Protection Profile for Mobile Device Management

7 March 2014

Version 1.1

Revision History

Version	Date	Description
1.0	21 October 2013	Initial Release
1.1	7 March 2014	Typographical changes and clarifications to front-matter

Table of Contents

Contents

1. INTRODUCTION 6
  1.1 Compliant Targets of Evaluation 6
  1.2 TOE Usage 7
2. SECURITY PROBLEM DESCRIPTION 7
  2.1 Threats 7
    2.1.1 Malicious and Flawed Applications 7
    2.1.2 Network Attack 8
    2.1.3 Network Eavesdropping 8
    2.1.4 Physical Access 8
  2.2 Assumptions 8
  2.3 Organization Security Policy 8
3. SECURITY OBJECTIVES 9
  3.1 Security Objectives for the TOE 9
    3.1.1 Protected Communications 9
    3.1.2 System Reporting 9
    3.1.3 Mobile Device Configuration 9
    3.1.4 Administration of Management Capability 10
  3.2 Security Objectives for the Operational Environment 10
4. SECURITY REQUIREMENTS 11
  4.1 Conventions 11
  4.2 TOE Security Functional Requirements 11
    Security Audit (FAU) 11
    Identification and Authentication (FIA) 13
    Security Management (FMT) 15
    Protection of the TSF (FPT) 21
  4.3 MDM Server or Platform Security Functional Requirements 22
    Security Audit (FAU) 22
    Cryptographic Support (FCS) 23
    Identification and Authentication (FIA) 42
    Protection of the TSF (FPT) 46
    Trusted Path/Channels (FTP) 48

4.4 MDM Agent or Platform Security Functional Requirements 50
Cryptographic Support (FCS) 50
Identification and Authentication (FIA) 67
Protection of the TSF (FPT) 70
4.5 Security Assurance Requirements 71
Class ASE: Security Target 72
Class ADV: Development 72
Class AGD: Guidance Documentation 74
Class ALC: Life-cycle Support 76
Class ATE: Tests 78
Class AVA: Vulnerability Assessment 79
5. RATIONALE 81
ANNEX A: SUPPORTING TABLES 82
A.1 Assumptions 82
A.2 Threats 83
A.3 Organizational Security Policies 83
A.4 Security Objectives for the TOE 84
A.5 Security Objectives for the Operational Environment 84
ANNEX B: OPTIONAL REQUIREMENTS 86
B.1 Optional TSF Requirements 86
Security Audit (FAU) 86
B.2 Optional MDM Server or MDM Server Platform Requirements 87
Security Audit (FAU) 87
ANNEX C: SELECTION-BASED REQUIREMENTS 89
C.1 Selection-Based TSF Requirements 89
Cryptographic Support (FCS) 89
C.2 Selection-Based MDM Server or MDM Server Platform Requirements 90
Cryptographic Support (FCS) 90
Identification and Authentication (FIA) 102
C.3 Selection-Based MDM Agent or MDM Agent Platform Requirements 102
Cryptographic Support (FCS) 102
Identification and Authentication (FIA) 112
C.4 Auditable Events 112
ANNEX D: OBJECTIVE REQUIREMENTS 117

D.1 Objective TSF Requirements 117
Security Audit (FAU) 117
Security Management (FMT) 118
D.2 Objective MDM Server or MDM Server Platform Requirements 119
TOE Access (FTA) 119
D.3 Objective MDM Agent or MDM Agent Platform Requirements 119
Security Audit (FAU) 119
Cryptographic Support (FCS) 120
ANNEX E: ENTROPY DOCUMENTATION AND ASSESSMENT 122
ANNEX F: GLOSSARY 123
Technical Definitions 123
Common Criteria Definitions 124
ANNEX G: INITIALIZATION VECTOR REQUIREMENTS 126

Table of Tables

Table 1 Security Assurance Requirements 72
Table 2 TOE Assumptions 82
Table 3 Threats 83
Table 4 Organizational Security Policies 83
Table 5 Security Objectives for the TOE 84
Table 6 Security Objectives for the Operational Environment 84
Table 7 TOE Security Functional Requirements and Auditable Events – Server 113
Table 8 TOE Security Functional Requirements and Auditable Events – Agent 115
Table 9 References and IV Requirements for NIST-approved Cipher Modes 126

1. INTRODUCTION

Mobile device management (MDM) products allow enterprises to apply security policies to mobile devices, such as smartphones and tablets. The purpose of these policies is to establish a security posture adequate to permit mobile devices to process enterprise data and connect to enterprise network resources.

This document provides a baseline set of Security Functional Requirements (SFRs) for an MDM system, which is the Target of Evaluation (TOE). The MDM system is only one component of an enterprise deployment of mobile devices. Other components, such as the mobile device platforms which enforce the security policies, and servers which host mobile application repositories, are out of scope.

1.1 Compliant Targets of Evaluation

The Mobile Device Management (MDM) system consists of two primary components: the MDM Server software and the MDM Agent. The **MDM** is considered the full collection of these parts, as they must act in concert. This situation necessitates a joint submission for evaluation of all components, regardless of vendor.

The MDM operational environment consists of the mobile device on which the MDM Agent resides, the platform on which the MDM Server runs, and an untrusted wireless network over which they communicate, as pictured below.

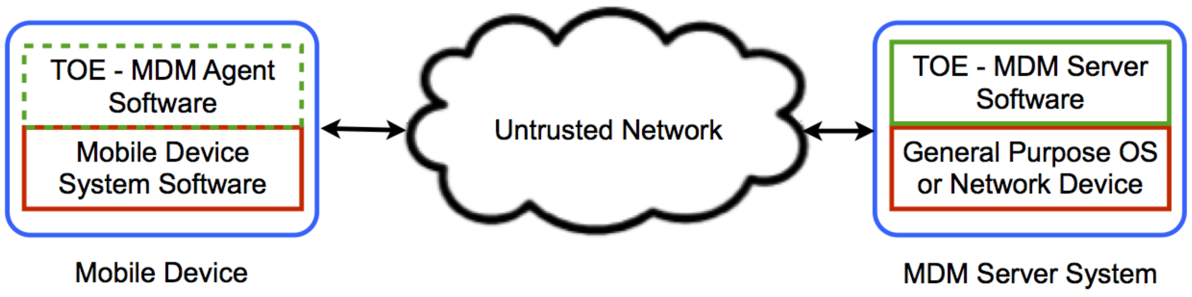
TOE - MDM Agent Software
Mobile Device System Software
Mobile Device
Untrusted Network
TOE - MDM Server Software
General Purpose OS or Network Device
MDM Server System

Figure 1: MDM System Operating Environment

The **MDM Agent** is installed on a mobile device as an application or is part of the mobile device's operating system (OS). The MDM Agent establishes a secure connection back to the MDM Server controlled by an enterprise administrator. The MDM Agent must closely interact with or be part of (as depicted by the dotted green line in Figure 1) the mobile device's platform to establish policies and receive queries about device status. The mobile device, in turn, has its own security requirements specified in the *Protection Profile for Mobile Device Fundamentals* against which the mobile device must be evaluated either concurrently with or before the MDM evaluation.

The **MDM Server** is an application on a general-purpose platform or on a network device, executing in a trusted network environment. The MDM server provides administration of the mobile device policies and reporting on mobile device behavior. The MDM Server is responsible for managing device enrollment, configuring and sending policies to the MDM Agents, collecting reports on device status, and sending commands to the Agents. The platform on which the MDM Server software runs is either a general purpose platform or a network device, as specified in the *General-Purpose Operating System Protection Profile* or the *Protection Profile for Network Devices*, respectively.

1.2 TOE Usage

Requirements in this Protection Profile are designed to address the security problem in the following use cases. Each use case involves acceptance of differing levels and types of risk.

- **Enterprise-owned device for general-purpose enterprise use**

  An Enterprise-owned device for general-purpose business use entails a significant degree of Enterprise control over configuration and software inventory. Enterprise administrators use an MDM product to establish policies on the mobile devices prior to user issuance. Users may use Internet connectivity to browse the web or access corporate mail or run Enterprise applications, but this connectivity may be under significant control of the Enterprise. The user may also be expected to store data and use applications for personal, non-enterprise use. The Enterprise administrator uses the MDM product to deploy security policies and query mobile device status. The MDM may issue commands for remediation actions.

- **Enterprise-owned device for specialized, high-security use**

  An Enterprise-owned device with intentionally-limited network connectivity, tightly-controlled configuration, and limited software inventory is appropriate for specialized, high-security use cases. As in the previous use case, the MDM product is used to establish such policies on mobile devices prior to issuance to users. The device may not be permitted connectivity to any external peripherals. It may only be able to communicate via its WiFi or cellular radios with the Enterprise-run network, which may not even permit connectivity to the Internet. Use of the device may require compliance with usage policies that would not be considered realistic in any general-purpose use case, yet may mitigate risks to highly sensitive information.

- **Personally-owned device for personal and enterprise use**

  A personally-owned device which is used for both personal activities and Enterprise data is commonly called Bring Your Own Device (BYOD). Unlike in the Enterprise-owned cases, in this scenario the Enterprise is limited in what security policies it can push to the device because the user purchased the device primarily for personal use and is unlikely to accept policies that limit the functionality of the device. However, because the Enterprise allows the user full (or nearly full) access to the Enterprise network, the Enterprise will require certain security policies, for example a password or screenlock policy, and health reporting, such as the integrity of the mobile device system software, before allowing access. The administrator of the MDM can establish remediation actions, such as wipe of the Enterprise data, for non-compliant devices.

2. SECURITY PROBLEM DESCRIPTION

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.

2.1 Threats

2.1.1 Malicious and Flawed Applications

Malicious or flawed application (app) threats exist because apps loaded onto a Mobile Device may include malicious or exploitable code. This code could be included unwittingly by its developer, perhaps as part of a software library. Malicious apps may attempt to exfiltrate data to which they have access. Malicious apps may be able to control the device's sensors (geolocation, camera, microphone, etc.) to gather intelligence about the user's surroundings even when those activities do not involve data resident or transmitted from the device. Flawed apps may give an attacker access to perform network-based or physical attacks that otherwise would have been prevented.

[T.MALICIOUS_APPS]

2.1.2 Network Attack

An attacker may position themselves on a wireless communications channel or elsewhere on the network infrastructure. From this vantage point, an attacker may initiate communication with the mobile device or alter communication between elements of the operating environment and other endpoints. By altering this communication, the attacker may be able to spoof endpoints such as the MDM Agent or MDM Server.

[T.NETWORK_ATTACK]

2.1.3 Network Eavesdropping

In a similar manner to the network attack threat, an attacker may position themselves on a wireless communications channel or elsewhere on the network infrastructure. The attacker may then monitor or gain access to data exchanged between elements of the TOE. By monitoring this data, the attacker may intercept security critical data including cryptographic keys and human-user authentication data.

[T.NETWORK_EAVESDROP]

2.1.4 Physical Access

Loss or theft of the underlying mobile device platform may give rise to loss of confidentiality of user data, including, most importantly, credentials. Physical access attacks involve attempts to access the device through external hardware ports, through its user interface, or through direct and possible destructive access to its storage media. Such attacks are intended to gain access to data from a lost or stolen mobile device that it is not expected to be returned to its owner. Although these attacks are primarily directed against the mobile device platform, the TOE provides features which address this threats.

[T.PHYSICAL_ACCESS]

2.2 Assumptions

The assumptions for the MDM are defined in Annex A.1 Assumptions.

2.3 Organization Security Policy

The organization security policies for the MDM are defined in Annex A.3 Organizational Security Policies.

3. SECURITY OBJECTIVES

3.1 Security Objectives for the TOE

Compliant TOEs will provide security functionality to address threats to it and to implement policies that address additional threats to the enterprise from inclusion of mobile devices. The following sections provide a description of this functionality in light of the threats previously discussed that motivate its inclusion in compliant TOEs. The security functionality provided includes protected communications to and between elements of the TOE, administrative access to the MDM Server, configuration of security policies for mobile devices, and system reporting for detection of security relevant events.

3.1.1 Protected Communications

To address the issues concerning transmitting sensitive data to and between the TOE (e.g., MDM Server to MDM Agent communications and remote administration to the MDM Server) described in Section 2.1.3, compliant TOEs will use a trusted communication path. The trusted channel between the MDM Server and MDM Agent is implemented using one (or more) of these standard protocols: IPsec, DTLS, or TLS. If remote administration to the MDM Server is provided, it is implemented using one or more of these standard protocols: IPsec or TLS/HTTPS.

To address the threat of network attacks described in Section 2.1.2, the protocols described in this document provide encryption and mutual authentication of each endpoint in a cryptographically secure manner; thus, any attempt by a malicious attacker to represent himself to either endpoint of the communications path as the other party would be detected.

O.DATA_PROTECTION_TRANSIT -> (FCS_CKM.1, FCS_CKM_EXT.2(*), FCS_CKM_EXT.4, FCS_COP.1.(*), FCS_DTLS_EXT.1, FCS_HTTPS_EXT.1, FCS_IPSEC_EXT.1, FCS_IV_EXT.1, FCS_RBG_EXT.1(*), FCS_STG_EXT.1, FCS_TLS_EXT.1, FIA_X509_EXT.1, FIA_X509_EXT.2, FPT_ITT.1, FTP_TRP.1, FTP_TRP.2)

3.1.2 System Reporting

To ensure that information exists that allows administrators to discover unintentional issues with the configuration and operation of the system, compliant TOEs have the capability of generating reports which may indicate such issues. Auditing of administrative activities provides information that may hasten corrective action.

O.ACCOUNTABILITY -> (FAU_ALT_EXT.1, FAU_ALT_EXT.2, FAU_GEN.1(*), FAU_SAR.1, FAU_SEL.1, FAU_STG_EXT.1, FAU_STG_EXT.2)

3.1.3 Mobile Device Configuration

Mobile devices can accept security policies defined by the Enterprise in order to ensure protection of enterprise data that they may store or process. The MDM Server is responsible for configuring and sending these policies to MDM Agents on devices, sending commands to the MDM Agents, and collecting reports from devices. The MDM Agent, in turn, is responsible for interacting with the mobile device platform to establish policies and execute commands from the MDM Server, and sending reports to the MDM Server.

O.APPLY_POLICY -> (FIA_ENR_EXT.1, FIA_X509_EXT.1, FIA_X509_EXT.2, FMT_POL_EXT.1, FMT_SMF.1(1), FMT_SMF.1(2))

3.1.4 Administration of Management Capability

To ensure the MDM Server software is operated only by authorized parties, it provides access controls around its management functionality. This includes authentication prior to administrative actions, as well as protections for its own integrity.

O.MANAGEMENT -> (FIA_UAU.1, FIA_X509_EXT.1, FIA_X509_EXT.2, FMT_MOF.1(*), FMT_SMF.1(3), FMT_SMR.1, FPT_TST_EXT.1, FPT_TUD_EXT.1)

3.2 Security Objectives for the Operational Environment

The objectives that are required to be met by the TOE's operational environment are defined in Annex A.5 Security Objectives for the Operational Environment.

4. SECURITY REQUIREMENTS

The Security Functional Requirements included in this section are derived from Part 2 of the *Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4*, with additional extended functional components.

4.1 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 <u>underlined</u> text;
- Assignment within a Selection: Indicated with <u>*italicized and underlined*</u> text;
- Iteration: Indicated by appending the iteration number in parenthesis, e.g., (1), (2), (3).

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

4.2 TOE Security Functional Requirements

This section identifies the SFRs for the TOE.

Security Audit (FAU)

FAU_ALT_EXT.1 Extended: Agent Alerts

FAU_ALT_EXT.1.1 The MDM Agent shall provide a alert via the trusted channel to the MDM Server in the event of any of the following:

a. successful application of policies to a mobile device;

b. [selection: <u>change in enrollment state, [assignment: *other events*], no other events</u>]

FAU_ALT_EXT.1.2 The MDM Server shall provide the ability to query for Agent network connectivity status.

Application Note:

The trusted channel is defined in FPT_ITT.1. "Alert" in this requirement could be as simple as an audit record or a notification.

This requirement is to ensure that the MDM Agent shall notify the MDM Server whenever one of the events listed above occurs. Lack of receipt of a successful policy installation indicates the failure of the *policy installation. It also ensures that either the MDM Server can poll devices to determine their network reachability, or the MDM Agent can be configured to regularly notify the Server that it is reachable.*

The ST author must either assign further events or select the "no other events" option. Note that alerts may take time to reach the MDM Server, or not arrive, due to poor connectivity.

Assurance Activity:

The evaluator shall examine the TSS and verify that it describes how the alert is implemented and under what circumstances, if any, the alert may not be generated (e.g., the device is powered off or disconnected from the trusted channel). The evaluator shall perform the following tests:

- *Test 1: The evaluator shall perform each of the actions listed in FAU_ALT_EXT.1.1 and verify that the alert does in fact reach the MDM Server.*
- *Test 2: The evaluator shall configure the MDM Server or MDM Agent to perform a network reachability test, both with and without such connectivity and ensure that results reflect each.*
- *Test 3: The evaluator shall confirm that events which cannot reach the MDM Agent due to connectivity issues are queued rather than discarded. When connectivity is restored the queued events should be resent.*

FAU_ALT_EXT.2 Extended: Server Alerts

FAU_ALT_EXT.2.1 The MDM Server shall alert the administrators in the event of any of the following:

a. change in enrollment status;

b. failure to apply policies to a mobile device;

c. [selection: [assignment: *other events*], no other events]

Application Note:

The MDM Agent is required to report to the MDM Server the failure of installing the policy in FMT_POL_EXT.1. This requirement is intended to ensure that the MDM Server notifies administrators when policies are not properly installed. Failure to properly install policy updates does not affect the enrollment status of the mobile device.

Assurance Activity:

The evaluator shall examine the TSS and verify that it describes how the alert system is implemented. The evaluator shall configure policies which the MDM agent should not be able to apply. These policies shall include:

- *a setting which is configurable on the MDM Server interface but not supported by the platform on which the MDM Agent runs, if any such settings exist*
- *an invalid configuration setting, which may require manual modification of the policy prior to transmission to the device*
- *a valid configuration setting with an invalid parameter, which may also require manual modification of the policy prior to transmission to the device*

The evaluator shall deploy such policies and verify that the MDM server is alerted about their failed application.

