/software.

*430 Application Note:*

431 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.

432 FPT_TUD_EXT.1.2 The TSF shall provide Administrators the ability to manually initiate updates to TOE firmware/software and [selection: support automatic updates, no other update mechanism].

433 FPT_TUD_EXT.1.3 The TSF shall provide means to authenticate firmware/software updates to the TOE using a [selection: digital signature mechanism, published hash] prior to installing those updates.

*Application Note:*

434 The digital signature mechanism referenced in FPT_TUD_EXT.1.3 is one of the algorithms specified in FCS_COP.1(3).

435 If certificates are used by the update verification mechanism, certificates are validated in accordance with FIA_X509_EXT.1 and should be selected in FIA_X509_EXT.2.1. Additionally, FPT_TUD_EXT.2.1 must be included in the ST.

436 "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.

437 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).

***Assurance Activity***:

438 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.

439 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.

440 The evaluator shall perform the following tests:

- Test 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 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)

### 5.8.2 FPT_VIV_EXT.1 VMM Isolation from VMs

441 FPT_VIV_EXT1.1 The TSF shall ensure that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform.

442 FPT_VIV_EXT.1.2 The TSF shall 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:*

443 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 Virtualization System 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.

444 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.

445 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.

446 At a minimum, the TSF should enforce the following:

a. On the x86 platform, a virtual System Management Interrupt (SMI) cannot invoke platform System Management Mode (SMM).

b. An attempt to update virtual firmware or virtual BIOS cannot cause physical platform firmware or physical platform BIOS to be modified.

c. An attempt to update virtual firmware or virtual BIOS cannot cause the VMM to be modified.

447 Of the above, (a) does not apply to platforms that do not support SMM. The rationale behind activity (c) 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.

448 This is an attestation requirement. The vendor must attest that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. The vendor must attest that there are no design or implementation flaws that permit the above.

***Assurance Activity**:*

449 The evaluator ensures that the ST includes the following statement attesting that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform:

450 Software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. There are no known design or implementation flaws that bypass or defeat VM isolation.

### 5.8.3 FPT_HCL_EXT.1 Hypercall Controls

451 FPT_HLC_EXT.1.1 The TSF shall provide a Hypercall interface for Guest VMs to use to invoke functionality provided by the VMM.

452 FPT_HCL_EXT.1.2 The TSF shall allow administrators to configure any VM's Hypercall interface to [selection: enable, disable] access to Hypercall functions.

453 FPT_HCL_EXT.1.3 The TSF shall permit exceptions to the configuration of the following Hypercall interface functions: [assignment: *list of functions that are not subject to the configuration controls in FPT_HCL_EXT.1.2*].

454 FPT_HCL_EXT.1.4 The TSF shall validate the parameters passed to the hypercall interface prior to execution of the VMM functionality exposed by that interface.

*Application Note:*

455 The purpose of this requirement is to help ensure the integrity of the VMM by reducing the attack surface exposed to Guest VMs.

456 A Hypercall interface allows a set of VMM functions to be invoked by software running within a VM. Hypercall interfaces are used by virtualization-aware VMs to communicate and exchange data with the VMM. For example, a VM could use a hypercall interface 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 VMs, these interfaces constitute attack surface. In order to minimize attack surface, these interfaces must be limited to the minimum needed to support VM functionality.

457 For the selection in FPT_HCL_EXT.1.2, the ST author selects the applicable actions that administrators can perform to configure functions supported by the interface.

458 For the assignment in FPT_HCL_EXT.1.3, the ST author lists the interface functions that cannot be configured per FPT_HCL_EXT.1.2.

459 A vendor-provided test harness may reduce evaluation time.

***460*** ***Assurance Activity:***

95 The evaluator shall examine the TSS or operational guidance to ensure it includes the documentation of the interface, including all possible functions available via the interface. Documentation must include, for each function, how to call the function, function parameters and legal values, configuration settings for enabling/disabling the function, and conditions under which the function can be disabled. The TSS must also specify those functions that cannot be disabled.

96 The evaluator shall examine the operational guidance to ensure it contains instructions for how to configure interface functions per FPT_HCL_EXT.1.2.

***461*** The evaluator shall perform the following tests:

1. For each configurable function that meets FPT_HCL_EXT.1.2, the evaluator shall follow the operational guidance to enable the function. The evaluator shall then attempt to call each function from within the VM. If the call is allowed, then the test succeeds.
2. For each configurable function, the evaluator shall configure the TSF to disable the function. The evaluator shall then attempt to call the function from within the VM. If the call is blocked, then the test succeeds.
3. For each function, the evaluator shall call the function from within a VM using parameter values outside the legal values specified in the TSS for that function. The test succeeds if all illegal values are rejected and the Virtualization System and VMM remain in a usable state.

Assurance Activity Note:

462 Test 3, above, is intended to ensure that Hypercall parameters are properly checked prior to use by the VMM. If an illegal Hypercall causes the Guest OS to crash, but the VS and VMM remains up and running, then the test passes (although a more graceful rejection would be preferable). If the VS or VMM crashes, then testing has exposed a potentially serious programming error and the test is failed.

### 5.8.4 FPT_VDP_EXT.1 Virtual Device Parameters

463 FPT_VDP_EXT.1.1 The TSF shall provide interfaces for virtual devices implemented by the VMM as part of the virtual hardware abstraction.

464 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:*

465 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 Virtualization System and the data comes from a virtual device driver supplied by the Virtualization Vendor.

***466*** ***Assurance Activity:***

***467*** The evaluator shall examine the TSS to ensure it includes the documentation of all virtual device interfaces, including I/O ports, protocols, and data formats. The evaluator shall perform the following test:

- Test 1: For each virtual device interface, the evaluator shall access the interface from within a VM using parameter values outside the legal values specified in the TSS. The test succeeds if all illegal values are rejected and the Virtualization System and VMM remain in a usable state.

Assurance Activity Note:

468 This test is intended to ensure that virtual device interface parameters are properly checked prior to use by the VMM. If an illegal virtual device access causes the Guest OS to crash, but the VS and VMM remain up and running, then the test passes (although a more graceful rejection would be preferable). If the VS or VMM crashes, then testing has exposed a potentially serious programming error and the test is failed.

### 5.8.5 FPT_HAS_EXT.1 Hardware Assists

469 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.

470 FPT_HAS_EXT.1.2 The VMM shall use [assignment: *list of hardware-based virtualization memory-handling assists*].

*Application Note:*

471 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.

472 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.

473 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.

***Assurance Activity***:

474 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.

### 5.8.6 FPT_EEM_EXT.1 Execution Environment Mitigations

475 FPT_EEM_EXT.1.1 The TSF shall take advantage of execution environment-based vulnerability mitigation mechanisms supported by the Platform such as:

476 [selection:

a) Address-space randomization,
b) Memory execution protection (e.g. DEP)
c) Stack buffer overflow protection,
d) Heap corruption detection
e) [assignment*: other mechanisms*]

].

*Application Note:*

477 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.

478 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.

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

***Assurance Activity***:

480 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.

### 5.8.7 FPT_RDM_EXT.1 Removable Devices and Media

481 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.

482 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, then

483 [selection:

a. the Administrator has granted explicit access for the media or device to be connected to the receiving domain.
b. the media in a device that is being transferred is ejected prior to the receiving domain being allowed access to the device.
c. the user of the receiving domain expressly authorizes the connection.

d. the device or media that is being transferred is prevented from being accessed by the receiving domain.

].

*Application Note:*

484 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.

485 Removable media is media that can be ejected from a device, such as a compact disc, floppy disk, SD, or compact flash memory card.

486 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).

487 For purposes of this requirement, an Information Domain is:

a. A VM or collection of VMs,
b. The Virtualization System,
c. Host OS, or
d. Management Subsystem.

488 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.

489 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.

490 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 switch, etc.).

***491*** ***Assurance Activity:***

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

***493*** The evaluator shall perform the following test for each listed media or device:

- Test 1: 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.

### 5.8.8 FPT_DVD_EXT.1 Non-Existence of Disconnected Virtual Devices

494 FPT_DVD_EXT.1.1 The TSF shall limit a Guest VM's access to virtual devices to those that are present in the VM's current hardware configuration.

*Application Note:*

495 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 accessible by the VM.

496 Such interfaces include memory buffers and processor I/O ports.

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

***498*** ***Assurance Activity:***

499 The evaluator shall perform the following tests:

- Test 1: The evaluator shall connect a device to a VM, then using a device driver running in the guest, scan the VM's processor I/O ports to ensure that the device's ports are present. (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 I/O ports are no longer present.

# 6 SECURITY ASSURANCE REQUIREMENTS

500 The Security Objectives for the TOE in Section 3 were constructed to address threats identified in Section 2. The Security Functional Requirements (SFRs) in Section 4.2 are a formal instantiation of the Security Objectives. The PP draws from the CC Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

501 While this section contains the complete set of SARs from the CC, the Assurance Activities to be performed by an evaluator are detailed both in Section 4.2 as well as in this section.

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

503 After the ST has been approved for evaluation, the Common Criteria Testing Laboratory (CCTL) will obtain the TOE, supporting environmental IT, and the administrative guides for the TOE. The Assurance Activities listed in the ST (which will be refined by the CCTL to be TOE-specific, either within the ST or in a separate document) will then be performed by the CCTL. The results of these activities will be documented and presented (along with the administrative guidance used) for validation.

504 For each family, "Developer Notes" are provided on the developer action elements to clarify what, if any, additional documentation/activity needs to be provided by the developer. For the content/presentation and evaluator activity elements, additional assurance activities are described as a whole for the family, rather than for each element. Additionally, the assurance activities described in this section are complementary to those specified in Section 4.2.

505 The TOE security assurance requirements, summarized in Table 2, identify the management and evaluative activities required to address the threats identified in Section 2 of this PP.

**Table 2: TOE Security Assurance Requirements**

| Assurance Class | Assurance Components | Assurance Components Description |
|---|---|---|
| **Development** | ADV_FSP.1 | Basic Functional Specification |
| **Guidance Documents** | AGD_OPE.1 | Operational User Guidance |
| | AGD_PRE.1 | Preparative Procedures |
| **Life-Cycle Support** | ALC_CMC.1 | Labeling of the TOE |
| | ALC_CMS.1 | TOE CM Coverage |
| | ALC_TSU_EXT | Timely Security Updates |
| **Security Target Evaluation** | ASE_CCL.1 | Conformance Claims |
| | ASE_ECD.1 | Extended Components Definition |
| | ASE_INT.1 | ST Introduction |

| Assurance Class | Assurance Components | Assurance Components Description |
|---|---|---|
| | ASE_OBJ.1 | Security Objectives for the Operational Environment |
| | ASE_REQ.1 | Stated Security Requirements |
| | ASE_TSS.1 | TOE Summary Specification |
| **Tests** | ATE_IND.1 | Independent Testing – Sample |
| **Vulnerability Assessment** | AVA_VAN.1 | Vulnerability Survey |

## 6.1 Class ADV: Development

506 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 Assurance Activities contained in Section 4.2 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

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

**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:*

507 There are no specific assurance activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Section 4.2, 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 assurance activities being performed; if the evaluator is unable to perform an activity because the there is insufficient interface information, then an adequate functional specification has not been provided.

508

## 6.2 Class AGD: Guidance Documents

509 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.

510 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.

511 Guidance pertaining to particular security functionality is also provided; specific requirements on such guidance are contained in the assurance activities specified in Section 4.2.

### 6.2.1 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 assurance 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 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 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 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 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 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.

***Assurance Activity:***

512 Some of the contents of the operational guidance will be verified by the assurance activities in Section 4.2 and evaluation of the TOE according to the CEM. The following additional information is also required.

513 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.

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

515 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(2) 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.*

### 6.2.2 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 assurance activities to determine the required content with respect to preparative procedures.

**Content and presentation elements:**

AGD_PRE.1.1C The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2C The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

**Evaluator action elements:**

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

AGD_PRE.1.2E The evaluator *shall apply* the preparative procedures to confirm that the TOE can be prepared securely for operation.

***Assurance Activity:***

516 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.

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

## 6.3 Class ALC: Life-cycle support

518 At the assurance level provided for TOEs conformant to this PP, life-cycle support is limited to an examination of the TOE vendor's development and configuration management process. This is a result of the critical role that a developer's practices play in contributing to the overall trustworthiness of a product.

### 6.3.1 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.

***Assurance Activity:***

519 The evaluator shall verify that the TOE has been provided with its unique reference labeled. The evaluator shall verify that the CM documentation has been provided and that it describes the method used to uniquely identify each configuration item. The evaluator shall verify that the developer has used a CM system and that this system uniquely identifies each configuration item.

### 6.3.2 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.

***Assurance Activity:***

520 The evaluator shall verify that the developer has provided a configuration list for the TOE that contains each item highlighted above. The evaluator shall verify that each item in the configuration list is uniquely identified and its developer is indicated.

### 6.3.3 ALC_TSU_EXT Timely security updates

521 This component requires the TOE developer, in conjunction with any other necessary parties, to provide information as to how the Virtualization System 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:**

522 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:**

523 ALC_ TSU_EXT.1.1C The description shall include the process for creating and deploying security updates for the TOE software/firmware.

524 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:*

525 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.

526 ALC_TSU_EXT.1.3C The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

*Application Note:*

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

528 **Evaluator action elements:**

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

## 6.4 Class ASE: Security Target Evaluation

### 6.4.1 ASE_CCL.1 Conformance claims

**Developer action elements:**

ASE_CCL.1.1D The developer shall provide a conformance claim.

ASE_CCL.1.2D The developer shall provide a conformance claim rationale.

**Content and presentation elements:**

ASE_CCL.1.1C The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.

ASE_CCL.1.2C The CC conformance claim shall describe the conformance of the ST to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.

ASE_CCL.1.3C The CC conformance claim shall describe the conformance of the ST to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.

ASE_CCL.1.4C The CC conformance claim shall be consistent with the extended components definition.

ASE_CCL.1.5C The conformance claim shall identify all PPs and security requirement packages to which the ST claims conformance.

ASE_CCL.1.6C The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.

ASE_CCL.1.7C The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.

ASE_CCL.1.8C The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.

ASE_CCL.1.9C The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PPs for which conformance is being claimed.

ASE_CCL.1.10C The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.

**Evaluator action elements:**

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

***Assurance Activity:***

530 The evaluator shall verify that the developer has provided a conformance claim that meets the standards outlined above. The evaluator shall verify that the developer has provided a conformance claim rationale that meets the standard above.

### 6.4.2 ASE_ECD.1 Extended components definition

**Developer action elements:**

ASE_ECD.1.1D The developer shall provide a statement of security requirements.

ASE_ECD.1.2D The developer shall provide an extended components definition.

**Content and presentation elements:**

ASE_ECD.1.1C The statement of security requirements shall identify all extended security requirements.

ASE_ECD1.2C The extended components definition shall define an extended component for each extended security requirement.

ASE_ECD.1.3C The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.

ASE_ECD.1.4C The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.

ASE_ECD.1.5C The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements can be demonstrated.

**Evaluator action elements:**

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

ASE_ECD.1.2E The evaluator *shall confirm* that no extended component can be clearly expressed using existing components.

***Assurance Activity:***

531 The evaluator shall verify that the developer has provided a statement of security requirements in accordance with ASE_REQ.1. The evaluator shall verify that the statement identifies all requirements. The evaluator shall confirm that the developer has provided an extended components definition that meets the standards listed above.

### 6.4.3 ASE_INT.1 ST introduction

**Developer action elements:**

ASE_INT.1.1D The developer shall provide an ST introduction.

**Content and presentation elements:**

ASE_INT.1.1C The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.

ASE_INT.1.2C The ST reference shall uniquely identify the ST.

ASE_INT.1.3C The TOE reference shall identify the TOE.

ASE_INT.1.4C The TOE overview shall summarize the usage and major security features of the TOE.

ASE_INT.1.5C The TOE overview shall identify the TOE type.

ASE_INT.1.6C The TOE overview shall identify any non-TOE hardware/software/firmware required by the TOE.

ASE_INT.1.7C The TOE description shall describe the physical scope of the TOE.

ASE_INT.1.8C The TOE description shall describe the logical scope of the TOE.

**Evaluator action elements:**

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

ASE_INT.1.2E The evaluator *shall confirm* that the TOE reference, the TOE overview, and the TOE description are consistent with each other.

***Assurance Activity:***

532 The evaluator shall verify that the developer has provided an ST introduction. The evaluator shall confirm that the ST introduction contains an ST reference, a TOE reference, a TOE overview, and a TOE description that meet the content and presentation elements listed above.