FAU_GEN.1(1) Audit Data Generation (MDM Server)

FAU_GEN.1.1(1) **Refinement:** The **MDM Server** shall be able to generate an **MDM Server** audit record of the following auditable events:

a. Start-up and shutdown of the MDM Server software;
b. All administrative actions;
c. Commands issued from the MDM Server to an MDM Agent;
d. Specifically defined auditable events listed in Table 7; and
e. [assignment: *other events*].

Application Note:

This requirement outlines the information to be included in audit records generated by the MDM Server software. These audit records may be written by the MDM Server software or may be dispatched to the operating system on which it runs. The ST author can include other auditable events in the assignment; they are not limited to the list presented. All audits must contain at least the information mentioned in FAU_GEN.1.2(1), but may contain more information which can be assigned.

Item b above requires all administrative actions to be auditable, so no additional specification for the auditability of these actions is specified in Table 7 *aside from those that require additional record content. Administrative actions refer to any management functions specified by FMT_MOF.1(1).*

Item c includes those commands which may be performed automatically based on triggers or on a schedule.

Assurance Activity:

The evaluator shall check the TSS and ensure that it lists all of the auditable events. The evaluator shall check to make sure that every audit event type mandated by the PP is described in the TSS. The evaluator shall configure the TOE to create audit records for all required audit events.

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.

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

Identification and Authentication (FIA)

FIA_ENR_EXT.1 Extended: Enrollment of Mobile Device into Management

FIA_ENR_EXT.1.1 The MDM Server shall authenticate the remote user over a trusted channel during the enrollment of a mobile device.

Application Note:

The MDM Server may use its own directory or a directory server to perform the authentication decision for users performing the remote enrollment of a mobile device.

Assurance Activity:

The evaluator shall examine the TSS and verify that it describes the process of enrollment. This description shall include the trusted path used for enrollment (FTP_TRP.2), the method of user authentication (username/password, token, etc.), the method of authentication decision (local or remote authentication services), the method by which the DN of the MDM Server is recorded, and the actions performed on the Mobile Device and on the MDM Server upon successful authentication. The evaluator shall perform the following tests:

- *Test 1: The evaluator shall attempt to enroll the device without providing correct credentials. The evaluator shall verify that the device is not enrolled and that the described enrollment actions are not taken.*
- *Test 2: The evaluator shall attempt to enroll the device providing correct credentials. The evaluator shall verify that the device is enrolled and that the described enrollment actions are taken.*

FIA_ENR_EXT.1.2 The MDM Server shall limit the user's enrollment of devices to [selection: specific devices, specific device models, a number of devices, specific time period].

Application Note:

This requirement is designed to permit the enterprise to restrict users' enrollment of devices.

Assurance Activity:

The evaluator shall examine the TSS and verify that it implements a policy to limit the users enrollment of devices. For each type of policy selected, the evaluator shall perform the following test:

- *Test 1: The evaluator shall attempt to configure the MDM Server according to the administrative guidance in order to prevent enrollment. The evaluator shall verify that the user cannot enroll a device outside of the configured limitation. (For example, the evaluator may try to enroll a disallowed device, or may try to enroll additional devices beyond the number allowed.)*

FIA_ENR_EXT.1.3 The MDM Agent shall record the DN of the MDM Server during the enrollment process.

Application Note:

The DN of the MDM Server may be the Domain Name or the IP address of the MDM Server. This requirement allows the specification of the DN to be used to establish a network connection and the expected DN for the trusted channel between the MDM Server and MDM Agent (FPT_ITT.1). The ST author shall describe in the TSS the way in which the DN is specified (preconfigured in the MDM Agent, by the user, by the MDM server, in a policy, etc.) and shall provide operational guidance as appropriate.

Assurance Activity:

The evaluator shall examine the TSS and the operational guidance to verify that it describes how the DN is specified and recorded and to verify that the MDM Agent is configured with the Domain Name or IP address of the MDM Server. The evaluator shall follow the operational guidance to establish the expected DN of the MDM server on the MDM Agent and in conjunction with other Assurance Activities verify that the MDM Agent can connect to the MDM Server and validate the MDM Server's certificate.

Security Management (FMT)

FMT_MOF.1(1) Management of functions in MDM Server

FMT_MOF.1.1(1) **Refinement:** The **MDM Server** shall restrict the ability to perform the functions

- listed in FMT_SMF.1(1)
- enable, disable, and modify policies listed in FMT_SMF.1(1)
- listed in FMT_SMF.1(3)

to *Authorized Administrators*.

Application Note:

This requirement outlines the functions that administrators will have the power to enable, disable, modify, and monitor functions and policies listed in FMT_SMF.1(1). It also includes functions necessary to maintain and configure the MDM Server itself.

Assurance Activity:

The evaluator shall examine the TSS and user documents to ensure that it describes what security management functions are restricted to the administrator and what actions can be taken for each management function. The evaluator shall verify that the security management functions are restricted to authorized administrators and the administrator is only able to take the actions as described in the user documents.

Test: The evaluator shall attempt to access the functions and policies in FMT_SMF.1(1) as an unauthorized user and verify that the attempt fails.

FMT_MOF.1(2) Management of Enrollment function

FMT_MOF.1.1(2) **Refinement:** The **MDM Server** shall restrict the ability to initiate the enrollment process to *Authorized Administrators and MD users*.

Application Note:

This requirement outlines the enrollment functions that both administrators and MD users may perform. The enrollment actions are identified in the TSS as a part of FIA_ENR_EXT.1.

Assurance Activity:

The evaluator shall examine the TSS and verify that it describes how unauthorized users are prevented from enrolling in the MDM services. The test of this function is performed in conjunction with FIA_ENR_EXT.1.

FMT_POL_EXT.1 Extended: Trusted Policy Update (MDM Agent)

FMT_POL_EXT.1.1 The MDM Agent shall report the successful installation of each policy update to the MDM Server.

Assurance Activity:

The evaluator ensures that the TSS (or the operational guidance) describes how the candidate policy updates are obtained; and the actions that take place for successful (policy update installed) and unsuccessful (policy update not installed) cases. The software components that are performing the *processing must also be identified in the TSS and verified by the evaluators. The evaluators shall perform the following test:*

- *Test 1: The evaluator shall perform a policy update in accordance with FMT_SMF.1(1). The evaluator shall verify the MDM Server provides the update to the MDM Agent. The evaluator shall verify the MDM Agent accepts the update, makes the configured changes, and reports the success of the policy update back to the MDM Server.*

FMT_SMF.1(1) Specification of management functions (Server configuration of Agent)

FMT_SMF.1.1(1) **Refinement:** The **MDM Server** shall be capable of **communicating the** following **commands to the MDM Agent**:

1. transition to the locked state,
2. full wipe of protected data,
3. unenroll from management,
4. install policies,
5. query connectivity status,
6. query the current version of the MD firmware/software
7. query the current version of the hardware model of the device
8. query the current version of installed mobile applications
9. import X.509v3 certificates into the Trust Anchor Database,
10. remove administrator-imported X.509v3 certificates and [selection: no other X.509v3 certificates, [assignment: list of other categories of X.509v3 certificates]] in the Trust Anchor Database,

and the following commands to the MDM Agent: [selection:

11. wipe sensitive data,
12. alert the administrator,
13. remove Enterprise applications,
14. import keys/secrets into the secure key storage,
15. remove imported keys/secrets and [selection: no other keys/secrets, [assignment: list of other categories of keys/secrets]] in the secure key storage,
16. remove applications,
17. update system software,
18. install applications,
19. read audit logs kept by the MD,
20. [assignment: list of other management functions to be provided by the MD], no other management functions]

and the following MD configuration policies:

21. password policy:
    a. minimum password length
    b. minimum password complexity
    c. maximum password lifetime
22. session locking policy:
    a. screen-lock enabled/disabled
    b. screen lock timeout

        c. number of authentication failures
23. wireless networks (SSIDs) to which the MD may connect
24. security policy for each wireless network:
    a. [selection: specify the CA(s) from which the MD will accept WLAN authentication server certificate(s), specify the FQDN(s) of acceptable WLAN authentication server certificate(s)]
    b. ability to specify security type
    c. ability to specify authentication protocol
    d. specify the client credentials to be used for authentication
    e. [assignment: any additional WLAN management functions]
25. application installation policy by [selection:
    a. specifying authorized application repository(s),
    b. specifying a set of allowed applications and versions (an application whitelist)
    c. denying application installation],
26. enable/disable policy for [assignment: list of audio or visual collection devices],

and the following MD configuration policies: [selection:

27. enable/disable policy for the VPN protection,
28. enable/disable policy for [assignment: list of radios],
29. enable/disable policy for data transfer capabilities over [assignment: list of externally accessible hardware ports],
30. enable/disable policy for [assignment: list of protocols where the device acts as a server],
31. enable/disable policy for developer modes,
32. enable policy for data-at rest protection,
33. enable policy for removable media's data-at-rest protection,
34. enable/disable policy for local authentication bypass,
35. the Access Point Name and proxy used for communications between the cellular network and other networks
36. the Bluetooth trusted channel policy:
    a. disable the Discoverable mode
    b. disallow Bluetooth connections using versions 1.0, 1.1, 1.2, 2.0, and [assignment: other Bluetooth version numbers]
    c. [selection: restrict Bluetooth profiles, disable legacy pairing and JustWorks pairing, and [selection: [assignment: other pairing methods], no other pairing methods]],
37. enable/disable policy for display notification in the locked state of: [selection:
    a. email notifications,
    b. calendar appointments,
    c. contact associated with phone call notification,
    d. text message notification,
    e. other application-based notifications,
    f. none]
38. policy for establishing a trusted channel or disallowing establishment if the MD cannot establish a connection to determine the validity of a certificate,
39. enable/disable policy for cellular voice functionality,
40. enable/disable policy for device messaging capabilities,

41. enable/disable policy for the cellular protocols used to connect to cellular network base stations,
42. enable/disable policy for voice command control of device functions,
43. policy for import and removal by applications of X.509v3 certificates in the Trust Anchor Database,
44. [selection: certificate, public-key] used to validate digital signature on applications,
45. policy for exceptions for shared use of keys/secrets by multiple applications,
46. policy for exceptions for destruction of keys/secrets by applications that did not import the key/secret,
47. the unlock banner policy,
48. [assignment: list of other policies to be provided by the MD], no other policies

].

Application Note:

This requirement captures all the configuration functionality the MDM Server provides the administrator to configure the MDM Agent. This requirement is broken into two configurable areas: MDM Agent commands and MDM Agent policies. The ST author can add more commands and configuration policies by completing the appropriate assignment statements

The ST author shall not claim any functionality not provided by the Mobile Device. All selections and assignments performed by the ST author in this requirement should match the selections and assignments of the validated Mobile Device ST. However, the MDM developer may choose not to implement management of optional functionality/policies in this Protection Profile even though the Mobile Device supports the functionality according to its ST.

In the future, function 10 may require destruction of any default trusted CA certificates, excepting those CA certificates necessary for continued operation of the TSF, such as the developer's certificate. At this time, the ST author shall indicate in the assignment whether pre-installed or any other category of X.509v3 certificates may be removed from the Trust Anchor Database.

The security policy number 24 addresses security types, such as WPA2-Enterprise, and authentication protocols, such as EAP-TLS. The CA or FQDN is specified for comparison by the MD.

The assignment in policy number 26 consists of all audio and visual devices, such as camera and microphone, which can be enabled and disabled by the MDM Agent.

The assignment in policy number 29 consists of all radios, such as Wi-Fi, GPS, cellular, NFC, and Bluetooth, which can be enabled and disabled by the MDM Agent.

The assignment in policy number 30 consists of all externally accessible hardware ports, such as USB, the SD card, and HDMI, whose data transfer capabilities can be enabled and disabled by the MDM Agent.

The assignment in policy number 31 consists of all protocols where the TSF acts as a server, such as WiFi tethering, which can be enabled and disabled by the MDM Agent.

Policy number 39 includes the ability to disable voice calls completely (except emergency dialing).

Policy number 40 includes the ability to disable device messaging completely (except any carrier-required and emergency SMS). Device messaging capabilities include SMS, MMS, and voicemail.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it describes each management function listed. The evaluator shall examine the TSS to verify that any differences between management functions and policies for each supported Mobile Device are listed. The evaluator shall also examine the ST of the claimed Mobile Device to verify that the selections and assignments in the functions and policies in the TSS do not exceed the capabilities of the supported MD.

The evaluator shall verify the AGD guidance includes detailed instructions of what options are available and how to configure each MDM Agent functional capability listed.

The evaluator shall verify the ability to command each MDM Agent functional capability and configure each MDM Agent policy listed above.

FMT_SMF.1(2) Specification of management functions (Agent configuration of platform)

FMT_SMF. 1(2) **Refinement:** The **MDM Agent** shall be capable of **interacting with the platform to perform** the following functions:

a. *Perform the functions listed in FMT_SMF.1(1) and the MDM configuration policies listed in FMT_SMF.1(1)*

b. *Configure the certificate to be used for authentication of MDM Agent communications*

c. *[assignment: additional functions].*

Application Note:

This requirement captures all the configuration functionality in the MDM Agent to configure the underlying Mobile Device with the configuration policies sent from the MDM Server to the Agent. The ST author can add more commands and configuration policies by completing the assignment statement; these additional commands or configuration policies must be supported by the Mobile Device.

The agent must configure the platform based on the commands and configuration policies received from the MDM Server. The possible commands and configuration policies are defined in FMT_SMF.1(1).

Assurance Activity:

The evaluator shall examine the TSS to ensure that it describes each management function listed.

The evaluator shall verify the AGD guidance includes detailed instructions of what options are available on the server. The test of function a is performed in conjunction with testing FMT_SMF.1(1).

The evaluator shall perform the following test:

- *Test: The evaluator shall configure the MDM Agent authentication certificate in accordance with the configuration guidance. The evaluator shall verify that the MDM Agent uses this certificate in performing the tests for FPT_ITT.1.*

FMT_SMF.1(3) Specification of management functions (Server Configuration of Server)

FMT_SMF.1.1(3) **Refinement:** The **MDM Server** shall be capable of **performing** the following management functions:

a) configure X.509v3 certificates for MDM Server use

b) configure the [selection: specific devices, specific device models, a number of devices, specific time period] allowed for enrollment
c) [assignment: additional functions required to support SFRs],
d) [selection: allow the administrator to choose whether to accept the certificate when a connection cannot be made to establish validity, no other management functions].

Application Note:

This requirement captures all the configuration functionality in the MDM Server to configure the underlying MDM Server. The ST author can add more commands and configuration policies by completing the assignment statement. The selection in b corresponds to the selection in FIA_ENR_EXT.1.2. The selection in d includes a function that corresponds to the selection in FIA_X509_EXT.2.2.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it describes each management function listed.

The evaluator shall verify the AGD guidance includes detailed instructions of what options are available on the server. The test of functions b and c are performed in conjunction with the use of the function. The evaluator shall perform the following test:

Test 1: The evaluator shall configure the MDM Server authentication certificate(s) and verify that the certificate is used in established trusted connections (FPT_ITT.1, FTP_TRP.1, FTP_TRP.2).

FMT_SMR.1 Security management roles

FMT_SMR.1.1 **Refinement:** The **MDM Server** shall maintain the roles *administrator, MD user, and [assignment: additional authorized identified roles]*.

FMT_SMR.1.2 **Refinement:** The **MDM Server** shall be able to associate users with roles.

Application Note:

It is envisioned that the MDM Server will be configured and maintained by different user roles. The assignment is used by the ST author to list the roles that are supported. At a minimum, one administrative role shall be supported. If not additional roles are supported, then "no additional roles" is stated. The MD user role is used for enrollment of mobile devices to the MDM according to FIA_ENR_EXT.1.

Assurance Activity:

The evaluator shall examine the TSS and user documents 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 and which interfaces are supported. 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 or TLS/HTTPS then both methods of administration must be exercised during the evaluation team's test activities.

Protection of the TSF (FPT)

FPT_ITT.1 Basic internal TSF data transfer protection

FPT_ITT.1.1 **Refinement:** The MDM Agent and MDM Server shall protect **all data** from <u>disclosure and modification</u> **through use of [selection: IPsec, TLS, DTLS]** when it is transferred between **the MDM Agent and MDM Server.**

Application Note:

This requirement is to ensure that the transmission of any audit logs, mobile device information data (software version, hardware model, and application versions), and configuration data collected by the MDM Agent and sent from the MDM Agent to the MDM Server, when commanded, or at configurable intervals, is properly protected. This trusted channel also protects any commands and policies sent by the MDM Server to the MDM Agent. Either the MDM Agent or the MDM Server is able to initiate the connection.

The trusted channel uses secure protocols that preserve the confidentiality and integrity of MDM communications. The ST author chooses the mechanism or mechanisms supported by the TOE, and then ensures the detailed requirements in Annex C corresponding to their selection are copied to the ST if not already present.

Protocol, RBG, Certificate validation, algorithm, and similar services may be met with platform provided services.

Assurance Activity:

The evaluator shall examine the TSS to determine that the methods of Agent-Server communication 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 configuring the communication channel 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) Agent-Server communication 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: The evaluator shall ensure, for each method of Agent-Server communication, the channel data is not sent in plaintext.*
- *Test 3: The evaluator shall ensure, for each method of Agent-Server communication, modification of the channel data is detected by the TOE.*

Further assurance activities are associated with the specific protocols.

FPT_TUD_EXT.1(1) Extended: Trusted Update (MDM Server)

FPT_TUD_EXT.1.1(1) The MDM Server shall provide Authorized Administrators the ability to query the current version of the MDM Server software.