### 6.4.4 ASE_OBJ.1 Security objectives for the operational environment

**Developer action elements:**

ASE_OBJ.1.1D The developer shall provide a statement of security objectives.

**Content and presentation elements:**

ASE_OBJ.1.1C The statement of security objectives shall describe the security objectives for the operational environment.

**Evaluator action elements:**

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

***Assurance Activity:***

533 The evaluator shall verify that the developer has provided a statement of security objectives describing the objectives for the TOE and the operational environment as a whole. The evaluator shall verify that the developer has provided a security objective rationale that meets the content and presentation requirements listed above.

### 6.4.5 ASE_REQ.1 Stated security requirements

**Developer action elements:**

ASE_REQ.1.1D The developer shall provide a statement of security requirements.

ASE_REQ.1.2D The developer shall provide a security requirements rationale.

**Content and presentation elements:**

ASE_REQ.1.1C The statement of security requirements shall describe the SFRs and the SARs.

ASE_REQ.1.2C All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.

ASE_REQ.1.3C The statement of security requirements shall identify all operations on the security requirements.

ASE_REQ.1.4C All operations shall be performed correctly.

ASE_REQ.1.5C Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.

ASE_REQ.1.6C The statement of security requirements shall be internally consistent.

**Evaluator action elements:**

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

***Assurance Activity:***

534 The evaluator shall verify that the developer has provided a statement of security requirements that describes all the SFRs and SARs. The evaluator shall ensure that the statement lists all operations on the security requirements and all the dependencies are satisfied. The evaluator shall verify that the developer has provided a security requirements rationale that meets the requirements listed above.

### 6.4.6 ASE_TSS.1 TOE summary specification

**Developer action elements:**

ASE_TSS.1.1D The developer shall provide a TOE summary specification.

**Content and presentation elements:**

ASE_TSS.1.1C The TOE summary specification shall describe how the TOE meets each SFR.

**Evaluator action elements:**

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

ASE_TSS.1.2E The evaluator *shall confirm* that the TOE summary specification is consistent with the TOE overview and the TOE description.

***Assurance Activity:***

535 The evaluator shall verify that the developer has provided the TSS. The evaluator shall verify that the TSS describes how the TOE meets each SFR. This evaluation can be done using the assurance requirements listed in Section 4.2

## 6.5 Class ATE: Tests

536 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 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.

### 6.5.1 ATE_IND.1 Independent testing – sample

537 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 4.2 are being met, although some additional testing is specified for SARs in Section 4.3. The Assurance Activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the 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.

***Assurance Activity:***

538 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 Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

539 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.

540 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).

541 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.

## 6.6 Class AVA: Vulnerability assessment

542 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 protection profiles.

### 6.6.1 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.

***Assurance Activity:***

543 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 Server 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.

# 7 RATIONALE

544 The rationale tracing the threats to the objectives and the objectives to the requirements is contained in the prose in Sections 2.0 and 3.0 and is included in table form below. The only outstanding mappings are those for the Assumptions and Organizational Security Policies; those are contained in Annex A below.

**Table 3 Threat-Objective-SFR Mapping**

| **SPD Threat Statements Section 2** | | **Security Objectives** | **SFRs** |
|---|---|---|---|
| | | | |
| Cross-Domain Data Leakage | T.DATA_LEAKAGE | O.VM_ISOLATION;<br>O.DOMAIN_INTEGRITY | FDP_VMS_EXT.1;<br>FDP_PPR_EXT.1;<br>FDP_VNC_EXT.1;<br>FDP_RIP_EXT.1;<br>FDP_RIP_EXT.2;<br>FDP_HBI_EXT.1;<br>FMT_MSA_EXT.1<br>FPT_VIV_EXT.1;<br>FPT_HCL_EXT.1;<br>FTP_UIF_EXT.1;<br>FTP_UIF_EXT.2;<br>FPT_EEM_EXT.1;<br>FPT_RDM_EXT.1 |
| Unauthorized Update | T.UNAUTHORIZED_UPDATE | O.VMM_INTEGRITY | FIA_PMG_EXT.1;<br>FIA_UIA_EXT.1;<br>FMT_SMF.1;<br>FMR_SMR.2;<br>FMT_MOF.1;<br>FMT_SMO_EXT.1;<br>FTP_TRP.1;<br>FMT_TUD_EXT.1;<br>FIA_X509_EXT.1;<br>FIA_X509_EXT.2 |
| Unauthorized Modification | T.UNAUTHORIZED_MODIFICATION | O.VMM_INTEGRITY<br>O.AUDIT | FAU_GEN.1;<br>FAU_SAR.1;<br>FAU_SAR.2;<br>FAU_SAR.3;<br>FAU_STG.1;<br>FDP_PPR_EXT.1;<br>FDP_HBI_EXT.1;<br>FTP_TRP.1;<br>FPT_VIV_EXT.1;<br>FPT_HPV_EXT.1;<br>FPT_VDP_EXT.1;<br>FPT_HAS_EXT.1;<br>FPT_EEM_EXT.1;<br>FPT_DVD_EXT.1;<br>FPT_CIM_EXT.1 |

| | | | |
|---|---|---|---|
| User Error | T.USER_ERROR | O.VM_ISOLATION | FDP_VMS_EXT.1;<br>FTP_UIF_EXT.1;<br>FTP_UIF_EXT.2;<br>FMT_MSA_EXT.1;<br>FPT_RDM_EXT.1 |
| Vulnerability In Third-Party Software | T.3P_SOFTWARE | O.VMM_INTEGRITY | FPT_EEM_EXT.1;<br>FPT_INT_EXT.1;<br>FPT_DDI_EXT.1 |
| VMM Compromise | T.VMM_COMPROMISE | O.VMM_INTEGRITY<br>O.VM_ISOLATION | FDP_HBI_EXT.1;<br>FDP_PPR_EXT.1;<br>FMT_SMO_EXT.1;<br>FTP_TRP.1;<br>FPT_TUD_EXT.1;<br>FIA_X509_EXT.1;<br>FIA_X509_EXT.2;<br>FPT_VIV_EXT.1;<br>FPT_HCL_EXT.1;<br>FPT_VDP_EXT.1;<br>FPT_HAS_EXT.1;<br>FPT_EEM_EXT.1;<br>FPT_DVD_EXT.1;<br>FPT_CIM_EXT.1 |
| Platform Compromise | T.PLATFORM_COMPROMISE | O.PLATFORM_INTEGRITY | FDP_HBI_EXT.1;<br>FDP_PPR_EXT.1;<br>FPT_VIV_EXT.1; |
| Unauthorized Access to Management Functions | T.UNAUTHORIZED_ACCESS | O.MANAGEMENT_ACCESS | FDP_VNC_EXT.1;<br>FIA_PMG_EXT.1;<br>FIA_UIA_EXT.1;<br>FMT_SMR.2;<br>FMT_MOF.1;<br>FMT_SMO_EXT.1;<br>FTP_TRP.1;<br>FCS_CKM.1;<br>FCS_CKM_EXT.4;<br>FCS_COP.1;<br>FCS_RBG_EXT.1;<br>FCS_HTTPS_EXT.1;<br>FCS_IPSEC_EXT.1;<br>FCS_SSH_EXT.1;<br>FCS_TLS_EXT.1 |
| Weak Cryptography Due to Insufficient Entropy | T.WEAK_CRYPTO | O.VM_ENTROPY | FCS_ENT_EXT.1 |

# 8 Annex A: Supporting Tables

545 In this Protection Profile, the focus in the initial sections of the document is to use a narrative presentation in an attempt to increase the overall understandability of the threats to Virtualization Systems; the methods used to mitigate those threats; and the extent of the mitigation achieved by compliant TOEs. This presentation style does not readily lend itself to a formalized evaluation activity, so this Annex contains the tabular artifacts that can be used for the evaluation activities associated with this document.

## Assumptions

546 The specific conditions listed in the following subsections are assumed to exist in the TOE's Operational Environment. These assumptions include both practical realities in the development of the TOE security requirements and the essential environmental conditions on the use of the TOE.

547 ST authors should ensure that the assumptions still hold for their particular technology; the table should be modified as appropriate.

**Table 4 Assumptions**

| Assumption Name | Assumption Description |
|---|---|
| A.PLATFORM_INTEGRITY | The platform has not been compromised prior to installation of the Virtualization System. |
| 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.TRUSTED_ADMIN | TOE Administrators are trusted to follow and apply all administrator guidance. |

548

## Threats

549 The following threats should be integrated into the threats that are specific to the technology by the ST authors when including the requirements described in this document. Modifications, omissions, and additions to the requirements may impact this list, so the ST author should modify or delete these threats as appropriate.

**Table 5 Threats**

| Threat Name | Threat Definition |
|---|---|
| T.DATA_LEAKAGE | If it is possible for data to leak between domains when prohibited by policy, then an adversary on one domain or network can obtain data from another domain. Such cross-domain data leakage can, for example, cause classified information, corporate proprietary information, or medical |

| | |
|---|---|
| | data to be made accessible to unauthorized entities. |
| T.UNAUTHORIZED_UPDATE | A malicious party attempts to supply the Administrator with an update to the product that may compromise the security features of the TOE. |
| T.UNAUTHORIZED_MODIFICATION | Malware running on the physical host must not be able to undetectably modify Virtualization System components while the system is running or at rest. Likewise, malicious code running within a virtual machine must not be able to modify Virtualization System components. |
| T.USER_ERROR | An administrator may unintentionally install or configure the TOE incorrectly, resulting in ineffective security mechanisms. |
| T.3P_SOFTWARE | Vulnerabilities in 3rd party software can lead to VMM compromise. Where possible, the VS should mitigate the results of potential vulnerabilities or malicious content in third-party code. |
| T.VMM_COMPROMISE | Failure of security mechanisms could lead to unauthorized intrusion into or modification of the VMM or bypass of the VMM altogether. |
| T.PLATFORM_COMPROMISE | The hosting of untrusted or malicious domains by the VS cannot be permitted to compromise the security and integrity of the platform on which the VS executes. |
| T.UNAUTHORIZED_ACCESS | A user may gain unauthorized access to the TOE data and TOE executable code. A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to data or TOE resources. A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data. |
| T.WEAK_CRYPTO | A threat of weak cryptography may arise if the VMM does not provide sufficient entropy to support security-related features that depend on entropy to implement cryptographic algorithms. |
| T.UNMANAGEABLE_NW | The Virtualization System itself is generally part of a larger enterprise network and must be updated and patched as a normal part of enterprise network operations. Such basic network hygiene is more difficult if the enterprise network is unmanageable. |

## Security Objectives for the TOE

550 The following tables contain objectives for the TOE and Operational Environment. As assumptions are added to the PP, these objectives should be augmented to reflect such additions.

**Table 6 Security Objectives for the TOE**

| Security Objective | Security Objective Definition |
|---|---|
| O.VM_ISOLATION | As basic functionality, the VMM must support a security policy that mandates no information transfer between VMs. |
| O.VMM_INTEGRITY | Integrity is a core security objective for Virtualization Systems. To achieve system integrity the integrity of each VMM component must be established and maintained. This objective concerns only the integrity of the Virtualization System—not the integrity of software running inside of VMs or of the physical platform. The overall objective is to ensure the integrity of critical components of a Virtualization System. |
| O.PLATFORM_INTEGRITY | The integrity of the VMM depends on the integrity of the hardware and software on which the VMM relies. Although the VS does not have complete control over the integrity of the platform, the VS should as much as possible try to ensure that no users or software hosted by the VS is capable of undermining the integrity of the platform. |
| O.DOMAIN_INTEGRITY | The VS responsible ensuring that software running in Guest VMs us not interfered with by VMs from other domains. |
| O.MANAGEMENT_ACCESS | Management functions must be exercised only by authorized Administrators. |
| O.MANAGEABLE_NETWORK | The VS must support standards and protocols that help enhance manageability of the VS as an IT product. |
| O.AUDIT | The purpose of audit is to capture and protect data about what happens on a system so that it can later be examined to determine what has happened in the past. |

## Security Objectives for the Operational Environment

**Table 7 Security Objectives for the Operational Environment**

| Security Objective | Security Objective Definition |
|---|---|
| OE.CONFIG | TOE administrators will configure the Virtualization System |

| | |
|---|---|
| | correctly to create the intended security policy. |
| 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. |

# 9 Annex B: Selection-Based Requirements

551 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 will need to be included.

## 9.1 FCS_HTTPS_EXT.1 HTTPS Protocol

552 FCS_HTTPS_EXT.1.1 The TSF shall implement the HTTPS protocol that complies with RFC 2818.

*Application Note:*

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

554 FCS_HTTPS_EXT.1.2 The TSF shall implement HTTPS using TLS as specified in FCS_TLS_EXT.1.

***Assurance Activity:***

555 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. 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.

## 9.2 FCS_IPSEC_EXT.1 IPsec Protocol

556 FCS_IPSEC_EXT.1.1 The TSF shall implement the IPsec architecture as specified in RFC 4301.

*557 Application Note:*

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

559 While there must be a means to order the rules, a general approach to ordering is not mandated, as long as the SPD 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.

***560 Assurance Activity:***

561 The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. 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 those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

562 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.

563 *Operational Guidance*

564 The evaluator shall examine the operational guidance to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases – a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the operational guidance is consistent with the description in the TSS, and that the level of detail in the operational guidance is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

565 *Tests*

566 The evaluator uses the operational guidance to configure the TOE to carry out the following tests:

- Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

- Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios shall exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and operational guidance. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.

567 FCS_IPSEC_EXT.1.2 The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

568 The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

569 *Tests*

570 The evaluator uses the operational guidance to configure the TOE to carry out the following tests:

- Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The 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 the rule to allow the packet to flow in plaintext 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/platform created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was dropped.

571 FCS_IPSEC_EXT.1.3 The TSF shall implement transport mode and [selection: tunnel mode, no other mode].

572 ***Assurance Activity:***

573 The evaluator checks the TSS to ensure it states that the VPN can be established to operate in tunnel mode and/or transport mode (as identified in FCS_IPSEC_EXT.1.3).

574 *Operational Guidance*

575 The evaluator shall confirm that the operational guidance contains instructions on how to configure the connection in each mode selected.

576 *Tests*

577 The evaluator shall perform the following test(s) based on the selections chosen:

- Test 1 (conditional): If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE/platform to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE/platform and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE/Platform to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

- Test 2: The evaluator uses the operational guidance to configure the TOE/platform to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE/platform and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE/platform to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

578 FCS_IPSEC_EXT.1.4 The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) and [selection: AES-GCM-128 (specified in RFC 4106), AES-GCM-256 (specified in RFC 4106), no other algorithms] together with a Secure Hash Algorithm (SHA)-based HMAC.

579 ***Assurance Activity:***

580 The evaluator shall examine the TSS to verify that the algorithms AES-CBC-128 and AES-CBC-256 are implemented. If the ST author has selected either AES-GCM-128 or AES-GCM-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4) Cryptographic Operations (for keyed-hash message authentication).

581 *Operational Guidance*

582 The evaluator checks the operational guidance to ensure it provides instructions on how to configure the TOE/platform to use the algorithms, and if either AES-GCM-128 or AES-GCM-256 have been selected the guidance instructs how to use these as well.

583 *Tests*

584 The evaluator shall configure the TOE/platform as indicated in the operational guidance configuring the TOE/platform to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

585 FCS_IPSEC_EXT.1.5 The TSF shall implement the protocol: [selection:

- IKEv1 as defined in RFCs 2407, 2408, 2409, RFC 4109, [selection: no other RFCs for extended sequence numbers, RFC 4304 for extended sequence numbers], and [selection: no other RFCs for hash functions, RFC 4868 for hash functions];
- IKEv2 as defined in RFC 5996 and [selection: with no support for NAT traversal, with mandatory support for NAT traversal as specified in RFC 5996, section 2.23)], and [selection: no other RFCs for hash functions, RFC 4868 for hash functions]

].

586 *Application Note:*

587 If the TOE implements SHA-2 hash algorithms for IKEv1 or IKEv2, the ST author shall select RFC 4868. If the ST author selects IKEv1, FCS_IPSEC_EXT.1.15 must also be included in the ST. IKEv2 will be required for those TOEs entering evaluation after Quarter 3, 2016.

588 ***Assurance Activity:***

589 The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

590 *Operational Guidance*

591 The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE/platform to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE/platform to perform NAT traversal for the following test (if selected).