Assurance Activity:

The evaluator shall perform the following test:

- *Test 1: The evaluator shall query the MDM Server for the current version of the software according to the AGD guidance and shall verify that the current version matches that of the documented and installed version.*

4.3 MDM Server or Platform Security Functional Requirements

This section identifies the SFRs that must be performed by the MDM Server or by the MDM Server's platform. Each requirement includes a selection for the ST author to indicate whether the MDM Server or the MDM Server's platform performs the functionality in the requirement. The assurance activity for those requirements for which the platform has been selected is to verify that the platforms identified by the ST author are Common Criteria validated and to ensure that the ST for the platform includes the functionality in the requirement.

Security Audit (FAU)

FAU_GEN.1(1) Audit Data Generation (MDM Server)

FAU_GEN.1.2(1) **Refinement:** The [selection: *MDM Server , MDM Server platform*] shall record within each **MDM Server** audit record at least the following information:

- date and time of the event,
- type of event,
- subject identity**,**
- (if relevant) the outcome (success or failure) of the event,
- additional information in Table 7,
- [assignment: *other audit relevant information*].

Application Note:

All audits must contain at least the information mentioned in FAU_GEN.1.2(1), but may contain more information which can be assigned. The ST author shall identify in the TSS which information of the audit record that is performed by the MDM Server and that which is performed by the MDM Server's platform.

Assurance Activity:

The evaluator shall check the TSS and ensure that it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field.

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. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

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

FAU_STG_EXT.1 Extended: External Audit Trail Storage

FAU_STG_EXT.1.1 The [selection: *MDM Server, MDM Server platform]* 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.

Application Note:

The TOE may rely on a non-TOE audit server for storage and review of audit records. Although the TOE generates audit records, the storage of these audit records and the ability to allow the administrator to review these audit records is provided by the operational environment. The MDM Server may rely on the underlying operating system for this functionality, and the first selection should be made appropriately.

In the second selection, the ST author chooses the means by which this connection is protected. The ST author also ensures that the supporting protocol requirement matching the selection is included in the ST.

Assurance Activity:

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.

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.

Cryptographic Support (FCS)

FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1(1) **Refinement:** The [selection: *MDM Server, MDM Server platform*] shall generate **asymmetric** cryptographic keys **used for key establishment** in accordance with [selection:

- *NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for finite field-based key establishment schemes;*
- *NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for elliptic curve-based key establishment schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, "Digital Signature Standard")*
- *NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography" for RSA-based key establishment schemes*]

and specified cryptographic key sizes *equivalent to, or greater than, a symmetric key strength of 112 bits*.

Application Note:

This component requires that the TOE be able to generate the public/private key pairs that are used for key establishment purposes for the various cryptographic protocols used by the TOE (e.g., trusted channel). If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection.

Since the domain parameters to be used are specified by the requirements of the protocol in this PP, it is not expected that the TOE will generate domain parameters, and therefore there is no additional domain parameter validation needed when the TOE complies with the protocols specified in this PP.

The generated key strength of 2048-bit DSA and RSA keys need to be equivalent to, or greater than, a symmetric key strength of 112 bits. See NIST Special Publication 800-57, "Recommendation for Key Management" for information about equivalent key strengths.

In the future, NIST SP 800-56A for elliptic curves will be required.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the key establishment claimed in that platform's ST contains the key establishment requirement in the MDM Server's ST. The evaluator shall also examine the TSS of the MDM Server's ST to verify that it describes (for each supported platform) how the key establishment functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the MDM Server; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Server

This assurance activity will verify the key generation and key establishments schemes used on the TOE.

Key Generation:

The evaluator shall verify the implementation of the key generation routines of the supported schemes using the applicable tests below.

Key Generation for RSA-Based Key Establishment Schemes

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.

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*

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

Key Generation for Finite-Field Cryptography (FFC) – Based 56A Schemes

<u>FFC Domain Parameter and Key Generation Tests</u>

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

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

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

and two ways to generate the cryptographic group generator g:

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

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

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

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

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

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

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

for each FFC parameter set and key pair.

Key Generation for Elliptic Curve Cryptography (ECC)- Based 56A Schemes

ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-284 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.

ECC Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-284 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 Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static 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.

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

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

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

<u>Validity Test</u>

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

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator 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).

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

At this time, detailed test procedures for RSA-based key establishment schemes are not available. In order to show that the TSF complies with 800-56A and/or 800-56B, depending on the selections made, the evaluator shall ensure that the TSS contains the following information:

- *The TSS shall list all sections of the appropriate 800-56 standard(s) to which the TOE complies.*
- *For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not"), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE.*

For each applicable section of 800-56A and 800-56B (as selected), any omission of functionality related to "shall" or "should" statements shall be described.

FCS_CKM.1.1(2) The [selection: *MDM Server, MDM Server platform*] shall generate asymmetric cryptographic keys used for authentication in accordance with a specified cryptographic key generation algorithm [selection:

- *FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 for RSA schemes;*
- *FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4 for ECDSA schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curves];*
- *ANSI X9.31-1998, Appendix A.2.4 Using AES for RSA schemes]*

and specified cryptographic key sizes [***equivalent to, or greater than, a symmetric key strength of 112 bits***].

Application Note:

While it is expected that the public key generated be associated with an identity in an X509v3 certificate, this association is not required to be performed by the TOE, and instead is expected to be performed by a Certificate Authority in the Operational Environment.

For elliptic curve-based schemes, the key size refers to the log_2 of the order of the base point.

The ANSI X9.31-1998 option will be removed from the selection in a future publication of this document. Presently, the selection is not exclusively limited to the FIPS PUB 186-4 options in order to allow industry some further time to complete the transition to the modern FIPS PUB 186-4 standard. As the preferred approach for cryptographic signature, elliptic curves will be required in future publications of this PP.

See NIST Special Publication 800-57, "Recommendation for Key Management" for information about equivalent key strengths.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the key generation function claimed in that platform's ST contains the key generation requirement in the MDM Server's ST. The evaluator shall also examine the TSS of the MDM Server's ST to verify that it describes (for each supported platform) how the key generation functionality is invoked (it should be *noted that this may be through a mechanism that is not implemented by the MDM Server; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).*

Requirement met by the MDM Server

If the TSF implements a FIPS 186-4 signature scheme, this requirement is verified under FCS_COP.1.1(1)).

If the ESF implements the ANSI X9.31-1998 scheme, the evaluator shall check to ensure that the TSS describes how the key pairs are generated. In order to show that the TSF implementation complies with ANSI X9.31-1998, the evaluator shall ensure that the TSS contains the following information:

- *The TSS shall list all sections of the standard to which the TOE complies;*
- *For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not"), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;*
- *For each applicable section of Appendix B, any omission of functionality related to "shall" or "should" statements shall be described.*

FCS_CKM_EXT.2(1) Cryptographic Key Storage (MDM Server)

FCS_CKM_EXT.2.1(1) The [selection: MDM Server, MDM Server platform] shall store persistent secrets and private keys when not in use, in [selection: platform-provided key storage, as specified in FCS_STG_EXT.1].

Application Note:

This requirement ensures that persistent secrets (credentials, secret keys) and private keys are stored securely when not in use. If some secrets/keys are manipulated by the TOE and others are manipulated by the platform, then both of the selections can be specified by the ST author and the ST author must identify in the TSS those keys which are manipulated by the TOE and those by the platform.

If the MDM Server is an application, and not a dedicated server, then it should store its private keys in the platform-provided key storage.

The ST author is responsible for selecting the manner in which the keys are stored and where they are stored in the selections above.

Assurance Activity:

Regardless of whether this requirement is met by the TOE or the TOE platform, the evaluator will check the TSS to ensure that it lists each persistent secret (credential, secret key) and private key needed to meet the requirements in the ST. For each of these items, the evaluator will confirm that the TSS lists for what purpose it is used, and how it is stored. The evaluator than performs the following actions.

Persistent secrets and private keys manipulated by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the persistent secrets and private keys listed as being stored by the platform in the MDM Server ST are identified as being protected in that platform's ST.

Persistent secrets and private keys manipulated by the TOE

The evaluator reviews the TSS for to determine that it makes a case that, for each item listed as being manipulated by the TOE, it is not written unencrypted to persistent memory, and that the item is stored by the platform.

FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1(1) The [selection: MDM Server, MDM Server platform] shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

Application Note:

The ST author should select the platform if the MDM Server platform performs no operations using plaintext secret, private cryptographic keys, and CSPs.

Any security related information (such as keys, authentication data, and passwords) must be zeroized when no longer in use to prevent the disclosure or modification of security critical data.

The zeroization indicated above applies to each intermediate storage area for plaintext key and Cryptographic Service Provider (CSP) (i.e., any storage, such as memory buffers, that is included in the path of such data) upon the transfer of the key/CSP to another location.

Since the TOE does not include the host IT environment, the extent of this capability is necessarily somewhat limited. For the purposes of this requirement, it is sufficient for the TOE to invoke the correct underlying functions of the host to perform the zeroization--it does not imply that the TOE has to include a kernel-mode memory driver to ensure the data are zeroized. It is assumed that the host platform appropriately performs zeroization of key material in its internal processes.

Assurance Activity:

Requirement met by the platform

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key that are not otherwise covered by the FCS_CKM_EXT.4 requirement levied on the TOE.

For each platform listed in the ST, the evaluator shall examine the TSS of the ST of the platform to ensure that each of the secret keys, private keys, and CSPs used to generate key listed above are covered.

Requirement met by MDM Server

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key; when they are zeroized (for example, immediately after use, on system shutdown, etc.); and the type of zeroization procedure that is performed (overwrite with zeros, overwrite three times with random pattern, etc.). 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 zeroization procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are zeroized by overwriting once with zeros, while secret keys stored on the internal hard drive are zeroized by overwriting three times with a random pattern that is changed before each write"). If a read-back is done to verify the zeroization, this shall be described as well.

For each key clearing situation described in the TSS the evaluator shall repeat the following test.

Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) for the TOE and instrumented TOE builds 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.*

Cryptographic TOE implementations in software shall be loaded and exercised under a debugger to perform such tests. The evaluator shall perform the following test for each key subject to clearing, including intermediate copies of keys that are persisted encrypted by the TOE:

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

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

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

Test 2: In cases where the TOE is implemented in firmware and operates in a limited operating environment that does not allow the use of debuggers, the evaluator shall utilize a simulator for the TOE on a general purpose operating system. The evaluator shall provide a rationale explaining the instrumentation of the simulated test environment and justifying the obtained test results.

FCS_COP.1(1) Cryptographic operation (Digital Signatures)

FCS_COP.1.1(1) **Refinement:** The [selection: MDM Server, MDM Server platform] shall perform *cryptographic signature services* in accordance with the following specified cryptographic algorithms [selection:

- ***RSA Digital Signature Algorithm (RSA) with a key size (modulus) of 2048 bits or greater that meets* FIPS PUB 186-2 or FIPS PUB 186-4, “Digital Signature Standard”,**
- ***Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater] that meets* FIPS PUB 186-4, “Digital Signature Standard” with “NIST curves” P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, “Digital Signature Standard”),**
- ***Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater that meets* FIPS PUB 186-4, “Digital Signature Standard”, no other cryptographic signature service].**

Application Note:

Both the MDM Server and MDM Agent must perform digital signatures in accordance with the protocols for FTP_ITC_EXT.1. The MDM Server may also send digitally signed policies and policy updates to the mobile device.

If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the digital signature functions claimed in that platform's ST contains the digital signature functions in the MDM Server's ST. The evaluator shall also examine the TSS of the MDM Server's ST to verify that it describes (for each supported platform) how the digital signature functionality is invoked for each operation they are used for in the MDM Server (it should be noted that this may be through a mechanism that is not implemented by the MDM Server; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Server

Key Generation:

Key Generation for RSA Signature Schemes

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.

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*

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

ECDSA Key Generation Tests

<u>FIPS 186-4 ECDSA Key Generation Test</u>

For each supported NIST curve, i.e., P-256, P-284 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

For each supported NIST curve, i.e., P-256, P-284 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.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-284 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

For each supported NIST curve (i.e., P-256, P-284 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.

RSA Signature Algorithm Tests

Signature Generation Test

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

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

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.

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.

FCS_COP.1(2) Cryptographic operation (Keyed-Hash Message Authentication)

FCS_COP.1.1(2) The [selection: MDM Server, MDM Server platform] shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-[selection: SHA-1, SHA-256, SHA-384, SHA-512], key sizes [assignment: key size (in bits) used in HMAC], and message digest sizes [selection: 160, 256, 384, 512] 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:

The intent of this requirement is to specify the keyed-hash message authentication function used when used for key establishment purposes for the various cryptographic protocols used by the TOE (e.g., trusted channel). The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(3).

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the keyed-hash function(s) claimed in that platform's ST contains the keyed-hash function(s) in the MDM Server’s ST. The evaluator shall also examine the TSS of the MDM Server’s ST to verify that it describes (for each supported platform) how the keyed-hash functionality is invoked for each mode and key size selected in the MDM Server’s ST (it should be noted that this may be through a mechanism that is not implemented by the MDM Server; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Server

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.

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.

FCS_COP.1(3) Cryptographic operation (Encryption and Decryption)

FCS_COP.1.1(3) The [selection: MDM Server, MDM Server platform] shall perform [***encryption/decryption***] in accordance with a specified cryptographic algorithm [selection:

- *AES-CBC (as defined in NIST SP 800-38A) mode,*
- *AES-GCM (as defined in NIST SP 800-38D),*
- *AES-CCM (as defined in NIST SP 800-38C,*
- *AES Key Wrap (KW) (as defined in NIST SP 800-38F),*
- *AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F)*

] and cryptographic key sizes [selection: *128-bit, 256-bit*] key sizes.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the encryption/decryption function(s) claimed in that platform's ST contains the encryption/decryption function(s) in the MDM Server’s ST. The evaluator shall also examine the TSS of the MDM Server’s ST to verify that it describes (for each supported platform) how the encryption/decryption functionality is *invoked for each mode and key size selected in the MDM Server's ST (it should be noted that this may be through a mechanism that is not implemented by the MDM Server; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).*

Requirement met by the MDM Server

AES-CBC Tests

<u>AES-CBC Known Answer Tests</u>

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.

- ***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.*

  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.

- ***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.*

  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.

- ***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].*

  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.

- ***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].*

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

AES-CBC Multi-Block Message Test

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

The evaluator shall also test the decrypt functionality 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.

AES-CBC Monte Carlo Tests

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:

```
# 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.*
- ***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.*

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

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator 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.

AES-GCM Monte Carlo Test

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

- ***128 bit and 256 bit keys***
- ***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.*
- ***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.*
- ***Two IV lengths.*** *If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.*

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length 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.

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.

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

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

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

- ***128 and 256 bit key encryption keys***
- ***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).*

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

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

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

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

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

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

FCS_COP.1(4) Cryptographic operation (Hashing)

FCS_COP.1.1(4) The [selection: MDM Server, MDM Server platform] shall perform cryptographic hashing in accordance with a specified cryptographic algorithm [selection: *SHA-1, SHA-256, SHA-384, SHA-512*] and message digest sizes [selection: *160, 256, 384, 512*] bits that meet the following: *FIPS Pub 180-4*.

Application Note:

In future versions of this document, SHA-1 may be removed as an option. SHA-1 for generating digital signatures will no longer be allowed after December 2013, and SHA-1 for verification of digital signatures is strongly discouraged as there may be risk in accepting these signatures.

The intent of this requirement is to specify the hashing function used for digital signature generation and verification associated with trusted updates and trusted channel. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1).

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the hash function(s) claimed in that platform's ST contains the hash function(s) in the MDM Server's ST. The evaluator shall also examine the TSS of the MDM Server's ST to verify that it describes (for each supported platform) how the hash functionality is invoked for each digest size selected in the MDM Server's ST (it should be noted that this may be through a mechanism that is not implemented by the MDM Server; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Server

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.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

*The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly 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.*

Pseudorandomly Generated Messages Test

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

FCS_RBG_EXT.1(1) Extended: Random Bit Generation

FCS_RBG_EXT.1.1(1) The [selection: MDM Server, MDM Server platform] shall perform all deterministic random bit generation services in accordance with [selection, *choose one of: NIST Special Publication 800-90A using* [selection: *Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES), Dual_EC_DRBG (any)]; FIPS Pub 140-2 Annex C: X9.31 Appendix 2.4 using AES*].

FCS_RBG_EXT.1.2(1) The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: *a TSF-hardware-based noise source , a TSF software-based noise source, a platform-based RBG*] with a minimum of [selection: *128 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.

Application Note:

For the first selection in FCS_RBG_EXT.1.1, the ST author should select whether the TOE or the platform on which the TOE is installed provides the RBG services.

NIST Special Pub 800-90B, Appendix C describes the minimum entropy measurement that will probably be required future versions of FIPS-140. If possible this should be used immediately and will be required in future versions of this PP.

For the second selection in FCS_RBG_EXT.1.1, the ST author should select the standard to which the RBG services comply (either 800-90A or 140-2 Annex C).

SP 800-90A contains four 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 (if 800-90A is selected), and include the specific underlying cryptographic primitives used in the requirement or in the TSS. 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 CT_DRBG are allowed. While any of the curves defined in 800-90A are allowed for Dual_EC_DRBG, the ST author not only must include the curve chosen, but also the hash algorithm used.

For the first selection in FCS_RBG_EXT.1.2, the ST author indicates whether the sources of entropy are software-based, hardware-based, platform-based, or some combination. If there are multiple sources of *entropy, the ST will describe each entropy source and whether it is hardware-, software-, or platform-based. Platform-based and hardware-based noise sources are preferred.*

The platform-based RBG source is the output of a validated RBG provided by the platform, which is used as an entropy source for a TSF-provided DRBG according to FCS_RBG_EXT.1.1. In this way, the developer has chained RBGs as described in NIST SP800-90C.

Note that for FIPS Pub 140-2 Annex C, currently only the method described in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3 is valid. 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.

The ST author also ensures that any underlying functions are included in the baseline requirements for the TOE.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the RBG functions claimed in that platform's ST contains the RBG functions in the MDM Server's ST. The evaluator shall also examine the TSS of the MDM Server's ST to verify that it describes (for each supported platform) how the RBG functionality is invoked for each operation they are used for in the MDM Server (it should be noted that this may be through a mechanism that is not implemented by the MDM Server; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Server

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with ANNEX E: ENTROPY DOCUMENTATION AND ASSESSMENT.

If the ST author has selected a platform-based noise source, the evaluator shall verify that the platform's RBG has been validated by examining the platform's ST. The evaluator shall verify that the platform's RBG is seeded with at least the amount of entropy selected by the ST author for this profile. In this case, the ST author is not responsible for Annex E documentation of the platform's RBG.

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

Implementations Conforming to FIPS 140-2, Annex C

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluator shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

The evaluator shall perform a Variable Seed Test. The evaluator shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluator shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluator ensures that the values returned by the TSF match the expected values.

The evaluator shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluator shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluator then invokes the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3. The evaluator ensures that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

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.

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).

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 reseed. The final value is additional input to the second generate call.

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

> ***Entropy input:*** *the length of the entropy input value must equal the seed length.*
>
> ***Nonce:*** *If a nonce is supported (CTR_DRBG with no 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.*

Identification and Authentication (FIA)

FIA_UAU.1 Timing of Authentication

FIA_UAU.1.1 **Refinement**: The [selection: MDM Server, MDM Server platform] shall allow [assignment: *list of* **MDM Server** *mediated actions*] on behalf of the user to be performed before the user is authenticated **with the Server.**

FIA_UAU.1.2 **Refinement**: The [selection: MDM Server, MDM Server platform] shall require each user to be successfully authenticated **with the Server** before allowing any other **MDM Server**-mediated actions on behalf of that user.

Application Note:

This requirement ensures that any user attempting to access the MDM Server must be authenticated. These users may be administrators attempting to administer the TOE or ordinary users attempting to enroll for management by the MDM system. The ST author is responsible for assigning the list of actions that can take place before this authentication. The MDM Server or MDM Server platform may utilize enterprise authentication to meet this requirement.

Assurance Activity:

The evaluator shall examine the TSS and verify that it describes the actions that can and cannot be performed before authentication. The evaluator shall perform the following tests:

- *Test 1: The evaluator shall attempt to perform the prohibited actions before authentication. The evaluator shall verify the actions cannot be performed.*
- *Test 2: The evaluator shall attempt to perform the prohibited actions after authentication. The evaluator shall verify the actions can be performed.*

FIA_X509_EXT.1(1) Extended: X509 Validation

FIA_X509_EXT.1.1(1) The [selection: *MDM Server, MDM Server platform*] shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- 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).
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.

Application Note:

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. Certificates may optionally be used for trusted updates of TSF Software (FPT_TUD_EXT.1.3) and for software integrity verification (FPT_TST_EXT.1.2) and, if implemented, must be validated to contain the Code Signing purpose extendedKeyUsage. If TLS, DTLS, or HTTPS is selected in FPT_ITT.1 or FTP_TRP.1 or FTP_TRP.2, certificates must be used to perform authentication and must be validated to contain the Client Authentication purpose extendedKeyUsage.

It should be noted that the validation is expected to end in a trusted root certificate.

While FIA_X509_EXT.1.1 requires that the TOE perform certain checks on the certificate presented by a TLS client, there are corresponding checks that the client will have to perform on the certificate presented by the client; namely that the extendedKeyUsage field of the client certificate includes "Server Authentication" and that the key agreement bit (for the Diffie-Hellman ciphersuites) or the key encipherment bit (for RSA ciphersuites) be set. Certificates obtained for use by the TOE will have to conform to these requirements in order to be used in the enterprise.

FIA_X509_EXT.1.2(1) The [selection: *MDM Server, MDM Server platform*] shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

Application Note:

This requirement applies to certificates that are used and processed by the MDM Server or platform.

Assurance Activity:

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

The 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, and then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator 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.

FIA_X509_EXT.2(1) Extended: X509 Authentication

FIA_X509_EXT.2.1(1) The [selection: *MDM Server, MDM Server platform*] shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [selection*: IPsec, TLS, HTTPS, DTLS*]], and [selection: *code signing for system software updates, code signing for integrity verification, policy signing, no additional uses*].

Application Note:

The ST author's selection shall match the selection of FPT_ITT.1 and FTP_TRP.1. Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1.3) and software integrity verification (FPT_TST_EXT.1.2). If either of the code signing uses is selected, FIA_X509_EXT.2.4 must be included in the main body. If FMT_POL_EXT.1.2 is included in the main body, policy signing must be selected. If some authentication services are provided by the TOE and others by the platform, the ST author shall clearly identify which services are provided by the TOE and which by the platform.

Each client device will have a unique X.509v3 certificate for use by the MDM Agent; the certificate is not to be reused among clients.

FIA_X509_EXT.2.2(1) When the [selection: *MDM Server, MDM Server platform*] cannot establish a connection to determine the validity of a certificate, the [selection: *MDM Server, MDM Server platform*] shall [selection: *allow the administrator to choose whether to accept the certificate in these cases, accept the certificate*, *not accept the certificate*].

Application Note:

Often a connection must be established to perform a verification of the revocation status of a certificate - either to download a CRL or to perform 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 second 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 function d in FMT_SMF.1(3).

FIA_X509_EXT.2.3(1) The [selection: *MDM Server, MDM Server platform*] shall not establish a trusted communication channel if the peer certificate is deemed invalid.

Application Note:

Trusted communication channels include any of IPsec, TLS, HTTPS, or DTLS performed by the TSF. Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

FIA_X509_EXT.2.5(1) The [selection: *MDM Server, MDM Server platform*] shall generate a Certificate Request Message as specified in RFC 2986 and be able to provide the following information in the request: public key, Common Name, Organization, Organizational Unit, and Country.

Application Note:

The public key referenced in FIA_X509_EXT.2.5 is the public key portion of the public-private key pair generated by the TOE as specified in FCS_CKM.1(2).

Assurance Activity

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

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

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 have the MDM Server generate a public-private key pair, and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE) for its signature. The values for the DN (Common Name, Organization, Organizational Unit, and Country) will also be passed in the request.

Test 3: 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.

Protection of the TSF (FPT)

FPT_TST_EXT.1(1): TSF Testing

FPT_TST_EXT.1.1(1) The [selection: MDM Server, MDM Server platform] shall run a suite of self tests during initial start-up (on power on) to demonstrate correct operation of the MDM Server.

FPT_TST_EXT.1.2(1) The [selection: MDM Server, MDM Server platform] shall provide the capability to verify the integrity of stored MDM Server executable code when it is loaded for execution through the use of the [selection: MDM Server, MDM Server platform]-provided cryptographic services.

Application Note:

While the TOE is typically a software package running in the IT Environment, it is still capable of performing the self-test activities required above. It should be understood, however, that there is a significant dependency on the host environment in assessing the assurance provided by the tests mentioned above (meaning that if the host environment is compromised, the self tests will not be meaningful).

Assurance Activity:

The evaluator shall examine the TSS to ensure that it details the self tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

The evaluator shall examine the TSS to ensure that it describes how to verify the integrity of stored TSF executable code when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. The evaluator shall perform the following tests:

- *Test 1: The evaluator performs the integrity check on a known good TSF executable and verifies that the check is successful.*
- *Test 2: The evaluator modifies the TSF executable, performs the integrity check on the modified TSF executable and verifies that the check fails.*

FPT_TUD_EXT.1(1) Extended: Trusted Update (MDM Server)

FPT_TUD_EXT.1.2(1) The [selection: MDM Server, MDM Server platform] shall provide Authorized Administrators the ability to initiate updates to MDM Server software.

FPT_TUD_EXT.1.3(1) The [selection: MDM Server, MDM Server platform] shall provide a means to verify software updates to the MDM Server using a digital signature mechanism prior to installing those updates.

Application Note:

The software on the MDM Server will occasionally need to be updated. This requirement is intended to ensure that the MDM Server only installs updates provided by the vendor, as updates provided by another source may contain malicious code. If the server is not an appliance, the update will be verified by the platform on which the server software runs. If the server is an appliance, the update must be verified by the MDM Server software or hardware.

Assurance Activity:

The evaluator shall examine the TSS and verify that it describes the standards by which the updates are digitally signed and how the signature verification process is implemented. The evaluator shall examine *the AGD guidance to verify that it describes how to query the current version of the MDM Server software and how to initiate updates. The evaluator shall perform the following tests:*

- *Test 1: The evaluator shall attempt to initiate an update digitally signed by the vendor and verify that the update is successfully installed.*
- *Test 2: The evaluator shall attempt to install an update not digitally signed by the vendor and verify that the update is not installed.*

Trusted Path/Channels (FTP)

FTP_TRP.1 Trusted path for Remote Administration

FTP_TRP.1.1 **Refinement:** The [selection: **MDM Server, MDM Server platform**] 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.*

FTP_TRP.1.2 **Refinement:** The [selection: **MDM Server, MDM Server platform**] shall permit **remote administrators** to initiate communication via the trusted path.

FTP_TRP.1.3 **Refinement:** The [selection: **MDM Server, MDM Server platform**] shall require the use of the trusted path for all remote administration actions.

Application Note:

This requirement ensures that authorized remote administrators initiate all communication with the TOE via a trusted path, and that all communications with the TOE by remote administrators is performed over this path. The data passed in this trusted communication channel are encrypted as defined the protocol chosen in the first selection. The ST author chooses the mechanism or mechanisms supported by the TOE, and then ensures the detailed requirements in Annex C corresponding to their selection are copied to the ST if not already present.

Assurance Activity:

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.*

Further assurance activities are associated with the specific protocols.

FTP_TRP.2 Trusted path for Enrollment

FTP_TRP.2.1 **Refinement:** The [selection: **MDM Server, MDM Server platform**] shall **use [selection: TLS, TLS/HTTPS] to** provide a trusted communication path between itself and **MD users** 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.*

FTP_TRP.2.2 **Refinement:** The [selection: **MDM Server, MDM Server platform**] shall permit **MD users** to initiate communication via the trusted path.

FTP_TRP.2.3 **Refinement:** The [selection: **MDM Server, MDM Server platform**] shall require the use of the trusted path for all MD user actions.

Application Note:

This requirement ensures that authorized MD users initiate all communication with the TOE via a trusted path, and that all communications with the TOE by MD users is performed over this path. The purpose of this connection is for enrollment by the MD user. The data passed in this trusted communication channel are encrypted as defined the protocol chosen in the first selection. The ST author chooses the mechanism or mechanisms supported by the TOE, and then ensures the detailed requirements in Annex C corresponding to their selection are copied to the ST if not already present.

Assurance Activity:

The evaluator shall examine the TSS to determine that the methods of remote enrollment are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of enrollment 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 enrollment 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) enrollment 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 enrollment 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 enrollment sessions without invoking the trusted path.*
- *Test 3: The evaluator shall ensure, for each method enrollment, the channel data is not sent in plaintext.*

- *Test 4: The evaluator shall ensure, for each method of enrollment, modification of the channel data is detected by the TOE.*

Further assurance activities are associated with the specific protocols.

4.4 MDM Agent or Platform Security Functional Requirements

This section identifies the SFRs that must be performed by the MDM Agent or by the MDM Agent's platform. Each requirement includes a selection for the ST author to indicate whether the MDM Agent or the MDM Agent's platform performs the functionality in the requirement. The assurance activity for those requirements for which the platform has been selected is to verify that the platforms identified by the ST author are Common Criteria validated and to ensure that the ST for the platform includes the functionality in the requirement.

Cryptographic Support (FCS)

FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1(3) **Refinement:** The [selection: MDM Agent, MDM Agent platform] shall generate **asymmetric** cryptographic keys **used for key establishment** in accordance with [selection:

- *NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for finite field-based key establishment schemes;*
- *NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for elliptic curve-based key establishment schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, "Digital Signature Standard")*
- *NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography" for RSA-based key establishment schemes*]

and specified cryptographic key sizes *equivalent to, or greater than, a symmetric key strength of 112 bits*.

Application Note:

This component requires that the TOE be able to generate the public/private key pairs that are used for key establishment purposes for the various cryptographic protocols used by the TOE (e.g., trusted channel). If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection.

Since the domain parameters to be used are specified by the requirements of the protocol in this PP, it is not expected that the TOE will generate domain parameters, and therefore there is no additional domain parameter validation needed when the TOE complies with the protocols specified in this PP.

The generated key strength of 2048-bit DSA and RSA keys need to be equivalent to, or greater than, a symmetric key strength of 112 bits. See NIST Special Publication 800-57, "Recommendation for Key Management" for information about equivalent key strengths.

In the future, NIST SP 800-56A for elliptic curves will be required.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the key establishment claimed in that platform's ST contains the key establishment requirement in the MDM Agent's ST. The evaluator shall also examine the TSS of the MDM Agent's ST to verify that it describes (for each supported platform) how the key establishment functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Agent

This assurance activity will verify the key generation and key establishments schemes used on the TOE.

Key Generation:

The evaluator shall verify the implementation of the key generation routines of the supported schemes using the applicable tests below.

Key Generation for RSA-Based Key Establishment Schemes

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.

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*

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

Key Generation for Finite-Field Cryptography (FFC) – Based 56A Schemes

<u>FFC Domain Parameter and Key Generation Tests</u>

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

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

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

and two ways to generate the cryptographic group generator g:

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

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

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

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

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

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

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

for each FFC parameter set and key pair.

Key Generation for Elliptic Curve Cryptography (ECC)- Based 56A Schemes

<u>ECC Key Generation Test</u>

For each supported NIST curve, i.e., P-256, P-284 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.*

<u>ECC Public Key Verification (PKV) Test</u>

For each supported NIST curve, i.e., P-256, P-284 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 Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

<u>Function Test</u>

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

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static 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.

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

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

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

<u>Validity Test</u>

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

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator 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).

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

At this time, detailed test procedures for RSA-based key establishment schemes are not available. In order to show that the TSF complies with 800-56A and/or 800-56B, depending on the selections made, the evaluator shall ensure that the TSS contains the following information:

- *The TSS shall list all sections of the appropriate 800-56 standard(s) to which the TOE complies.*
- *For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not"), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE.*

For each applicable section of 800-56A and 800-56B (as selected), any omission of functionality related to "shall" or "should" statements shall be described.

FCS_CKM_EXT.2(2) Cryptographic Key Storage (MDM Agent)

FCS_CKM_EXT.2.1(2) The [selection: MDM Agent, MDM Agent platform] shall store persistent secrets and private keys when not in use in platform-provided key storage.

Application Note:

This requirement ensures that persistent secrets (credentials, secret keys) and private keys are stored securely when not in use. If some secrets/keys are manipulated by the TOE and others are manipulated by the platform, then both of the selections can be specified by the ST author and the ST author must identify in the TSS those keys which are manipulated by the TOE and those by the platform.

This requirement assumes persistent secrets and private keys used by the MDM Agent will be stored by the mobile platform.

Assurance Activity:

Regardless of whether this requirement is met by the TOE or the TOE platform, the evaluator will check the TSS to ensure that it lists each persistent secret (credential, secret key) and private key needed to meet the requirements in the ST. For each of these items, the evaluator will confirm that the TSS lists for what purpose it is used, and how it is stored. The evaluator than performs the following actions.

Persistent secrets and private keys manipulated by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the persistent secrets and private keys listed as being stored by the platform in the MDM Agent ST are identified as being protected in that platform's ST.

Persistent secrets and private keys manipulated by the TOE

The evaluator reviews the TSS for to determine that it makes a case that, for each item listed as being manipulated by the TOE, it is not written unencrypted to persistent memory, and that the item is stored by the platform.

FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1(2) The [selection: MDM Agent, MDM Agent platform] shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

Application Note:

The ST author should select the platform if the MDM Agent performs no operations using plaintext secret, private cryptographic keys, and CSPs.

Any security related information (such as keys, authentication data, and passwords) must be zeroized when no longer in use to prevent the disclosure or modification of security critical data.

The zeroization indicated above applies to each intermediate storage area for plaintext key and Cryptographic Service Provider (CSP) (i.e., any storage, such as memory buffers, that is included in the path of such data) upon the transfer of the key/CSP to another location.

Since the TOE does not include the host IT environment, the extent of this capability is necessarily somewhat limited. For the purposes of this requirement, it is sufficient for the TOE to invoke the correct underlying functions of the host to perform the zeroization--it does not imply that the TOE has to include a kernel-mode memory driver to ensure the data are zeroized. It is assumed that the host platform appropriately performs zeroization of key material in its internal processes.

Assurance Activity:

Requirement met by the platform

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key that are not otherwise covered by the FCS_CKM_EXT.4 requirement levied on the TOE.

For each platform listed in the ST, the evaluator shall examine the TSS of the ST of the platform to ensure that each of the secret keys, private keys, and CSPs used to generate key listed above are covered.

Requirement met by MDM Agent

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key; when they are zeroized (for example, immediately after use, on system shutdown, etc.); and the type of zeroization procedure that is performed (overwrite with zeros, overwrite three times with random pattern, etc.). 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 zeroization procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are zeroized by overwriting once with zeros, while secret keys stored on the internal hard drive are zeroized by overwriting three times with a random pattern that is changed before each write"). If a read-back is done to verify the zeroization, this shall be described as well.

For each key clearing situation described in the TSS the evaluator shall repeat the following test.

Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) for the TOE and instrumented TOE builds 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.

Cryptographic TOE implementations in software shall be loaded and exercised under a debugger to perform such tests. The evaluator shall perform the following test for each key subject to clearing, including intermediate copies of keys that are persisted encrypted by the TOE:

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

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

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

Test 2: In cases where the TOE is implemented in firmware and operates in a limited operating environment that does not allow the use of debuggers, the evaluator shall utilize a simulator for the TOE on a general purpose operating system. The evaluator shall provide a rationale explaining the instrumentation of the simulated test environment and justifying the obtained test results.

FCS_COP.1(5) Cryptographic operation (Digital Signatures)

FCS_COP.1.1(5) **Refinement:** The [selection: MDM Agent, MDM Agent platform] shall perform *cryptographic signature services* in accordance with the following specified cryptographic algorithms [selection:

- ***RSA Digital Signature Algorithm (RSA) with a key size (modulus) of 2048 bits or greater that meets* FIPS PUB 186-2 or FIPS PUB 186-4, "Digital Signature Standard",**

- ***Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater] that meets*** **FIPS PUB 186-4, “Digital Signature Standard” with “NIST curves” P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, “Digital Signature Standard”),**
- ***Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater that meets*** **FIPS PUB 186-4, “Digital Signature Standard”, no other cryptographic signature service].**

Application Note:

Both the MDM Server and MDM Agent must perform digital signatures in accordance with the protocols for FTP_ITC_EXT.1. The MDM Server may also send digitally signed policies and policy updates to the mobile device . The MDM Agent is required to validate those signed policies.

If multiple schemes are supported, then the ST author should iterate this requirement to capture this capability. The scheme used will be chosen by the ST author from the selection.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the digital signature functions claimed in that platform's ST contains the digital signature functions in the MDM Agent's ST. The evaluator shall also examine the TSS of the MDM Agent’s ST to verify that it describes (for each supported platform) how the digital signature functionality is invoked for each operation they are used for in the MDM Agent (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Agent

Key Generation:

Key Generation for RSA Signature Schemes

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.

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*

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

ECDSA Key Generation Tests

FIPS 186-4 ECDSA Key Generation Test

For each supported NIST curve, i.e., P-256, P-284 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

For each supported NIST curve, i.e., P-256, P-284 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.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-284 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

For each supported NIST curve (i.e., P-256, P-284 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.

RSA Signature Algorithm Tests

Signature Generation Test

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

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

> *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.*

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.

FCS_COP.1(6) Cryptographic operation (Keyed-Hash Message Authentication)

FCS_COP.1.1(6) The [selection: MDM Agent, MDM Agent platform] shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-[selection: SHA-1, SHA-256, SHA-384, SHA-512], key sizes [assignment: key size (in bits) used in HMAC], and message digest sizes [selection: 160, 256, 384, 512] 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:

The intent of this requirement is to specify the keyed-hash message authentication function used when used for key establishment purposes for the various cryptographic protocols used by the TOE (e.g., trusted channel). The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(3).

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the keyed-hash function(s) claimed in that platform's ST contains the keyed-hash function(s) in the MDM Agent's ST. The evaluator shall also examine the TSS of the MDM Agent's ST to verify that it describes (for each supported platform) how the keyed-hash functionality is invoked for each mode and key size selected in the MDM Agent's ST (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Agent

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.

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.

FCS_COP.1(7) Cryptographic operation (Encryption and Decryption)

FCS_COP.1.1(7) The [selection: MDM Agent, MDM Agent platform] shall perform [***encryption/decryption***] in accordance with a specified cryptographic algorithm [selection:

- *AES-CBC (as defined in NIST SP 800-38A) mode*,
- *AES-GCM (as defined in NIST SP 800-38D)*

] and cryptographic key sizes [<u>selection:</u> *<u>128-bit, 256-bit</u>*<u>]</u> *<u>key sizes</u>*.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the encryption/decryption function(s) claimed in that platform's ST contains the encryption/decryption function(s) in the MDM Agent's ST. The evaluator shall also examine the TSS of the MDM Agent's ST to verify that it describes (for each supported platform) how the encryption/decryption functionality is invoked for each mode and key size selected in the MDM Agent's ST (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Agent

AES-CBC Tests

<u>AES-CBC Known Answer Tests</u>

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.

- ***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.*

  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.

- ***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.*

  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.

- ***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].*

  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.*

- ***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].*

  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.

<u>AES-CBC Multi-Block Message Test</u>

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.

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.

<u>AES-CBC Monte Carlo Tests</u>

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:

```
# 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.*
- ***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.*

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

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator 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.

AES-GCM Monte Carlo Test

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

- ***128 bit and 256 bit keys***

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

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length 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.

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.

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

FCS_COP.1(8) Cryptographic operation (Hashing)

FCS_COP.1.1(8) The [selection: MDM Agent, MDM Agent platform] shall perform cryptographic hashing in accordance with a specified cryptographic algorithm [selection: *SHA-1, SHA-256, SHA-384, SHA-512*] and message digest sizes [selection: *160, 256, 384, 512*] bits that meet the following: *FIPS Pub 180-4*.

Application Note:

In future versions of this document, SHA-1 may be removed as an option. SHA-1 for generating digital signatures will no longer be allowed after December 2013, and SHA-1 for verification of digital signatures is strongly discouraged as there may be risk in accepting these signatures.

The intent of this requirement is to specify the hashing function used for digital signature generation and verification associated with trusted updates and trusted channel. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(7).

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the hash function(s) claimed in that platform's ST contains the hash function(s) in the MDM Agent's ST. The evaluator shall also examine the TSS of the MDM Agent's ST to verify that it describes (for each supported platform) how the hash functionality is invoked for each digest size selected in the MDM Agent's ST (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Agent

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.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

*The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly 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.*

Pseudorandomly Generated Messages Test

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

FCS_RBG_EXT.1(2) Extended: Random Bit Generation

FCS_RBG_EXT.1.1(2) The [selection: MDM Agent, MDM Agent platform] shall perform all deterministic random bit generation services in accordance with [selection, *choose one of: NIST Special Publication 800-90A using* [selection: *Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES), Dual_EC_DRBG (any)]; FIPS Pub 140-2 Annex C: X9.31 Appendix 2.4 using AES*].

FCS_RBG_EXT.1.2(2) The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: *a software-based noise source, a platform-based RBG*] with a minimum of [selection: *128 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.

Application Note:

For the first selection in FCS_RBG_EXT.1.1, the ST author should select whether the TOE or the platform on which the TOE is installed provides the RBG services. If the MDM Server and MDM Agent act differently, the ST author should select both.

NIST Special Pub 800-90B, Appendix C describes the minimum entropy measurement that will probably be required future versions of FIPS-140. If possible this should be used immediately and will be required in future versions of this PP.

For the second selection in FCS_RBG_EXT.1.1, the ST author should select the standard to which the RBG services comply (either 800-90 or 140-2 Annex C).

SP 800-90A contains four 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 (if 800-90A is selected), and include the specific underlying cryptographic primitives used in the requirement or in the TSS. 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 CT_DRBG are allowed. While any of the curves defined in 800-90A are allowed for Dual_EC_DRBG, the ST author not only must include the curve chosen, but also the hash algorithm used.

For the first selection in FCS_RBG_EXT.1.2, the ST author indicates whether the sources of entropy are software-based or platform-based, or both. If there are multiple sources of entropy, the ST will describe each entropy source and whether it is software- or platform-based. Platform-based noise sources are preferred.

The platform-based RBG source is the output of a validated RBG provided by the platform, which is used as an entropy source for a TSF-provided DRBG according to FCS_RBG_EXT.1.1. In this way, the developer has chained RBGs as described in NIST SP800-90C.

Note that for FIPS Pub 140-2 Annex C, currently only the method described in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3 is valid. 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(1), the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

The ST author also ensures that any underlying functions are included in the baseline requirements for the TOE.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the RBG functions claimed in that platform's ST contains the RBG functions in the MDM Agent's ST. The evaluator shall also examine the TSS of the MDM Agent's ST to verify that it describes (for each supported platform) how the RBG functionality is invoked for each operation they are used for in the MDM Agent (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Agent

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with ANNEX E: ENTROPY DOCUMENTATION AND ASSESSMENT.

If the ST author has selected a platform-based noise source, the evaluator shall verify that the platform's RBG has been validated by examining the platform's ST. The evaluator shall verify that the platform's RBG is seeded with at least the amount of entropy selected by the ST author for this profile. In this case, the ST author is not responsible for Annex E documentation of the platform's RBG.

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

Implementations Conforming to FIPS 140-2, Annex C

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluator shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

The evaluator shall perform a Variable Seed Test. The evaluator shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluator shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluator ensures that the values returned by the TSF match the expected values.

The evaluator shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluator shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluator then invokes the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3. The evaluator ensures that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

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.

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).*

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 reseed. The final value is additional input to the second generate call.

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

***Entropy input:** the length of the entropy input value must equal the seed length.*

***Nonce:** If a nonce is supported (CTR_DRBG with no 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.*

Identification and Authentication (FIA)

FIA_X509_EXT.1(2) Extended: X509 Validation

FIA_X509_EXT.1.1(2) The [selection: *MDM Agent, MDM Agent platform*] shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- 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).
    - Client 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.

Application Note:

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. Certificates may optionally be used for trusted updates of TSF Software (FPT_TUD_EXT.1.3) and for software integrity verification (FPT_TST_EXT.1.2) and, if implemented, must be validated to contain the Code Signing purpose extendedKeyUsage. If TLS, DTLS, or HTTPS is selected in FPT_ITT.1 or FTP_TRP.1 or FTP_TRP.2, certificates must be used to perform authentication and must be validated to contain the Server Authentication purpose extendedKeyUsage.

It should be noted that the validation is expected to end in a trusted root certificate.

While FIA_X509_EXT.1.1 requires that the TOE perform certain checks on the certificate presented by a TLS client, there are corresponding checks that the client will have to perform on the certificate presented by the client; namely that the extendedKeyUsage field of the client certificate includes "Client Authentication" and that the key agreement bit (for the Diffie-Hellman ciphersuites) or the key encipherment bit (for RSA ciphersuites) be set. Certificates obtained for use by the TOE will have to conform to these requirements in order to be used in the enterprise.

FIA_X509_EXT.1.2(2) The [selection: *MDM Agent, MDM Agent platform*] shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

Application Note:

This requirement applies to certificates that are used and processed by the MDM Agent or platform.

Assurance Activity

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

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

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, and then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions may require to ensure the validation *is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator 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.

FIA_X509_EXT.2(2) Extended: X509 Authentication

FIA_X509_EXT.2.1(2) The [selection: *MDM Agent, MDM Agent platform*] shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [selection*: IPsec, TLS, HTTPS, DTLS*]], and [selection: *code signing for system software updates, code signing for software integrity verification, policy signing, no additional uses*].

Application Note:

The ST author's selection shall match the selection of FPT_ITT.1. Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1.3) and software integrity verification (FPT_TST_EXT.1.2). If any of the code signing uses is selected, FIA_X509_EXT.2.4(2) must be included in the main body. If FMT_POL_EXT.1.3 is included in the main body, policy signing must be selected and FIA_X509_EXT.2.6(2) must be included in the main body.

Each client device will have a unique X.509v3 certificate for use by the MDM Agent; the certificate is not to be reused among clients.

FIA_X509_EXT.2.2(2) When the [selection: *MDM Agent, MDM Agent platform*] cannot establish a connection to determine the validity of a certificate, the [selection: *MDM Agent, MDM Agent platform*] shall [selection: *allow the administrator to choose whether to accept the certificate in these cases, accept the certificate, not accept the certificate*].

Application Note:

Often a connection must be established to perform a verification of the revocation status of a certificate - either to download a CRL or to perform 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 second 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 function 38 in FMT_SMF.1(1).

FIA_X509_EXT.2.3(2) The [selection: *MDM Agent, MDM Agent platform*] shall not establish a trusted communication channel if the peer certificate is deemed invalid.

Application Note:

Trusted communication channels include any of IPsec, TLS, HTTPS, or DTLS performed by the TSF. Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

Assurance Activity

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

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

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.

Protection of the TSF (FPT)

FPT_TST_EXT.1(2): TSF Testing

FPT_TST_EXT.1.1(2) The [selection: MDM Agent, MDM Agent platform] shall run a suite of self tests during initial start-up (on power on) to demonstrate correct operation of the MDM Agent.

FPT_TST_EXT.1.2(2) The [selection: MDM Agent, MDM Agent platform] shall provide the capability to verify the integrity of stored MDM Agent executable code when it is loaded for execution through the use of the [selection: MDM Agent, MDM Agent platform]-provided cryptographic services.

Application Note:

While the TOE is typically a software package running in the IT Environment, it is still capable of performing the self-test activities required above. It should be understood, however, that there is a *significant dependency on the host environment in assessing the assurance provided by the tests mentioned above (meaning that if the host environment is compromised, the self tests will not be meaningful).*

Assurance Activity:

The evaluator shall examine the TSS to ensure that it details the self tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

The evaluator shall examine the TSS to ensure that it describes how to verify the integrity of stored TSF executable code when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. The evaluator shall perform the following tests:

- *Test 1: The evaluator performs the integrity check on a known good TSF executable and verifies that the check is successful.*
- *Test 2: The evaluator modifies the TSF executable, performs the integrity check on the modified TSF executable and verifies that the check fails.*

4.5 Security Assurance Requirements

The Security Objectives for the TOE in Section 3 were constructed to address threats identified in Section 2 The Security Functional Requirements (SFRs) in Sections 4.2, 4.3, and 4.4 are a formal instantiation of the Security Objectives. The PP draws from EAL1 the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

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 Sections 4.2, 4.3, and 4.4 as well as in this section.

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

After the ST has been approved for evaluation, the ITSEF will obtain the TOE, supporting environmental IT, and the administrative guides for the TOE. The Assurance Activities listed in the ST (which will be refined by the ITSEF to be TOE-specific, either within the ST or in a separate document) will then be performed by the ITSEF. The ITSEF is also expected to perform all of the actions mandated by the Common Evaluation Methodology (CEM) for EAL1. The results of these activities will be documented and presented (along with the administrative guidance used) for validation.

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 (to those already contained in Sections 4.2, 4.3, and 4.4 and the CEM for EAL1) 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 Sections 4.2, 4.3, and 4.4.

The TOE security assurance requirements, summarized in Table 1, identify the management and evaluative activities required to meet the objectives identified in Section 3 of this PP.

Table 1 Security Assurance Requirements

Assurance Class	**Assurance Components**
Security Target (ASE)	ST introduction (ASE_INT.1)
	Conformance claims (ASE_CCL.1)
	Security objectives for the operational environment (ASE_OBJ.1)
	Extended components definition (ASE_ECD.1)
	Stated security requirements (ASE_REQ.1)
	TOE summary specification (ASE_TSS.1)
Development (ADV)	Basic functional specification (ADV_FSP.1)
Guidance documents (AGD)	Operational user guidance (AGD_OPE.1)
	Preparative procedures (AGD_PRE.1)
Life cycle support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM coverage (ALC_CMS.1)
Tests (ATE)	Independent testing – sample (ATE_IND.1)
Vulnerability assessment (AVA)	Vulnerability survey (AVA_VAN.1)

Class ASE: Security Target

As per ASE activities defined in CEM.

Class ADV: Development

The information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (TSS) portion of the ST. The TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The Assurance Activities contained in Sections 4.2, 4.3, and 4.4 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

Basic Functional Specification (ADV_FSP)

The functional specification describes the Target Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this PP will necessarily have interfaces to the Operational Environment that are not directly invokable by TOE users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this PP, the activities for this family should focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional "functional specification" documentation is necessary to satisfy the assurance activities specified.

The interfaces that need to be evaluated are characterized through the information needed to perform the assurance activities listed, rather than as an independent, abstract list.

Developer action elements:

ADV_FSP.1.1D The developer shall provide a functional specification.

ADV_FSP.1.2D The developer shall provide a tracing from the functional specification to the SFRs.

Application Note:

As indicated in the introduction to this section, the functional specification is comprised of the information contained in the AGD_OPE and AGD_PRE documentation.

The developer may reference a website accessible to application developers and the evaluator.

The 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.

Assurance Activity:

There are no specific assurance activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Sections 4.2, 4.3, and 4.4, 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.*

Class AGD: Guidance Documentation

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

Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes:

- instructions to successfully install the TSF in that environment; and
- instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
- instructions to provide a protected administrative capability.

Guidance pertaining to particular security functionality must also be provided; requirements on such guidance are contained in the assurance activities specified with each requirement.

Operational User Guidance (AGD_OPE)

Developer action elements:

AGD_OPE.1.1D The developer shall provide operational user guidance.

Application Note:

The operation user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages. Where appropriate, the guidance documentation is expressed in the eXtensible Configuration Checklist Description Format (XCCDF) to support security automation.

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

Content and presentation elements:

AGD_OPE.1.1C The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

Application Note:

User and administrator are to be considered in the definition of user role.

AGD_OPE.1.2C The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3C The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4C The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5C The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AGD_OPE.1.6C The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7C The operational user guidance shall be clear and reasonable.

Evaluator action elements:

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

Assurance Activity:

Some of the contents of the operational guidance will be verified by the assurance activities in Sections 4.2, 4.3, and 4.4and evaluation of the TOE according to the CEM. The following additional information is also required.

If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The documentation must describe the process for verifying updates to the TOE by verifying a digital signature – this may be done by the TOE or the underlying platform. The evaluator shall verify that this process includes the following steps:

Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.

The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

Preparative Procedures (AGD_PRE)

Developer action elements:

AGD_PRE.1.1D The developer shall provide the TOE, including its preparative procedures.

Application Note:

As with the operational guidance, the developer should look to the assurance activities to determine the required content with respect to preparative procedures.

Content and presentation elements:

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

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

Evaluator action elements:

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

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

Assurance Activity:

As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this PP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor’s development and configuration management process. This is not meant to diminish the critical role that a developer’s practices play in contributing to the overall trustworthiness of a product; rather, it’s a reflection on the information to be made available for evaluation at this assurance level.

Labeling of the TOE (ALC_CMC)

This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user.

Developer action elements:

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

Content and presentation elements:

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

Evaluator action elements:

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

Assurance Activity:

The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

TOE CM Coverage (ALC_CMS)

Given the scope of the TOE and its associated evaluation evidence requirements, this component's assurance activities are covered by the assurance activities listed for ALC_CMC.1.

Developer action elements:

ALC_CMS.2.1D The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.2.1C The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.2.2C The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

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

Assurance Activity:

The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component.

Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

Assurance Activity:

The evaluator shall ensure that the developer has identified (in public-facing development documentation for their platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow *protection mechanisms in the environment(s) are invoked (e.g., compiler flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled.*

The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

Class ATE: Tests

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

Since many of the APIs are not exposed at the user interface (e.g., touch screen), the ability to stimulate the necessary interfaces requires a developer's test environment. This test environment will allow the evaluator, for example, to access APIs and view file system information that is not available on consumer mobile devices.

Independent Testing – Conformance (ATE_IND)

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operational) documentation provided. The focus of the testing is to confirm that the requirements specified in Sections 4.2, 4.3, and 4.4 being met, although some additional testing is specified for SARs in Section 4.5. 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:

The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the CEM and the body of this PP's Assurance Activities. While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered.

The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS/HTTPS, SSH).

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

Class AVA: Vulnerability Assessment

For the first generation of this protection profile, the evaluation lab is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, the evaluator will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

Vulnerability Survey (AVA_VAN)

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:

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 network infrastructure devices and the implemented communication protocols 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. 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.

5. RATIONALE

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. The only outstanding mappings are those for the Assumptions and Organizational Security Policies; those are contained in Annex A below.

ANNEX A: SUPPORTING TABLES

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 MDM 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.

A.1 Assumptions

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.

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

Table 2 TOE Assumptions

Assumption Name	Assumption Name
A.CONNECTIVITY	The TOE relies on network connectivity to carry out its management activities. The TOE will robustly handle instances when connectivity is unavailable or unreliable.
A.MOBILE_DEVICE_PLATFORM	The MDM Agent relies upon an evaluated Mobile platform and hardware to provide policy enforcement as well as cryptographic services and data protection.
A.MDM_SERVER_PLATFORM	The MDM Server relies upon a trustworthy platform and local network from which it provides administrative capabilities. The MDM Server relies on this platform to provide logon services via a local or network directory service, and to provide basic audit log management functions. The platform is expected to be configured specifically to provide MDM services, employing features such as a host-based firewall which limits its network role to providing MDM functionality.
A.PROPER_ADMIN	One or more competent, trusted personnel who are not careless, willfully negligent, or hostile, are assigned and authorized as the TOE Administrators, and do so using and abiding by guidance documentation.
A.PROPER_USER	Mobile device users are not willfully negligent or hostile, and use the device within compliance of a reasonable Enterprise security policy.

A.TIMESTAMP	The platforms on which the MDM Agent and MDM Server operate shall be able to provide reliable time stamps.

A.2 Threats

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 3 Threats

Threat	Description of Threat
T.MALICIOUS_APPS	An administrator of the MDM or mobile device user may inadvertently import malicious code, or an attacker may insert malicious code into the TOE or OE, resulting in the compromise of TOE or TOE data.
T.NETWORK_ATTACK	An attacker may masquerade as MDM Server and attempt to compromise the integrity of the mobile device by sending malicious management commands. An attacker may masquerade as MDM Agent and attempt to compromise the integrity of the MDM by sending malicious records.
T.NETWORK_EAVESDROP	Unauthorized entities may intercept communications between the MDM and mobile devices to monitor, gain access to, disclose, or alter remote management commands. Unauthorized entities may intercept unprotected wireless communications between the mobile device and the Enterprise to monitor, gain access to, disclose, or alter TOE data.
T.PHYSICAL_ACCESS	The mobile device may be lost or stolen, and an unauthorized individual may attempt to access OE data.

A.3 Organizational Security Policies

An organizational security policy is a set of rules, practices, and procedures imposed by an organization to address its security needs. The following OSPs must be enforced by the TOE or its operational environment.

Table 4 Organizational Security Policies

Policy Name	Policy Definition
P.ADMIN	The configuration of the mobile device security functions must adhere to the Enterprise security policy.
P.DEVICE_ENROLL	A mobile device must be enrolled for a specific user by the administrator

	of the MDM prior to being used in the Enterprise network by the user.
P.NOTIFY	The mobile user must immediately notify the administrator if mobile device is lost or stolen.
P.ACCOUNTABILITY	Personnel operating the TOE shall be accountable for their actions within the TOE.

A.4 Security Objectives for the TOE

The following table identifies security objectives for the TOE.

Table 5 Security Objectives for the TOE

Objective	Objective Description
O.APPLY_POLICY	The TOE must facilitate configuration and enforcement of enterprise security policies on mobile devices via interaction with the mobile OS. This will include the initial enrollment of the device into management, through its lifecycle including policy updates and through its possible unenrollment from management services.
O.ACCOUNTABILITY	The TOE must provide logging facilities which record management actions undertaken by its administrators
O.DATA_PROTECTION_TRANSIT	Data exchanged between and from elements of the TOE and its operating environment must be protected from being monitored, accessed and altered.
O.MANAGEMENT	The TOE provides access controls around its management functionality.

A.5 Security Objectives for the Operational Environment

The following table contains objectives for the Operational Environment. As assumptions are added to the PP, these objectives should be augmented to reflect such additions.

Table 6 Security Objectives for the Operational Environment

Objective	Objective Description
OE.IT_ENTERPRISE	The Enterprise IT infrastructure provides security for a network that is available to the TOE and mobile devices that prevents unauthorized access.
OE.MOBILE_DEVICE_PLATFORM	The MDM Agent relies upon the trustworthy Mobile platform and hardware to provide policy enforcement as well as

	cryptographic services and data protection.
OE.MDM_SERVER_PLATFORM	The MDM Server relies upon a trustworthy platform and local network from which it provides administrative capabilities.
OE.PROPER_ADMIN	TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.
OE.PROPER_USER	Users of the mobile device are trained to securely use the mobile device and apply all guidance in a trusted manner.
OE.WIRELESS_NETWORK	A wireless network will be available to the mobile devices.
OE.TIMESTAMP	Reliable timestamp is provided by the operational environment for the TOE.

ANNEX B: OPTIONAL REQUIREMENTS

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. Additionally, there are three other types of requirements specified in Appendices B, C, and D.

The first type (in this Appendix) are requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this PP. The second type (in Appendix C) are requirements based on selections in the body of the PP: if certain selections are made, then additional requirements in that appendix will need to be included. The third type (in Appendix D) are components that are not required in order to conform to this PP, but will be included in the baseline requirements in future versions of this PP, so adoption by VPN Client vendors is encouraged. Note that the ST author is responsible for ensuring that requirements that may be associated with those in Appendix B, Appendix C, and/or Appendix D but are not listed (e.g., FMT-type requirements) are also included in the ST.

This Annex is divided into two subsections: optional requirements that may be performed by the TSF and optional requirements that may be performed by the MDM Server or its underlying platform

B.1 Optional TSF Requirements

Security Audit (FAU)

FAU_SEL.1(1) Security Audit Event Selection (MDM Server)

FAU_SEL.1.1(1) The **MDM Server** shall be able to select the set of events to be audited from the set of all auditable events based on the following attributes:

a. **event type;**
b. **success of auditable security events;**
c. **failure of auditable security events; and**
d. **[assignment: other attributes].**

Application Note:

The intent of this requirement is to identify all criteria that can be selected to trigger an audit event. The ST author must select whether the MDM Server or the platform maintains the audit record. For the ST author, the assignment is used to list any additional criteria or "none".

Assurance Activity:

The evaluator shall review the administrative guidance to ensure that the guidance itemizes all event types, as well as describes all attributes that are to be selectable in accordance with the requirement, to include those attributes listed in the assignment. The administrative guidance shall also contain instructions on how to set the pre-selection as well as explain the syntax (if present) for multi-value pre-selection. The administrative guidance shall also identify those audit records that are always recorded, regardless of the selection criteria currently being enforced.

The evaluator shall also perform the following tests:

Test 1: For each attribute listed in the requirement, the evaluator shall devise a test to show that selecting the attribute causes only audit events with that attribute (or those that are always recorded, as identified in the administrative guidance) to be recorded.

Test 2 [conditional]: If the TSF supports specification of more complex audit pre-selection criteria (e.g., multiple attributes, logical expressions using attributes) then the evaluator shall devise tests showing that this capability is correctly implemented. The evaluator shall also, in the test plan, provide a short narrative justifying the set of tests as representative and sufficient to exercise the capability.

B.2 Optional MDM Server or MDM Server Platform Requirements

Security Audit (FAU)

FAU_SAR.1 Audit Review (MDM Server)

FAU_SAR.1.1 **Refinement:** The [selection: *MDM Server, MDM Server platform*] shall provide **Authorized Administrators** with the capability to read **all audit data** from the audit records.

FAU_SAR.1.2 Refinement: The [selection: *MDM Server, MDM Server platform*] shall provide the audit records in a manner suitable for the Authorized Administrators to interpret the information.

Application Note:

The intent of this requirement is to ensure that the administrator can view and interpret the audit records and to prevent unauthorized users from accessing the logs.

Assurance Activity:

The evaluator shall check the AGD guidance and ensure that it describes how the administrator accesses the audit data and describes the format of the audit record.

Test 1: The evaluator shall attempt to view the audit record as the authorized administrator and verify that the action succeeds. The evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide.

FAU_STG_EXT.2 Extended: Audit Event Storage

FAU_STG_EXT.2.1 The [selection: MDM Server, MDM Server platform] shall protect the stored audit records in the audit trail from unauthorized modification.

Application Note:

The purpose of this requirement is to ensure that audit records are stored securely. The ST author is responsible for selecting whether audit records are maintained when audit storage or failure occurs. The ST author must choose a means by which audit records are saved and select the events during which the records will be saved. The MDM Server may rely on the underlying operating system for this functionality, and the first selection should be made appropriately.

Assurance Activity:

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 TOE uses audit trail specific protection mechanisms. The evaluator shall perform the following tests:

- *Test 1: The evaluator shall access the audit trail as an unauthorized user 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 user and attempt to modify and delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records intended for modification and deletion are modified and deleted.*

ANNEX C: SELECTION-BASED REQUIREMENTS

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

Additionally, depending on the requirements selected, the appropriate information from Section C.4 Auditable Events will need to be added to the auditable events table in the ST.

This Annex is divided into four subsections: selection-based requirements that may be performed by the TSF, selection-based requirements that may be performed by the MDM Server or its underlying platform, selection-based requirements that may be performed by the MDM Agent or its underlying platform, and auditable events.

C.1 Selection-Based TSF Requirements

Cryptographic Support (FCS)

FCS_IV_EXT.1(1) Extended: Initialization Vector Generation

FCS_IV_EXT.1.1(1) The MDM Server shall generate IVs in accordance with Table 9.

Application Note:

Table 9 *lists the requirements for composition of IVs according to the corresponding NIST Special Publications for each cipher mode. The composition of IVs generated for encryption according to a cryptographic protocol is addressed by the protocol. Thus, this requirement addresses only IVs generated for key storage encryption.*

Assurance Activity:

The evaluator shall examine the TSS to ensure that it details the encryption of user credentials, persistent secrets, and private keys and the generation of the IVs used for that encryption. The evaluator shall ensure that the generation of IVs for each key encrypted by the same KEK meets Table 9.

FCS_STG_EXT.1 Encrypted Cryptographic Key Storage (MDM Server)

FCS_STG_EXT.1.1 The MDM Server shall encrypt all keys using AES in the [selection: *Key Wrap (KW) mode, Key Wrap with Padding (KWP) mode, GCM, CCM, CBC mode*].

Application Note:

This requirement states that keys used by the TSF shall not be kept in plaintext. The intent of this requirement is to ensure that the private keys, credentials, and persistent secrets cannot be accessed in the TOE in an unencrypted state, allowing an attacker to access keys without having to exhaust the AES key space. This requirement must be including in the ST if the selection in FCS_CKM_EXT.2(1) indicates that the MDM Server is protecting private keys and persistent secrets with encryption rather than the platform-provided key storage.

If this requirement is included in the ST, FCS_IV_EXT.1 must also be included.

Assurance Activity:

The evaluator shall examine the TSS to ensure it describes in detail how user credentials, persistent secret and private keys are stored and encrypted. The evaluator shall review the TSS to determine that it makes a case that key material is not written unencrypted to persistent memory and that it identifies the mode of encryption.

C.2 Selection-Based MDM Server or MDM Server Platform Requirements

Cryptographic Support (FCS)

FCS_DTLS_EXT.1 Extended: DTLS Implementation

FCS_DTLS_EXT.1.1(1) The [selection: MDM Server, MDM Server platform] shall implement the DTLS protocol in accordance with one or more of [selection: DTLS 1.0 (RFC 4347), DTLS 1.2 (RFC 6347)].

FCS_DTLS_EXT.1.2(1) The [selection: MDM Server, MDM Server platform] shall implement the requirements in FCS_TLS_EXT.1 for the DTLS implementation, except where variations are allowed according to [selection: RFC 4347, RFC 6347].

Application Note:

Differences between DTLS and TLS are outlined in RFC 4347 and RFC 6347; otherwise the protocols are the same. In particular, for the applicable security characteristics defined for the TOE, the two protocols do not differ. Therefore, all application notes and assurance activities that are listed for FCS_TLS_EXT.1 apply to the DTLS implementation.

Assurance Activity:

The evaluator shall perform the assurance activities listed for FCS_TLS_EXT.1 to verify this component.

FCS_HTTPS_EXT.1 Extended: HTTPS Implementation

FCS_HTTPS_EXT.1.1(1) The [selection: MDM Server, MDM Server platform] shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2(1) The [selection: MDM Server, MDM Server platform] shall implement HTTPS using TLS as specified in FCS_TLS_EXT.1.

Application Note:

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.

Assurance Activity:

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.

FCS_IPSEC_EXT.1 Extended: Internet Protocol Security (IPsec) Communications

FCS_IPSEC_EXT.1.1(1) The [selection: MDM Server, MDM Server platform] shall implement the IPsec architecture as specified in RFC 4301.

Assurance Activity:

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 DISCARD, BYPASS and PROTECT.

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

Test 1: The evaluator shall configure the SPD such that there is a rule for DISCARD, BYPASS, PROTECT. The selectors used in the construction of the rule shall be different such that the evaluator can send in three network packets with the appropriate fields in the packet header that each packet will match one of the three rules. The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packet was dropped, allowed through without modification, was encrypted by the IPsec implementation.

Test 2: The evaluator shall devise two equal SPD entries with alternate operations – BYPASS and PROTECT. The entries should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first entry is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.

Test 3: The evaluator shall repeat the procedure above, except that the two entries should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

FCS_IPSEC_EXT.1.2(1) The [selection: MDM Server, MDM Server platform] shall implement [selection: tunnel mode, transport mode].

Assurance Activity:

The evaluator shall check the TSS to ensure it states that the TOE can operate in tunnel mode and/or transport mode (as selected). The evaluator shall confirm that the operational guidance instructs the administrator how the TOE is configured in each mode selected. The evaluator shall also perform the following tests:

- *Test 1 (conditional): If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in tunnel mode and also configures a VPN GW to operate in tunnel mode. The evaluator configures the TOE and the VPN GW 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 to connect to the VPN GW peer. The evaluator observes in the audit trail and the captured packets that a successful connection was established using the tunnel mode.*
- *Test 2 (conditional): If transport mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in transport mode and also configures a VPN GW to operate in transport mode. The evaluator configures the TOE and the VPN GW 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 to connect to the VPN GW. The evaluator observes in the audit trail and the captured packets that a successful connection was established using the transport mode.*

FCS_IPSEC_EXT.1.3(1) The [selection: MDM Server, MDM Server platform] shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

Assurance Activity:

The evaluator shall examine the TSS to verify that the TSS provides a description of how a packet is processed against the SPD and that if no "rules" are found to match, that a final rule exists, either implicitly or explicitly, that causes the network packet to be discarded.

The evaluator checks that the operational guidance provides instructions on how to construct the SPD and uses the guidance to configure the TOE/platform for the following tests.

The evaluator shall perform the following test:

Test 1: The evaluator shall configure the SPD such that it has entries that contain operations that DISCARD, BYPASS, and PROTECT network packets. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches a BYPASS entry and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE/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 not permitted to flow to any of the TOE's interfaces.

FCS_IPSEC_EXT.1.4(1) The [selection: MDM Server, MDM Server platform] shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-GCM-128, AES-GCM-256 as specified in RFC 4106, [selection: AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC, no other algorithms].

Assurance Activity:

The evaluator shall examine the TSS to verify that the algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the ST author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. The evaluator shall check the operational guidance to ensure it provides instructions on how to configure the TOE to use the AES-GCM-128, and AES-GCM-256 algorithms, and if either AES-CBC-128 or AES-CBC-256 have been selected the guidance instructs how to use these as well. The evaluator shall perform the following tests:

- *Test 1: The evaluator shall configure the TOE as indicated in the operational guidance configuring the TOE to using each of the AES-GCM-128, and AES-GCM-256 algorithms, and attempt to establish a connection using ESP. If the ST Author has selected either AES-CBC-128 or AES-CBC-256, the TOE is configured to use those algorithms and the evaluator attempts to establish a connection using ESP for those algorithms selected.*

FCS_IPSEC_EXT.1.5(1) 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 RFCs 5996 (with mandatory support for NAT traversal as specified in section 2.23), 4307, and [selection: no other RFCs for hash functions, RFC 4868 for hash functions]].

Assurance Activity:

The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the following test:

- *Test 1: The evaluator shall configure the TOE 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.*

FCS_IPSEC_EXT.1.6(1) The [selection: MDM Server, MDM Server platform] shall ensure the encrypted payload in the [selection: IKEv1, IKEv2] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [selection: AES-GCM-128, AES-GCM-256 as specified in RFC 5282, no other algorithm].