592 *Tests*

Tests are performed in conjunction with the other IPsec evaluation activities.

- (conditional): The evaluator shall configure the TOE/platform so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

593 FCS_IPSEC_EXT.1.6 The TSF shall ensure the encrypted payload in the [selection: IKEv1, IKEv2] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 3602 and [selection: AES-GCM-128, AES-GCM-256 as specified in RFC 5282, no other algorithm].

*594* *Application Note:*

595 AES-GCM-128 and AES-GCM-256 may only be selected if IKEv2 is also selected, as there is no RFC defining AES-GCM for IKEv1.

***596*** ***Assurance Activity:***

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

*598* *Operational Guidance*

599 The evaluator ensures that the operational guidance describes the configuration of the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE/platform to perform the following test for each ciphersuite selected.

*600* *Tests*

601 The evaluator shall configure the TOE/platform to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

602 FCS_IPSEC_EXT.1.7 The TSF shall ensure that [selection:

- IKEv1 Phase 1 SA lifetimes can be configured by an Security Administrator based on [selection:
  - number of packets/number of bytes];
  - length of time, where the time values can configured within [assignment*: integer range including 24*] hours;

  ];
- IKEv2 SA lifetimes can be configured by an Security Administrator based on [selection:
  - number of packets/number of bytes;
  - length of time, where the time values can configured within [assignment: *integer range including 24*] hours

  ]

].

*603* *Application Note:*

604 The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5). The ST author chooses either packet/volume-based lifetimes or time-based lifetimes. This requirement must be accomplished by providing Security Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE). Hardcoded limits are not acceptable. 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.

***605*** ***Assurance Activity***

*606* *Operational Guidance*

607 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the operational guidance. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 1 SA values for 24 hours. Currently there are no values mandated for the number of packets or number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

*608* *Tests*

609 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

610 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

- Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of packets (or bytes) allowed following the operational guidance. The evaluator shall configure a test peer with a packet/byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of packets (or bytes) through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

- Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 24 hours for the Phase 1 SA following the operational guidance. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that once 24 hours has elapsed, a new Phase 1 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

611 FCS_IPSEC_EXT.1.8 The TSF shall ensure that [selection:

- IKEv1 Phase 2 SA lifetimes can be configured by a Security Administrator based on [selection:
  - number of packets/number of bytes;
  - length of time, where the time values can be configured within [assignment: *integer range including 8*] hours;

  ];

- IKEv2 Child SA lifetimes can be configured by a Security Administrator based on [selection:
  - number of packets/number of bytes;
  - length of time, where the time values can be configured within [assignment: *integer range including 8*] hours;

]

].

612 *Application Note:*

613 The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5). The ST author chooses either packet/volume-based lifetimes or time-based lifetimes. This requirement must be accomplished by providing Security Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE). Hardcoded limits are not acceptable. 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.

614 ***Assurance Activity***

615 *Operational Guidance*

616 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the operational guidance. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 2 SA values for 8 hours. Currently there are no values mandated for the number of packets or number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

617 *Tests*

618 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

619 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

- Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of packets (or bytes) allowed following the operational guidance. The evaluator shall configure a test peer with a packet/byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of packets (or bytes) through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

- Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 8 hours for the Phase 2 SA following the operational guidance. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once 8 hours has elapsed, a new Phase 2 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

620 FCS_IPSEC_EXT.1.9 The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [assignment: *(one or more) number(s) of bits that is at least twice the security strength of the negotiated Diffie-Hellman group*] bits.

*621* *Application Note:*

622 For DH groups 19 and 20, the "x" value is the point multiplier for the generator point G.

623 Since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the SAs, the assignment in FCS_IPSEC_EXT.1. may contain multiple values. For each DH group supported, the ST author consults Table 2 in NIST SP 800-57 "Recommendation for Key Management – Part 1: General" to determine the security strength ("bits of security") associated with the DH group. Each unique value is then used to fill in the assignment. For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192.

***624*** ***Assurance Activity:***

625 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.). The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" meets the stipulations in the requirement.

626 FCS_IPSEC_EXT.1.10 The TSF shall generate nonces used in [selection: *IKEv1, IKEv2*] exchanges of length [selection:

- [assignment: *security strength associated with the negotiated Diffie-Hellman group*];
- at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash

] .

*627* *Application Note:*

628 The ST author must select the second option for nonce lengths if IKEv2 is also selected (as this is mandated in RFC 5996). The ST author may select either option for IKEv1.

629 For the first option for nonce lengths, since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the SAs, the assignment in FCS_IPSEC_EXT.1. may contain multiple values. For each DH group supported, the ST author consults Table 2 in NIST SP 800-57 "Recommendation for Key Management –Part 1: General" to determine the security strength ("bits of security") associated with the DH group. Each unique value is then used to fill in the assignment. For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192.

630 Because nonces may be exchanged before the DH group is negotiated, the nonce used should be large enough to support all TOE-chosen proposals in the exchange.

***631*** ***Assurance Activity:***

*632* *Tests*

- (conditional) If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.
- (conditional) If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

633 FCS_IPSEC_EXT.1.11 The TSF shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), and [selection: 19 (256-bit Random ECP), 5 (1536-bit MODP), 24 (2048-bit MODP with 256-bit POS), 20 (384-bit Random ECP), no other DH groups].

634 *Application Note:*

635 The selection is used to specify additional DH groups supported. This applies to IKEv1 and IKEv2 exchanges. For products entering into evaluation after Quarter 3, 2015, DH Group 19 (256-bit Random ECP) and DH Group 20 (384-bit Random ECP) will be required. 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.

636 ***Assurance Activity:***

637 The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

638 *Tests*

639 For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

640 FCS_IPSEC_EXT.1.12 The TSF 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.

641 *Application Note:*

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

***Assurance Activity:***

642 The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to

ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

643 *Tests*

644 The evaluator simply follows the guidance to configure the TOE/platform to perform the following tests.

- Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

- Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

- Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

- Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

645 FCS_IPSEC_EXT.1.13 The TSF shall ensure that all IKE protocols perform peer authentication using a [selection: RSA, ECDSA] that use X.509v3 certificates that conform to RFC 4945 and [selection: Pre-shared Keys, no other method].

646 *Application Note:*

647 At least one public-key-based Peer Authentication method is required in order to conform to this PP; one or more of the public key schemes is chosen by the ST author to reflect what is implemented. The ST author also ensures that appropriate FCS requirements reflecting the algorithms used (and key generation capabilities, if provided) are listed to support those methods. Note that the TSS will elaborate on the way in which these algorithms are to be used (for example, 2409 specifies three authentication methods using public keys; each one supported will be described in the TSS). Peer authentication using ECDSA X.509v3 certificates will be required for TOEs entering evaluation after Quarter 3, 2015.

***Assurance Activity:***

648 The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description shall be consistent with the algorithms as specified in FCS_COP.1(2) Cryptographic Operations (for cryptographic signature).

649 If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The evaluator shall check that the operational guidance describes how pre-shared keys are to be generated and established. The description in the TSS and the operational guidance shall also indicate how pre-shared

key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

*650* *Operational Guidance*

651 The evaluator ensures the operational guidance describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

652 In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the operational guidance 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 and marked “trusted”.

*653* *Tests*

654 For efficiency sake, the testing that is performed may be combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1. The following tests shall be repeated for each peer authentication selected in the FCS_IPSEC_EXT.1.1 selection above:

- Test 1: 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 2 [conditional]: The evaluator shall generate a pre-shared key off-TOE and use it, as indicated in the operational guidance, to establish an IPsec connection with the peer.

655 FCS_IPSEC_EXT.1.14 The TSF shall establish a trusted channel only to peers with valid certificates.

*656* *Application Note:*

Supported peer certificate algorithms are the same as FCS_IPSEC_EXT.1.1.

***Assurance Activity:***

657 The evaluator shall verify that the TSS describes how the DN in the certificate is compared to the expected DN.

*658* *Operational Guidance*

659 The evaluator shall ensure that the operational guidance includes configuration of the expected DN for the connection.

*660* *Tests*

661 The evaluator shall, if necessary, configure the expected DN according to the operational guidance. The evaluator shall send a peer certificate signed by a trusted CA with a DN that does not match an expected DN and verify that the TOE denies the connection.

662 (selection-based) FCS_IPSEC_EXT.1.15 The TSF shall ensure that IKEv1 Phase 1 exchanges use only main mode.