Assurance Activity:

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. The evaluator shall ensure that the operational guidance describes how the TOE can be configured to use the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE to perform the following test:

- *Test 1: The evaluator shall configure the TOE to use AES-CBC-128 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 AES-CBC-128. The evaluator will consult the audit trail to confirm the algorithm was that used in the negotiation.*

FCS_IPSEC_EXT.1.7(1) The [selection: MDM Server, MDM Server platform] shall ensure that IKEv1 Phase 1 exchanges use only main mode

Application Note:

FCS_IPSEC_EXT.1.7 is only applicable if IKEv1 is selected

Assurance Activity:

The evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol supported by the TOE, 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. If the mode requires configuration of the TOE prior to its operation, the evaluator shall check the operational guidance to ensure that instructions for this configuration are contained within that guidance. The evaluator shall perform the following test:

- *Test 1 (conditional): The evaluator shall configure the TOE 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. This test is not applicable if IKEv1 is not selected above in the FCS_IPSEC_EXT.1.5 protocol selection.*

FCS_IPSEC_EXT.1.8(1) The [selection: MDM Server, MDM Server platform] shall ensure that [selection: IKEv2 SA lifetimes can be configured by an Authorized Administrator based on number of packets/number of bytes or length of time, where the time values can be limited to: 24 hours for Phase

1 SAs and 8 hours for Phase 2 SAs; IKEv1 SA lifetimes can be configured by an Authorized Administrator based on number of packets/number of bytes or length of time, where the time values can be limited to: 24 hours for Phase 1 SAs and 8 hours for Phase 2 SAs].

Application Note:

The ST Author is afforded a selection based on the version of IKE in their implementation. An implementation that allows an administrator to configure the MDM Server that pushes the SA lifetime down to the Agent are both acceptable.

As far as SA lifetimes are concerned, the TOE can limit the lifetime based on the number of bytes transmitted, or the number of packets transmitted. Either packet-based or volume-based SA lifetimes are acceptable.

The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5. The IKEv1 requirement can be accomplished either by providing Authorized Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE), or by "hard coding" the limits in the implementation. For IKEv2, there are no hardcoded limits, but in this case it is required than an administrator be able to configure the values. In general, instructions for setting the parameters of the implementation, including lifetime of the SAs, should be included in the administrative guidance generated for AGD_OPE. It is appropriate to refine the requirement in terms of number of MB/KB instead of number of packets, as long as the TOE is capable of setting a limit on the amount of traffic that is protected by the same key (the total volume of all IPsec traffic protected by that key).

Assurance Activity:

How the lifetimes are established and enforced is described in the RFCs and the evaluator examines the TSS as stated at the beginning of this section. 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. The evaluator shall ensure that the Administrator is able to configurable Phase 1 SAs values for 24 hours and 8 hours for Phase 2 SAs. Currently there are no values mandated for the number of packets or number of bytes, the evaluator just ensures that this can be configured.

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

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

- *Test 1: The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.*
- *Test 2: The evaluator shall construct a test where a Phase 1 SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that*

this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.

- *Test 3: The evaluator shall perform a test similar to Test 1 for Phase 2 SAs, except that the lifetime will be 8 hours instead of 24.*

FCS_IPSEC_EXT.1.9(1) The [selection: MDM Server, MDM Server platform] 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 “bits of security” value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General] bits.

FCS_IPSEC_EXT.1.10(1) The [selection: MDM Server, MDM Server platform] shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[assignment: (one or more) “bits of security” value(s) associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General] .

Application Note:

*Since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the SAs, the assignments in FCS_IPSEC_EXT.1.9 and FCS_IPSEC_EXT.1.10 may contain multiple values. For each DH group supported, the ST author consults Table 2 in 800-57 to determine the “bits of security” associated with the DH group. Each unique value is then used to fill in the assignment (for 1.9 they are doubled; for 1.10 they are inserted directly into the assignment). For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192. For FCS_IPSEC_EXT.1.9, then, the assignment would read “*224, 384+” and for FCS_IPSEC_EXT.1.10 it would read “*112,192+” (although in this case the requirement should probably be refined so that it makes sense mathematically).*

FCS_IPSEC_EXT.1.11(1) The [selection: MDM Server, MDM Server platform] 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), [assignment: other DH groups that are implemented by the TOE], no other DH groups].

Application Note:

The selection is used to specify additional DH groups supported. This applies to IKEv1 and IKEv2 exchanges. In future versions of this PP, DH Group 19 and 20 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.

Assurance Activity:

The evaluator shall check to ensure that, for each DH group supported by the TSF, the TSS describes the process for generating "x" (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" and the nonces meet the stipulations in the requirement.

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. The evaluator shall also perform the following test:

- *Test 1: For each supported DH group, the evaluator shall test to ensure that all IKE protocols can be successfully completed using that particular DH group.*

FCS_IPSEC_EXT.1.12(1) The [selection: MDM Server, MDM Server platform] shall ensure that all IKE protocols perform peer authentication using a [selection: RSA, ECDSA] that use X.509v3 certificates that conform to RFC 4945 and [selection: Pre-shared Keys, no other method].

Application Note:

At least one public-key-based Peer Authentication method is required for conformant TOEs; one or more of the public key schemes is chosen by the ST author to reflect what is implemented by the TOE. 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).

FCS_IPSEC_EXT.1.13(1) The [selection: MDM Server, MDM Server platform] shall not establish an SA if the distinguished name (DN) contained in a certificate does not match the expected DN for the entity attempting to establish a connection.

Assurance Activity:

The evaluator shall ensure that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The evaluator shall check that the operational guidance describes how pre-shared keys are to be generated and established for a TOE. The description in the TSS and the operational guidance shall also indicate how pre-shared key establishment is accomplished for both TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key. The evaluator shall ensure the operational guidance describes how to set up the TOE to use the cryptographic algorithms RSA and/or ECDSA.

In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the operation guidance also describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE and marked "trusted".

The evaluator shall perform the tests to verify the certificate validation in conjunction with the tests for FIA_X509_EXT.2.1

- *Test 1 [conditional]: The evaluator shall generate a pre-shared key and use it, as indicated in the operational guidance, to establish an IPsec connection between the TOE and its peer. If the TOE supports generation of the pre-shared key, the evaluator shall ensure that establishment of the key is carried out for an instance of the TOE generating the key as well as an instance of the TOE merely taking in and using the key.*

FCS_IPSEC_EXT.1.14(1) The [selection: MDM Server, MDM Server platform] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: IKEv1 Phase 1, IKEv2 IKE_SA] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: IKEv1 Phase 2, IKEv2 CHILD_SA] connection.

Application Note:

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 a TOE to allow 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:

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.

The evaluator simply follows the guidance to configure the TOE to perform the following tests:

- *Test 1: This test shall be performed for each version of IKE supported by the TOE. 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 by the TOE. 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 by the TOE. 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 by the TOE. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.*

FCS_SSH_EXT.1 Extended: SSH Implementation

FCS_SSH_EXT.1.1 The [selection: MDM Server, MDM Server platform] shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254, 4335, 5656, 6187 and 6668.

Application Note:

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.

*In the next version of this PP, a requirement will be added regarding rekeying. The requirement will read "*The TSF shall ensure that the SSH connection be rekeyed after no more than 2^{28} packets have been transmitted using that key."

FCS_SSH_EXT.1.2 The [selection: MDM Server, MDM Server platform] shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, password-based.

Assurance Activity:

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_SSH_EXT.1.5, and ensure that password-based authentication methods are also allowed. The evaluator shall also perform the following tests:

- *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.*
- *Test 2: 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.*

FCS_SSH_EXT.1.3 The [selection: MDM Server, MDM Server platform] shall ensure that, as described in RFC 4253, packets greater than [*assignment: number of bytes*] bytes in an SSH transport connection are dropped.

Application Note:

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.

Assurance Activity:

The evaluator shall check that the TSS describes how "large packets" in terms of RFC 4253 are detected and handled. The evaluator shall also perform the following test:

- *Test 1: The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.*

FCS_SSH_EXT.1.4 The [selection: MDM Server, MDM Server platform] shall ensure that the SSH transport implementation uses the following encryption algorithms: AES-CBC-128, AES-CBC-256, [selection: AEAD_AES_128_GCM, AEAD_AES_256_GCM, *no other algorithms*].

Application Note:

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.

Assurance Activity:

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. 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). The evaluator shall also perform the following test:*

- *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.*

FCS_SSH_EXT.1.5 The [selection: MDM Server, MDM Server platform] shall ensure that the SSH transport implementation uses SSH_RSA and [selection: PGP-SIGN-RSA, PGP-SIGN-DSS, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, no other public key algorithms] as its public key algorithm(s).

Application Note:

RFC 4253 specifies required and allowable public key algorithms. This requirement makes SSH-RSA "required" and allows others to be claimed in the ST. The ST author should make the appropriate selection, selecting "no other public key algorithms" if only SSH_RSA is implemented.

Assurance Activity:

The evaluator shall check the TSS to ensure that it lists the supported public key algorithms, and that that list corresponds to the list in this component. The evaluator shall also check the operational guidance to ensure that it contains instructions to the administrator on how to configure the public key used for the SSH protocol. The evaluator shall also perform the following test:

- *Test 1: The evaluator shall establish a SSH connection using each of the public key 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.*

FCS_SSH_EXT.1.6 The [selection: MDM Server, MDM Server platform] shall ensure that data integrity algorithms used in SSH transport connection is [selection: hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512].

Assurance Activity:

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. 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). The evaluator shall also perform the following test:

- *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.*

FCS_SSH_EXT.1.7 The [selection: MDM Server, MDM Server platform] shall ensure that the SSH key exchange method uses diffie-hellman-group14-sha1 and [*selection: ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, no other key exchange algorithms*].

Assurance Activity:

The evaluator shall ensure that operational guidance contains configuration information that will allow the security administrator to configure the TOE to use the key exchange methods listed in this requirement. 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. If group 14 is "hard-coded" into the TOE, the evaluator shall check the TSS to ensure that this is stated in the discussion of the SSH protocol. The evaluator shall also perform the following test:

- *Test 1: The evaluator shall attempt to perform a disallowed key exchange algorithm (e.g., diffie-hellman-group1-sha1), and observe that the attempt fails. The evaluator shall then attempt to perform each of the key exchange algorithms in the requirement, and observe that the attempts succeed.*

FCS_TLS_EXT.1 Extended: TLS Implementation

FCS_TLS_EXT.1.1(1) The [selection: MDM Server, MDM Server platform] shall implement one or more of the following protocols [selection: TLS 1.0 (RFC 2246), TLS 1.1 (RFC 4346), TLS 1.2 (RFC 5246)] supporting the following ciphersuites:

Mandatory Ciphersuites: TLS_RSA_WITH_AES_128_CBC_SHA and

Optional Ciphersuites: [selection: None, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_ SHA256, TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA].

FCS_TLS_EXT.1.2(1) The [selection: MDM Server, MDM Server platform] shall not establish a trusted channel if the distinguished name (DN) contained in a certificate does not match the expected DN for the peer.

Application Note:

The ciphersuites to be used in the evaluated configuration are limited by this requirement; however other ciphersuites may be implemented. 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. If administrative steps need to be taken so that the suites negotiated by the implementation are limited to those in this requirement, the appropriate instructions need to be contained in the guidance called for by AGD_OPE.

The Suite B algorithms (RFC 6460) listed above are the preferred algorithms for implementation. TLS 1.2 is the preferred protocol and may be required in the future. In addition, future publications of this PP will require that the TOE offer a means to deny all connection attempts using specified older versions of the SSL/TLS protocol.

The DN may be in the Subject Name field or the Subject Alternative Name extension of the certificate. The expected DN must be compared to the list of authorized MD users, either using a local or remote directory service.

Assurance Activity:

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. 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).

The evaluator shall verify that the TSS describes how the DN in the certificate is compared to the expected DN. If the DN is not compared automatically to the Domain Name or IP address, the evaluator shall ensure that the AGD guidance includes configuration of the expected DN for the connection.

Additional tests may be added in the future to test compliance with RFC 5246. The evaluator shall also perform the following tests:

- *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).*
- *Test 2: The following test is repeated for each supported certificate signing algorithm supported. 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.*
- *Test 3: The evaluator shall attempt a connection with a certificate where the DN matches either the configured expected DN or the Domain Name/IP address of the peer. The evaluator shall verify that the TSF is able to successfully connect. The evaluator shall attempt a connection with a certificate where the DN does not match either the configured expected DN or the Domain Name/IP address of the peer. The evaluator shall verify that the TSF is not able to successfully connect.*
- *Test 4: The evaluator shall configure the server to send a 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.*
- *Test 5: The evaluator shall setup a man-in-the-middle tool between the TOE and the server and shall perform the following modifications to the traffic:*

- *Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify 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.*
- *(conditional) If a DHE or ECDHE ciphersuite is supported, modify the signature block in the Server's KeyExchange handshake message, and verify that the client rejects the connection after receiving the Server KeyExchange.*
- *Modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify that the server rejects the connection after receiving the Client Finished handshake 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.*

Identification and Authentication (FIA)

FIA_X509_EXT.2(1) Extended: X509 Authentication

FIA_X509_EXT.2.4(1) The [selection: MDM Server, MDM Server platform] shall not [selection: install, execute] code if the code signing certificate is deemed invalid.

Application Note:

Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1.3) and software integrity verification (FPT_TST_EXT.1.2). If any of the code signing uses is selected in FIA_X509_EXT.2.1, FIA_X509_EXT.2.4 must be included in the main body.

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.

C.3 Selection-Based MDM Agent or MDM Agent Platform Requirements

Cryptographic Support (FCS)

FCS_DTLS_EXT.1 Extended: DTLS Implementation

FCS_DTLS_EXT.1.1(2) The [selection: MDM Agent, MDM Agent platform] shall implement the DTLS protocol in accordance with one or more of [selection: DTLS 1.0 (RFC 4347), DTLS 1.2 (RFC 6347)].

FCS_DTLS_EXT.1.2(2) The [selection: MDM Agent, MDM Agent platform] shall implement the requirements in FCS_TLS_EXT.1 for the DTLS implementation, except where variations are allowed according to [selection: RFC 4347, RFC 6347].

Application Note:

Differences between DTLS and TLS are outlined in RFC 4347 and RFC 6347; otherwise the protocols are the same. In particular, for the applicable security characteristics defined for the TOE, the two protocols do not differ. Therefore, all application notes and assurance activities that are listed for FCS_TLS_EXT.1 apply to the DTLS implementation.

Assurance Activity:

The evaluator shall perform the assurance activities listed for FCS_TLS_EXT.1 to verify this component.

FCS_HTTPS_EXT.1 Extended: HTTPS Implementation

FCS_HTTPS_EXT.1.1(2) The [selection: MDM Agent, MDM Agent platform] shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2(2) The [selection: MDM Agent, MDM Agent platform] shall implement HTTPS using TLS as specified in FCS_TLS_EXT.1.

Application Note:

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.

Assurance Activity:

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.

FCS_IPSEC_EXT.1 Extended: Internet Protocol Security (IPsec) Communications

FCS_IPSEC_EXT.1.1(2) The [selection: MDM Agent, MDM Agent platform] shall implement the IPsec architecture as specified in RFC 4301.

Assurance Activity:

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 DISCARD, BYPASS and PROTECT.

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

Test 1: The evaluator shall configure the SPD such that there is a rule for DISCARD, BYPASS, PROTECT. The selectors used in the construction of the rule shall be different such that the evaluator can send in three network packets with the appropriate fields in the packet header that each packet will match one of the three rules. The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packet was dropped, allowed through without modification, was encrypted by the IPsec implementation.

Test 2: The evaluator shall devise two equal SPD entries with alternate operations – BYPASS and PROTECT. The entries should then be deployed in two distinct orders and in each case the evaluator shall *ensure that the first entry is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.*

Test 3: The evaluator shall repeat the procedure above, except that the two entries should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

FCS_IPSEC_EXT.1.2(2) The [selection: MDM Agent, MDM Agent platform] shall implement [selection: tunnel mode, transport mode].

Assurance Activity:

The evaluator shall check the TSS to ensure it states that the TOE can operate in tunnel mode and/or transport mode (as selected). The evaluator shall confirm that the operational guidance instructs the administrator how the TOE is configured in each mode selected. The evaluator shall also perform the following tests:

- *Test 1 (conditional): If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in tunnel mode and also configures a VPN GW to operate in tunnel mode. The evaluator configures the TOE and the VPN GW 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 to connect to the VPN GW peer. The evaluator observes in the audit trail and the captured packets that a successful connection was established using the tunnel mode.*
- *Test 2 (conditional): If transport mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in transport mode and also configures a VPN GW to operate in transport mode. The evaluator configures the TOE and the VPN GW 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 to connect to the VPN GW. The evaluator observes in the audit trail and the captured packets that a successful connection was established using the transport mode.*

FCS_IPSEC_EXT.1.3(2) The [selection: MDM Agent, MDM Agent platform] shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

Assurance Activity:

The evaluator shall examine the TSS to verify that the TSS provides a description of how a packet is processed against the SPD and that if no "rules" are found to match, that a final rule exists, either implicitly or explicitly, that causes the network packet to be discarded.

The evaluator checks that the operational guidance provides instructions on how to construct the SPD and uses the guidance to configure the TOE/platform for the following tests.

The evaluator shall perform the following test:

Test 1: The evaluator shall configure the SPD such that it has entries that contain operations that DISCARD, BYPASS, and PROTECT network packets. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches a BYPASS entry and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE/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 not permitted to flow to any of the TOE's interfaces.*

FCS_IPSEC_EXT.1.4(2) The [selection: MDM Agent, MDM Agent platform] shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-GCM-128, AES-GCM-256 as specified in RFC 4106, [selection: AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC, no other algorithms].

Assurance Activity:

The evaluator shall examine the TSS to verify that the algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the ST author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. The evaluator shall check the operational guidance to ensure it provides instructions on how to configure the TOE to use the AES-GCM-128, and AES-GCM-256 algorithms, and if either AES-CBC-128 or AES-CBC-256 have been selected the guidance instructs how to use these as well. The evaluator shall perform the following tests:

- *Test 1: The evaluator shall configure the TOE as indicated in the operational guidance configuring the TOE to using each of the AES-GCM-128, and AES-GCM-256 algorithms, and attempt to establish a connection using ESP. If the ST Author has selected either AES-CBC-128 or AES-CBC-256, the TOE is configured to use those algorithms and the evaluator attempts to establish a connection using ESP for those algorithms selected.*

FCS_IPSEC_EXT.1.5(2) 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 RFCs 5996 (with mandatory support for NAT traversal as specified in section 2.23), 4307, and [selection: no other RFCs for hash functions, RFC 4868 for hash functions]].