*663* *Application Note:*

664 FCS_IPSEC_EXT.1.15 is applicable only if IKEv1 is selected in FCS_IPSEC_EXT.1.5.

***665*** ***Assurance Activity:***

666 The evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

667 *Operational Guidance*

668 If the mode requires configuration of the TOE/platform prior to its operation, the evaluator shall check the operational guidance to ensure that instructions for this configuration are contained within that guidance.

669 *Tests*

670 The evaluator shall configure the TOE/platform as indicated in the operational guidance, and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.

## 9.3 FCS_SSHC_EXT.1 SSH Client Protocol

671 **FCS_SSHC_EXT.1.1** The TSF shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254, and [selection: 5647, 5656, 6187, 6668, no other RFCs].

672 *Application Note:*

673 The ST author selects which of the additional RFCs to which conformance is being claimed. Note that these need to be consistent with selections in later elements of this component (e.g., cryptographic algorithms permitted). RFC 4253 indicates that certain cryptographic algorithms are "REQUIRED". This means that the implementation must include support, not that the algorithms must be enabled for use. Ensuring that algorithms indicated as "REQUIRED" but not listed in the later elements of this component are implemented is out of scope of the assurance activity for this requirement.

674 **FCS_SSHC_EXT.1.2** The TSF shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, password-based.

675 ***Assurance Activity:***

676 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHC_EXT.1.5, and ensure that password-based authentication methods are also allowed.

677 *Tests*

678 Test 1: The evaluator shall, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection to an SSH server. Any configuration activities required to support this test shall be performed according to instructions in the operational guidance.

679 Test 2: Using the operational guidance, the evaluator shall configure the TOE to perform password-based authentication to an SSH server, and demonstrate that a user can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.

680 **FCS_SSHC_EXT.1.3** The TSF shall ensure that, as described in RFC 4253, packets greater than [assignment: *number of bytes*] bytes in an SSH transport connection are dropped.

681 *Application Note:*

682 RFC 4253 provides for the acceptance of “large packets” with the caveat that the packets should be of “reasonable length” or dropped. The assignment should be filled in by the ST author with the maximum packet size accepted, thus defining “reasonable length” for the TOE.

683 ***Assurance Activity:***

684 The evaluator shall check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

685 *Tests*

686 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

687 **FCS_SSHC_EXT.1.4** The TSF shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: *aes128-cbc, aes256-cbc, [selection: AEAD_AES_128_GCM, AEAD_AES_256_GCM, no other algorithms]*.

688 *Application Note:*

689 RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as encryption algorithms when the same algorithm is being used as the MAC algorithm. In the assignment, the ST author can select the AES-GCM algorithms, or "no other algorithms" if AES-GCM is not supported. If AES-GCM is selected, there should be corresponding FCS_COP entries in the ST.

690 ***Assurance Activity***

691 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

692 *Operational Guidance*

693 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

694 *Tests*

695 Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

696 Test 2: The evaluator shall configure an SSH server to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

697 **FCS_SSHC_EXT.1.5** The TSF shall ensure that the SSH transport implementation uses [selection: ssh-rsa, ecdsa-sha2-nistp256] and [selection: ecdsa-sha2-nistp384, x509v3-ecdsa-sha2-nistp256, x509v3-ecdsa-sha2-nistp384, no other public key algorithms] as its public key algorithm(s) and rejects all other public key algorithms.

698 *Application Note*

699 Implementations that select only ssh-rsa will not achieve the 112-bit security strength in the digital signature generation for SSH authentication as is recommended in NIST SP 800-131A. Future versions of this profile may remove ssh-rsa as a selection. If x509v3-ecdsa-sha2-nistp256 or x509v3-ecdsa-sha2-nistp384 are selected, then the list of trusted certification authorities must be selected in FCS_SSHC_EXT.1.9.

700 ***Assurance Activity***

701 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

702 *Operational Guidance*

703 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

704 *Tests*

705 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

706 Test 2: The evaluator shall configure an SSH server to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

707 **FCS_SSHC_EXT.1.6** The TSF shall ensure that the SSH transport implementation uses [selection: hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512] and [selection: AEAD_AES_128_GCM, AEAD_AES_256_GCM, no other MAC algorithms] as its data integrity MAC algorithm(s) and rejects all other MAC algorithm(s).

708 *Application Note*

709 RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as MAC algorithms when the same algorithm is being used as the encryption algorithm. RFC 6668 specifies the use of the sha2 algorithms in SSH.

710 ***Assurance Activity***

711 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

712 *Operational Guidance*

713 The evaluator shall also check the operational guidance to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the “none” MAC algorithm is not allowed).

714 *Tests*

715 Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

716 Test 2: The evaluator shall configure an SSH server to only allow the “none” MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

717 Test 3: The evaluator shall configure an SSH server to only allow the hmac-md5 MAC algorithm. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

718 **FCS_SSHC_EXT.1.7** The TSF shall ensure that [selection: diffie-hellman-group14-sha1, ecdh-sha2-nistp256] and [selection: ecdh-sha2-nistp384, ecdh-sha2-nistp521, no other methods] are the only allowed key exchange methods used for the SSH protocol.

719 ***Assurance Activity***

720 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

721 *Operational Guidance*

722 The evaluator shall also check the operational guidance to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

723 *Tests*

724 Test 1: The evaluator shall configure an SSH server to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

725 Test 2: The evaluator shall configure an SSH server to permit all allowed key exchange methods. The evaluator shall attempt to connect from the TOE to the SSH server using each allowed key exchange method, and observe that each attempt succeeds.

726 **FCS_SSHC_EXT.1.8** The TSF shall ensure that the SSH connection be rekeyed after no more than 2^28 packets have been transmitted using that key.

727 ***Assurance Activity***

728 *Tests*

729 The evaluator shall configure the TOE to create a log entry when a rekey occurs. The evaluator shall connect to the TOE with an SSH client and cause 2^28 packets to be transmitted from the client to the TOE, and subsequently review the audit log to ensure that a rekey occurred.

730 **FCS_SSHC_EXT.1.9** The TSF shall ensure that the SSH client authenticates the identity of the SSH server using a local database associating each host name with its corresponding public key or [selection: *a list of trusted certification authorities, no other methods*] as described in RFC 4251 section 4.1.

731 *Application Note*

732 The list of trusted certification authorities can only be selected if x509v3-ecdsa-sha2-nistp256 or x509v3-ecdsa-sha2-nistp384 are selected in FCS_SSHC_EXT.1.5.

733 ***Assurance Activity***

*734* *Tests*

735 Test 1: The evaluator shall delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator shall initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the user to accept or deny the key before continuing the connection.

736 Test 2: The evaluator shall add an entry associating a host name with a public key into the TOE's local database. The evaluator shall replace, on the corresponding SSH server, the server's host key with a different host key. The evaluator shall initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords).

737

## 9.4 FCS_SSHS_EXT.1 SSH Server Protocol

738 **FCS_SSHS_EXT.1.1** The TSF shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254, and [selection: 5647, 5656, 6187, 6668, no other RFCs].

*739* *Application Note*

740 The ST author selects which of the additional RFCs to which conformance is being claimed. Note that these need to be consistent with selections in later elements of this component (e.g., cryptographic algorithms permitted). RFC 4253 indicates that certain cryptographic algorithms are "REQUIRED". This means that the implementation must include support, not that the algorithms must be enabled for use. Ensuring that algorithms indicated as "REQUIRED" but not listed in the later elements of this component are implemented is out of scope of the assurance activity for this requirement.

741 **FCS_SSHS_EXT.1.2** The TSF shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, password-based.

***742*** ***Assurance Activity***

743 The evaluator shall check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHS_EXT.1.5, and ensure that password-based authentication methods are also allowed.

*744* *Tests*

745 Test 1: The evaluator shall, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection. Any configuration activities required to support this test shall be performed according to instructions in the operational guidance.

746 Test 2: The evaluator shall choose one public key algorithm supported by the TOE. The evaluator shall generate a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

747 Test 3: Using the operational guidance, the evaluator shall configure the TOE to accept password-based authentication, and demonstrate that a user can be successfully authenticated to the TOE over SSH using a password as an authenticator.

748 Test 4: The evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

749 **FCS_SSHS_EXT.1.3** The TSF shall ensure that, as described in RFC 4253, packets greater than [assignment: *number of bytes*] bytes in an SSH transport connection are dropped.

*750* *Application Note*

751 RFC 4253 provides for the acceptance of “large packets” with the caveat that the packets should be of “reasonable length” or dropped. The assignment should be filled in by the ST author with the maximum packet size accepted, thus defining “reasonable length” for the TOE.

***752*** ***Assurance Activity***

753 The evaluator shall check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

*754* *Tests*

755 The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

756 **FCS_SSHS_EXT.1.4** The TSF shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: AES128-cbc, AES256-cbc*,* [selection: AEAD_AES_128_GCM, AEAD_AES_256_GCM, no other algorithms].

*757* *Application Note*

758 RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as encryption algorithms when the same algorithm is being used as the MAC algorithm. In the assignment, the ST author can select the AES-GCM algorithms, or "no other algorithms" if AES-GCM is not supported. If AES-GCM is selected, there should be corresponding FCS_COP entries in the ST.

***759*** ***Assurance Activity***

760 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

*761* *Operational Guidance*

762 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

*763* *Tests*

764 Test 1: The evaluator shall establish a SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

765 Test 2: The evaluator shall configure an SSH client to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

766 **FCS_SSHS_EXT.1.5** The TSF shall ensure that the SSH transport implementation uses [selection: ssh-rsa, ecdsa-sha2-nistp256] and [selection: ecdsa-sha2-nistp384, x509v3-ecdsa-sha2-nistp256, x509v3-ecdsa-sha2-nistp384, no other public key algorithms] as its public key algorithm(s) and rejects all other public key algorithms.

767 *Application Note:*

768 Implementations that select only ssh-rsa will not achieve the 112-bit security strength in the digital signature generation for SSH authentication as is recommended in NIST SP 800-131A. Future versions of this profile may remove ssh-rsa as a selection.

769 ***Assurance Activity:***

770 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

771 *Operational Guidance*

772 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

773 *Tests*

774 Test 1: The evaluator shall establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate the TOE to an SSH client. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

775 Test 2: The evaluator shall configure an SSH client to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator shall attempt to establish an SSH connection from the SSH client to the TOE and observe that the connection is rejected.

776 **FCS_SSHS_EXT.1.6** The TSF shall ensure that the SSH transport implementation uses [selection: U] and [selection: AEAD_AES_128_GCM, AEAD_AES_256_GCM, no other MAC algorithms] as its MAC algorithm(s) and rejects all other MAC algorithm(s).

777 *Application Note*

778 RFC 5647 specifies the use of the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms in SSH. As described in RFC 5647, AEAD_AES_128_GCM and AEAD_AES_256_GCM can only be chosen as MAC algorithms when the same algorithm is being used as the encryption algorithm. RFC 6668 specifies the use of the sha2 algorithms in SSH.

779 ***Assurance Activity***

780 The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

781 *Operational Guidance*

782 The evaluator shall also check the operational guidance to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the “none” MAC algorithm is not allowed).

*783* *Tests*

784 Test 1: The evaluator shall establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

785 Test 2: The evaluator shall configure an SSH client to only allow the “none” MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

786 Test 3: The evaluator shall configure an SSH client to only allow the hmac-md5 MAC algorithm. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

787 **FCS_SSHS_EXT.1.7** The TSF shall ensure that [selection: diffie-hellman-group14-sha1, ecdh-sha2-nistp256] and [selection: ecdh-sha2-nistp384, ecdh-sha2-nistp521, no other methods] are the only allowed key exchange methods used for the SSH protocol.

***788*** ***Assurance Activity***

789 The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

*790* *Operational Guidance*

791 The evaluator shall also check the operational guidance to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

*792* *Tests*

793 Test 1: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

794 Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

795 **FCS_SSHS_EXT.1.8** The TSF shall ensure that the SSH connection be rekeyed after no more than 2^28 packets have been transmitted using that key.

***796*** ***Assurance Activity***

*797* *Tests*

798 The evaluator shall configure the TOE to create a log entry when a rekey occurs. The evaluator shall connect to the TOE with an SSH client and cause 2^28 packets to be transmitted from the client to the TOE, and subsequently review the audit log to ensure that a rekey occurred.

## 9.5 FCS_TLSC_EXT.1 TLS Client Protocol

799 **FCS_TLSC_EXT.1.1** The TSF shall implement [selection: TLS 1.2 (RFC 5246), TLS 1.1 (RFC 4346)] supporting the following ciphersuites:

- Mandatory Ciphersuites:
  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- [selection: Optional Ciphersuites:
  - TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
  - TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
  - TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
  - TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
  - no other ciphersuite].

800 *Application Note:*

801 The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other than the mandatory suites, then "None" should be selected. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

802 These requirements will be revisited as new TLS versions are standardized by the IETF.

803 If any ciphersuites are selected using ECDHE, then FCS_TLSC_EXT.1.5 is required.

804 In a future version of this cPP TLS v1.2 will be required for all TOEs.

805 ***Assurance Activity:***

806 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

807 *Tests*

808 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

809 Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

810 Test 3: The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

811 Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.

812 Test 5: The evaluator perform the following modifications to the traffic:

- Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
- Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.
- Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
- Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
- Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

813 **FCS_TLSC_EXT.1.2** The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

*814* *Application Note:*

815 The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server's certificate.

816 The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the assurance activity.

817 ***Assurance Activity:***

818 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

819 *Tests*

820 The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

821 Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.

822 Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

823 Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.

824 Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

825 Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

- The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

- The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

826 Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

827 Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

828 **FCS_TLSC_EXT.1.3** The TSF shall establish a trusted channel only if the peer certificate is valid.

*829* *Application Note:*

830 Validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

***831*** ***Assurance Activity:***

*832* *Tests*

833 Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

834 **FCS_TLSC_EXT.1.4** The TSF shall support mutual authentication using X.509v3 certificates.

*835* *Application Note:*

836 If TLS is used for FPT_ITC.1, then this component is required.

837 The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1. This requirement adds that this use must include the client must be capable of presenting a certificate to a TLS server for TLS mutual authentication.

***838*** ***Assurance Activity:***

839 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

*840* *Tests*

841 Test 1: The evaluator shall perform the following modification to the traffic:

- Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field shall not be the CA used to sign the client's certificate. The evaluator shall verify the connection is unsuccessful.

842 **FCS_TLSC_EXT.1.5** The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: secp256r1, secp384r1, secp521r1] and no other curves.

843 *Application Note:*

844 If ciphersuites with elliptic curves were selected in FCS_TLSC_EXT.1.1, this component is required.

845 This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1(2) and FCS_CKM.1 and FCS_CKM.2. This extension is required for clients supporting Elliptic Curve ciphersuites.

846 ***Assurance Activity:***

847 The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured.

848 *Tests*

849 Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

850

851 **FCS_TLSC_EXT.1 TLS Client Protocol**

852 **FCS_TLSS_EXT.1.1** The TSF shall implement [selection: TLS 1.2 (RFC 5246), TLS 1.1 (RFC 4346)] supporting the following ciphersuites:

- Mandatory Ciphersuites:
  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- [selection: Optional Ciphersuites:
  - TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
  - TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
  - TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
  - TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- no other ciphersuite].

853 *Application Note:*

854 The ciphersuites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional ciphersuites that are supported; if there are no ciphersuites supported other than the mandatory suites, then "None" should be selected. It is necessary to limit the ciphersuites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

855 These requirements will be revisited as new TLS versions are standardized by the IETF.

856 If any ciphersuites are selected using ECDHE, then FCS_TLSS_EXT.1.5 is required.

857 In a future version of this cPP TLS v1.2 will be required for all TOEs.

858 ***Assurance Activity:***

859 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

860 *Operational Guidance*

861 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

862 *Tests*

863 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

864 Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

865 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

866 Test 4: The evaluator shall perform the following modifications to the traffic:

- Modify at a byte in the client's nonce in the Client Hello handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

- Modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

- Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

- After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.

- Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

867 **FCS_TLSS_EXT.1.2** The TSF shall deny connections from clients requesting SSL 1.0, SSL 2.0, SSL 3.0, TLS 1.0, and [selection: TLS 1.1, none].

*868* *Application Note:*

869 All SSL versions and TLS v1.0 shall be denied. Any TLS versions not selected in FCS_TLSS_EXT.1.1 should be selected here.

***870*** ***Assurance Activity:***

871 The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

*872* *Operational Guidance*

873 The evaluator shall verify that any configuration necessary to meet the requirement are contained in the AGD guidance.

*874* *Tests*

875 The evaluator shall send a Client Hello requesting a connection with version SSL 1.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 2.0, SSL 3.0, TLS 1.0, and any selected TLS versions.

876 **FCS_TLSS_EXT.1.3** The TSF shall generate key agreement parameters [selection: over NIST curves [selection: secp256r1, secp384r1] and no other curves; Diffie-Hellman parameters of size 2048 bits and [selection: 3072 bits, no other size]].

*877* *Application Note:*

878 If the ST lists a DHE ciphersuite in FCS_TLSS_EXT.1.1, the ST must include the Diffie-Hellman selection in the requirement. FMT_SMF.1 requires the configuration of the key agreement parameters in order to establish the security strength of the TLS connection.

***879*** ***Assurance Activity:***

880 The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

881 *Operational Guidance*

882 The evaluator shall verify that any configuration necessary to meet the requirement is contained in the AGD guidance.

883 *Tests*

884 The evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

885 **(optional) FCS_TLSS_EXT.1.4** The TSF shall support mutual authentication of TLS clients using X.509v3 certificates.

886 ***Assurance Activity:***

887 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

888 *Operational Guidance*

889 The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

890 *Tests*

891 Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

892 Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

893 Test 3: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

894 Test 4: The evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied.

895 Test 5: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

896 Test 6: The evaluator shall perform the following modifications to the traffic:

- Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.

- Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

897 **FCS_TLSS_EXT.1.5** The TSF shall not establish a trusted channel if the peer certificate is invalid.

898 *Application Note:*

899 The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1. This requirement adds that this use must include support for client-side certificates for TLS mutual authentication.

900 Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

901 ***Assurance Activity:***

902 The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

903 *Operational Guidance*

904 The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

905 *Tests*

906 Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

907 Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

908 Test 3: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

909 Test 4: The evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied.

910 Test 5: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

911 Test 6: The evaluator shall perform the following modifications to the traffic:

- Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.

- Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

912 **FCS_TLSS_EXT.1.6** The TSF shall not establish a trusted channel if the distinguished name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match the expected identifier for the peer.

913 *Application Note*

914 This requirement only applies to those TOEs performing mutually-authenticated TLS (FCS_TLSS_EXT.1.4). The peer identifier may be in the Subject field or the Subject Alternative Name extension of the certificate. The expected identifier may either be configured, may be compared to the Domain Name, IP address, username, or email address used by the peer, or may be passed to a directory server for comparison. Matching should be performed by a bit-wise comparison.

915 ***Assurance Activity:***

916 (conditional) If the TOE implements mutual authentication, the evaluator shall verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

917 *Operational Guidance*

918 (conditional) If the TOE implements mutual authentication, and if the DN is not compared automatically to the Domain Name or IP address, username, or email address, then the evaluator shall ensure that the AGD guidance includes configuration of the expected DN or the directory server for the connection.

919 *Tests*

920 The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

## 9.6 FPT_TUD_EXT.2 Trusted Update based on Certificates

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, FPT_TUD_EXT.2.1 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.

***Assurance Activity:***

The assurance activity for this requirement is performed in conjunction with the assurance activity for FIA_X509_EXT.1 and FIA_X509_EXT.2.

## 9.7 Auditable Events

921 Depending on the specific requirements selected by the ST author from Section C1.1, the ST author should include the appropriate auditable events in the corresponding table in the ST for the requirements selected.

| Requirement | Auditable Events | Additional Audit Record Contents |
|---|---|---|
| FCS_HTTPS_EXT.1 | Failure to establish a HTTPS Session.<br>Establishment/Termination of a HTTPS session. | Reason for failure.<br>Non-TOE endpoint of connection (IP address) for both successes and failures. |
| FCS_IPSEC_EXT.1 | Failure to establish an IPsec SA.<br>Establishment/Termination of an IPsec SA. | Reason for failure.<br>Non-TOE endpoint of connection (IP address) for both successes and failures. |
| FCS_SSHC_EXT.1 | Failure to establish an SSH session<br>Establishment/Termination of an SSH session<br>Successful SSH rekey | Reason for failure<br>Non-TOE endpoint of connection (IP address). |
| FCS_SSHS_EXT.1 | Failure to establish an SSH session<br>Establishment/Termination of an SSH session<br>Successful SSH rekey | Reason for failure<br>Non-TOE endpoint of connection (IP address). |
| FCS_TLSC_EXT.1 | Failure to establish a TLS Session.<br>Establishment/Termination of a TLS session. | Reason for failure.<br>Non-TOE endpoint of connection (IP address). |
| FCS_TLSS_EXT.1 | Failure to establish a TLS Session.<br>Establishment/Termination of a TLS session. | Reason for failure.<br>Non-TOE endpoint of connection (IP address). |
| FPT_TUD_EXT.2 | None. | None. |

# 10 Annex C: Optional Requirements

922 The baseline requirements (those that must be performed by the TOE) are contained in the body of this cPP. Additionally, there are two other types of requirements specified in Annexes B and D.

923 The first type (in this Annex) is requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this cPP. The second type (in Annex B) is requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements in that appendix will need to be included in the body of the ST (e.g., cryptographic protocols selected in a trusted channel requirement).

924 Annex D contains Objective Requirements. These are requirements that are not required at this time, but are likely to be moved into the main body of this document in the future.

925 *No optional items have been identified at this time*.

# 11 Annex D: Objective Requirements

926 The baseline requirements (those that must be performed by the TOE) are contained in the body of this cPP. This Annex includes requirements that are desirable, but are not yet mandated. They are likely to become baseline requirements in a future version of this document.

## 11.1 FPT_INT_EXT.1 Support for Introspection

927 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:*

928 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.

929 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.

***Assurance Activity:***

930 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.

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

## 11.2 FPT_DDI_EXT.1 Device Driver Isolation

932 FPT_DDI_EXT.1.1 The TSF shall ensure that device drivers for physical devices are isolated from the VMM.

*933 Application Note:*

934 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.

935 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.

***Assurance Activity:***

936 The evaluator shall examine the TSS documentation to verify that it describes the mechanism used for device driver isolation.

## 11.3 FPT_CIM_EXT.1 Collection of Integrity Measurements

937 FPT_CIM_EXT.1.1 The TSF shall support the collection of integrity measurements from the host platform and make them available to the Management Subsystem for display to an Administrator.

*938* *Application Note:*

939 If the platform computes integrity measurements of the platform or Virtualization Solution (e.g. firmware measurements stored in a TPM), then the measurements must be made available to the Management Subsystem. An Administrator should be able to examine the measurements and make a judgment about the integrity of the host platform. This requirement does not mandate that the VS itself compute measurements, nor does it require that the measurements be sent off platform for attestation to a third-party.

***Assurance Activity:***

940 The evaluator shall verify that the TSS or Operational Guidance describes how integrity measurements are performed and made available to the Management Subsystem. The evaluator shall examine the operational guidance to verify that it documents how to access the measurements in the Management Subsystem.

941 The evaluator shall perform the following tests:

- Test 1: The evaluator shall start the VS, login as an Administrator, and verify that the measurements are viewable in the Management Subsystem.

## 11.4 FPT_IDV_EXT.1 Software Identification and Versions

942 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.

943 FPT_IDV_EXT.1.2 The TSF shall store SWIDs in a .swidtag file as defined in ISO/IEC 19770-2:2009.

*Application Note:*

944 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.

***Assurance Activity***:

945 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.

946 The evaluator shall perform the following test:

- Test 1: 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.

## 11.5 **Auditable Events**

947 Depending on the specific requirements selected by the ST author from Section C.1.1, the ST author should include the appropriate auditable events in the corresponding table in the ST for the requirements selected.

| Requirement | Auditable Events | Additional Audit Record Contents |
|---|---|---|
| FPT_INT_EXT.1 | None. | None. |
| FPT_DDI_EXT.1 | None. | None. |
| FPT_CIM_EXT.1 | Integrity measurements | None |
| FPT_IDV_EXT.1 | None. | None. |

# 12 Annex E: Entropy Documentation And Assessment

948 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.

## 12.1 Design Description

949 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.

950 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.

## 12.2 Entropy Justification

951 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.

## 12.3 Operating Conditions

952 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.

## 12.4 Health Testing

953 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.