Assurance Activity:

The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the following test:

- *Test 1: The evaluator shall configure the TOE 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.*

FCS_IPSEC_EXT.1.6(2) The [selection: MDM Agent, MDM Agent platform] shall ensure the encrypted payload in the [selection: IKEv1, IKEv2] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [selection: AES-GCM-128, AES-GCM-256 as specified in RFC 5282, no other algorithm].

Assurance Activity:

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. The evaluator shall ensure that the operational guidance describes how the TOE can be configured to use the mandated algorithms, as *well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE to perform the following test:*

- *Test 1: The evaluator shall configure the TOE to use AES-CBC-128 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 AES-CBC-128. The evaluator will consult the audit trail to confirm the algorithm was that used in the negotiation.*

FCS_IPSEC_EXT.1.7(2) The [selection: MDM Agent, MDM Agent platform] shall ensure that IKEv1 Phase 1 exchanges use only main mode

Application Note:

FCS_IPSEC_EXT.1.7 is only applicable if IKEv1 is selected

Assurance Activity:

The evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol supported by the TOE, 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. If the mode requires configuration of the TOE prior to its operation, the evaluator shall check the operational guidance to ensure that instructions for this configuration are contained within that guidance. The evaluator shall perform the following test:

- *Test 1 (conditional): The evaluator shall configure the TOE 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. This test is not applicable if IKEv1 is not selected above in the FCS_IPSEC_EXT.1.5 protocol selection.*

FCS_IPSEC_EXT.1.8(2) The [selection: MDM Agent, MDM Agent platform] shall ensure that [selection: IKEv2 SA lifetimes can be configured by an Authorized Administrator based on number of packets/number of bytes or length of time, where the time values can be limited to: 24 hours for Phase 1 SAs and 8 hours for Phase 2 SAs; IKEv1 SA lifetimes can be configured by an Authorized Administrator based on number of packets/number of bytes or length of time, where the time values can be limited to: 24 hours for Phase 1 SAs and 8 hours for Phase 2 SAs].

Application Note:

The ST Author is afforded a selection based on the version of IKE in their implementation. An implementation that allows an administrator to configure the MDM Server that pushes the SA lifetime down to the Agent are both acceptable.

As far as SA lifetimes are concerned, the TOE can limit the lifetime based on the number of bytes transmitted, or the number of packets transmitted. Either packet-based or volume-based SA lifetimes are acceptable.

The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5. The IKEv1 requirement can be accomplished either by providing Authorized Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE), or by "hard coding" the limits in the implementation. For IKEv2, there are no hardcoded limits, but in this case it is required than an administrator be able to configure the values. In general, instructions for setting the parameters of the implementation, including lifetime of the SAs, should be *included in the administrative guidance generated for AGD_OPE. It is appropriate to refine the requirement in terms of number of MB/KB instead of number of packets, as long as the TOE is capable of setting a limit on the amount of traffic that is protected by the same key (the total volume of all IPsec traffic protected by that key).*

Assurance Activity:

How the lifetimes are established and enforced is described in the RFCs and the evaluator examines the TSS as stated at the beginning of this section. 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. The evaluator shall ensure that the Administrator is able to configurable Phase 1 SAs values for 24 hours and 8 hours for Phase 2 SAs. Currently there are no values mandated for the number of packets or number of bytes, the evaluator just ensures that this can be configured.

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

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

- *Test 1: The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.*
- *Test 2: The evaluator shall construct a test where a Phase 1 SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.*
- *Test 3: The evaluator shall perform a test similar to Test 1 for Phase 2 SAs, except that the lifetime will be 8 hours instead of 24.*

FCS_IPSEC_EXT.1.9(2) The [selection: MDM Agent, MDM Agent platform] 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 "bits of security" value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General] bits.

FCS_IPSEC_EXT.1.10(2) The [selection: MDM Agent, MDM Agent platform] shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[assignment: (one or more) "bits of security" value(s) associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General] .

Application Note:

*Since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the SAs, the assignments in FCS_IPSEC_EXT.1.9 and FCS_IPSEC_EXT.1.10 may contain multiple values. For each DH group supported, the ST author consults Table 2 in 800-57 to determine the "bits of security" associated with the DH group. Each unique value is then used to fill in the assignment (for 1.9 they are doubled; for 1.10 they are inserted directly into the assignment). For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192. For FCS_IPSEC_EXT.1.9, then, the assignment would read "*224, 384+" and for FCS_IPSEC_EXT.1.10 it would read "*112,192+" (although in this case the requirement should probably be refined so that it makes sense mathematically).*

FCS_IPSEC_EXT.1.11(2) The [selection: MDM Agent, MDM Agent platform] 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), [assignment: other DH groups that are implemented by the TOE], no other DH groups].

Application Note:

The selection is used to specify additional DH groups supported. This applies to IKEv1 and IKEv2 exchanges. In future versions of this PP, DH Group 19 and 20 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.

Assurance Activity:

The evaluator shall check to ensure that, for each DH group supported by the TSF, the TSS describes the process for generating "x" (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" and the nonces meet the stipulations in the requirement.

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. The evaluator shall also perform the following test:

- *Test 1: For each supported DH group, the evaluator shall test to ensure that all IKE protocols can be successfully completed using that particular DH group.*

FCS_IPSEC_EXT.1.12(2) The [selection: MDM Agent, MDM Agent platform] shall ensure that all IKE protocols perform peer authentication using a [selection: RSA, ECDSA] that use X.509v3 certificates that conform to RFC 4945 and [selection: Pre-shared Keys, no other method].

Application Note:

At least one public-key-based Peer Authentication method is required for conformant TOEs; one or more of the public key schemes is chosen by the ST author to reflect what is implemented by the TOE. 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).*

FCS_IPSEC_EXT.1.13(2) The [selection: MDM Agent, MDM Agent platform] shall not establish an SA if the distinguished name (DN) contained in a certificate does not match the expected DN for the entity attempting to establish a connection.

Assurance Activity:

The evaluator shall ensure that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The evaluator shall check that the operational guidance describes how pre-shared keys are to be generated and established for a TOE. The description in the TSS and the operational guidance shall also indicate how pre-shared key establishment is accomplished for both TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key. The evaluator shall ensure the operational guidance describes how to set up the TOE to use the cryptographic algorithms RSA and/or ECDSA.

In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the operation guidance also describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE and marked "trusted".

The evaluator shall perform the tests to verify the certificate validation in conjunction with the tests for FIA_X509_EXT.2.1

- *Test 1 [conditional]: The evaluator shall generate a pre-shared key and use it, as indicated in the operational guidance, to establish an IPsec connection between the TOE and its peer. If the TOE supports generation of the pre-shared key, the evaluator shall ensure that establishment of the key is carried out for an instance of the TOE generating the key as well as an instance of the TOE merely taking in and using the key.*

FCS_IPSEC_EXT.1.14(2) The [selection: MDM Agent, MDM Agent platform] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: IKEv1 Phase 1, IKEv2 IKE_SA] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: IKEv1 Phase 2, IKEv2 CHILD_SA] connection.

Application Note:

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 a TOE to allow 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:

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.

The evaluator simply follows the guidance to configure the TOE to perform the following tests:

- *Test 1: This test shall be performed for each version of IKE supported by the TOE. 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 by the TOE. 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 by the TOE. 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 by the TOE. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.*

FCS_TLS_EXT.1 Extended: TLS Implementation

FCS_TLS_EXT.1.1(2) The [selection: MDM Agent, MDM Agent platform] shall implement one or more of the following protocols [selection: TLS 1.0 (RFC 2246), TLS 1.1 (RFC 4346), TLS 1.2 (RFC 5246)] supporting the following ciphersuites:

Mandatory Ciphersuites: TLS_RSA_WITH_AES_128_CBC_SHA and

Optional Ciphersuites: [selection: None, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_ SHA256, TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA].

FCS_TLS_EXT.1.2(2) The [selection: MDM Agent, MDM Agent platform] shall not establish a trusted channel if the distinguished name (DN) contained in a certificate does not match the expected DN for the peer.

Application Note:

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 (RFC 6460) listed above are the preferred algorithms for implementation. TLS 1.2 is the preferred protocol and may be required in the future. In addition, future publications of this PP will *require that the TOE offer a means to deny all connection attempts using specified older versions of the SSL/TLS protocol.*

The DN may be in the Subject Name field or the Subject Alternative Name extension of the certificate. The expected DN is the Domain Name or IP address used by the peer established during enrollment.

Assurance Activity

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. 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).

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

Additional tests may be added in the future to test compliance with RFC 5246. The evaluator shall also perform the following tests:

- *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).*
- *Test 2: The following test is repeated for each supported certificate signing algorithm supported. 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.*
- *Test 3: The evaluator shall attempt a connection with a certificate where the DN matches the expected DN. The evaluator shall verify that the TSF is able to successfully connect. The evaluator shall attempt a connection with a certificate where the DN does not match the expected DN. The evaluator shall verify that the TSF is not able to successfully connect.*
- *Test 4: The evaluator shall configure the server to send a 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.*
- *Test 5: The evaluator shall setup a man-in-the-middle tool between the TOE and the server and shall perform the following modifications to the traffic:*
  - *Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify 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.*
- *(conditional) If a DHE or ECDHE ciphersuite is supported, modify the signature block in the Server's KeyExchange handshake message, and verify that the client rejects the connection after receiving the Server KeyExchange.*
- *Modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify that the server rejects the connection after receiving the Client Finished handshake 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.*

Identification and Authentication (FIA)

FIA_X509_EXT.2(2) Extended: X509 Authentication

FIA_X509_EXT.2.4(2) The [selection: MDM Agent, MDM Agent platform] shall not [selection: install, execute] code if the code signing certificate is deemed invalid.

Application Note:

Certificates may optionally be used for trusted updates of system software (FPT_TUD_EXT.1.3) and software integrity verification (FPT_TST_EXT.1.2). If the code signing use is selected in FIA_X509_EXT.2.1, FIA_X509_EXT.2.4 must be included in the main body.

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.

FIA_X509_EXT.2.6(2) The [selection: MDM Agent, MDM Agent platform] shall not install policies if the policy signing certificate is deemed invalid.

Application Note:

Certificates may optionally be used for policy signing (FMT_POL_EXT.1). If the policy signing use is selected in FIA_X509_EXT.2.1, FIA_X509_EXT.2.6 must be included in the main body.

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.

C.4 Auditable Events

Depending on the specific requirements selected by the ST author REQUIREMENTS, ANNEX B: OPTIONAL REQUIREMENTS, ANNEX C: SELECTION-BASED REQUIREMENTS, and

ANNEX D: OBJECTIVE REQUIREMENTS, the ST author should include the appropriate auditable events in the ST for the requirements selected.

Table 7 TOE Security Functional Requirements and Auditable Events – Server

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ALT_EXT.1	Type of alert.	Identity of MDM Agent that sent alert.
FAU_ALT_EXT.2	Type of alert.	Identity of MDM Agent that sent alert.
FAU_GEN.1(1)	None.	
FAU_SAR.1	None.	
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	No additional information.
FAU_STG_EXT.1	None.	
FAU_STG_EXT.2	None.	
FCS_CKM_EXT.2(1)	None.	
FCS_CKM_EXT.4	Failure of the key zeroization process.	Identity of object or entity being cleared.
FCS_CKM.1	Failure of the key generation activity.	No additional information.
FCS_COP.1(1)	Failure of cryptographic signature.	Cryptographic mode of operation, name/identifier of object being signed/verified.
FCS_COP.1(2)	Failure in Cryptographic Hashing for Non-Data Integrity.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_COP.1(3)	Failure of encryption or decryption.	Cryptographic mode of operation, name/identifier of object being encrypted/decrypted.
FCS_COP.1(4)	Failure of hashing function.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_DTLS_EXT.1	None.	
FCS_HTTPS_EXT.1	None.	
FCS_IPSEC_EXT.1	Decisions to DISCARD, BYPASS, PROTECT network packets processed by the TOE. Failure to establish an IPsec SA. Establishment/Termination of an IPsec	Presumed identity of source subject. Identity of destination subject. Transport layer protocol, if applicable. Source subject service identifier, if applicable.

Requirement	Auditable Events	Additional Audit Record Contents
	SA.	The entry in the SPD that applied to the decision. Reason for failure. Non-TOE endpoint of connection (IP address) for both successes and failures.
FCS_IV_EXT.1	None.	
FCS_RBG_EXT.1(1)	Failure of the randomization process.	No additional information.
FCS_STG_EXT.1	None.	
FCS_TLS_EXT.1	Failure to establish a TLS session. Establishment/termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).
FIA_ENR_EXT.1.1	Failure of MD user authentication.	Presented credentials.
FIA_ENR_EXT.1.2	Failure of enrollment.	Reason for failure.
FIA_UAU.1	None.	
FIA_X509_EXT.1	Failure of X.509 certificate validation.	Reason for failure of validation.
FIA_X509_EXT.2	Generation of a Certificate Request Message.	Content of Certificate Request Message.
FMT_MOF.1(1)	Issuance of command to perform function. Change of policy settings.	Command sent and identity of MDM Agent recipient. Policy changed and value or full policy.
FMT_MOF.1(2)	Enrollment by a user	Identity of user.
FMT_POL_EXT.1	None.	
FMT_SMF.1(1)	None.	
FMT_SMF.1(3)	Success or failure of function.	No additional information.
FMT_SMR.1	None.	
FPT_ITT.1	Initiation of the trusted channel. Termination of the trusted channel. Failures of the trusted channel functions.	Identification of the initiator and target of failed trusted channels establishment attempt.
FPT_TST_EXT.1	Execution of this set of TSF self-tests. Detected integrity violations.	For integrity violations, the TSF code file that caused the integrity violation.

Requirement	Auditable Events	Additional Audit Record Contents
FPT_TUD_EXT.1	Initiation of update. Success or failure of update.	Version of update.
FTP_TRP.1	Initiation of the trusted channel. Termination of the trusted channel. Failures of the trusted path functions.	Identification of the claimed user identity.
FTP_TRP.2	Initiation of the trusted channel. Termination of the trusted channel. Failures of the trusted path functions.	Identification of the claimed user identity.

Table 8 TOE Security Functional Requirements and Auditable Events – Agent

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ALT_EXT.1	Type of alert.	No additional information.
FAU_GEN.1(2)	None.	
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	No additional information.
FCS_CKM_EXT.2(2)	None.	
FCS_CKM_EXT.4	Failure of the key zeroization process.	Identity of object or entity being cleared.
FCS_CKM.1	Failure of the key generation activity.	No additional information.
FCS_COP.1(5)	Failure of cryptographic signature.	Cryptographic mode of operation, name/identifier of object being signed/verified.
FCS_COP.1(6)	Failure in Cryptographic Hashing for Non-Data Integrity.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_COP.1(7)	Failure of encryption or decryption.	Cryptographic mode of operation, name/identifier of object being encrypted/decrypted.
FCS_COP.1(8)	Failure of hashing function.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_DTLS_EXT.1	None.	
FCS_HTTPS_EXT.1	None.	
FCS_IPSEC_EXT.1	Decisions to DISCARD, BYPASS,	Presumed identity of source subject.

Requirement	Auditable Events	Additional Audit Record Contents
	PROTECT network packets processed by the TOE. Failure to establish an IPsec SA. Establishment/Termination of an IPsec SA.	Identity of destination subject. Transport layer protocol, if applicable. Source subject service identifier, if applicable. The entry in the SPD that applied to the decision. Reason for failure. Non-TOE endpoint of connection (IP address) for both successes and failures.
FCS_RBG_EXT.1(2)	Failure of the randomization process.	No additional information.
FCS_TLS_EXT.1	Failure to establish a TLS session. Establishment/termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).
FIA_X509_EXT.1	Failure of X.509 certificate validation.	Reason for failure of validation.
FIA_X509_EXT.2	None.	
FMT_POL_EXT.1	Failure of policy validation.	Reason for failure of validation.
FMT_SMF.1(2)	Success or failure of function.	No additional information.
FPT_ITT.1	Initiation of the trusted channel. Termination of the trusted channel. Failures of the trusted channel functions.	Identification of the initiator and target of failed trusted channels establishment attempt.
FPT_TST_EXT.1	Execution of this set of TSF self-tests. Detected integrity violations.	For integrity violations, the TSF code file that caused the integrity violation.

ANNEX D: OBJECTIVE REQUIREMENTS

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements that specify security functionality that is desirable and these requirements are contained in this Annex. It is expected that these requirements will transition from objective requirements to baseline requirements in future versions of this PP.

At any time these may be included in the ST such that the TOE is still conformant to this PP.

This Annex is divided into two subsections: objective requirements that may be performed by the TSF and objective requirements that may be performed by the MDM Agent or its underlying platform.

D.1 Objective TSF Requirements

Security Audit (FAU)

FAU_GEN.1(2) Audit Data Generation (MDM Agent)

FAU_GEN.1.1(2) **Refinement:** The **MDM Agent** shall be able to generate an **MDM Agent** audit record of the following auditable events:

- Start-up and Shutdown of the audit functions;
- Change in MDM policy; and
- Any modification commanded by the MDM Server.

Specifically defined auditable events listed in

- Table 8
- [assignment: *other events*].

Application Note:

This requirement is added to the ST if the MDM Agent has the functionality to generate audit records. This requirement outlines the information to be included in the MDM Agent's audit records. The ST author can include other auditable events directly in the table in FAU_GEN.1.1; they are not limited to the list presented.

The change of the MDM policy must minimally indicate that the policy changed. The event record need not contain the differences between the prior policy and the new policy. Modifications commanded by the MDM Server are those commands listed in FMT_SMF.1.1.

Assurance Activity:

The evaluator shall check the TSS and ensure that it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field.

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. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

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

FAU_SEL.1(2) Security Audit Event Selection (MDM Agent)

FAU_SEL.1.1(2) The **MDM Agent** shall be able to select the set of events to be audited from the set of all auditable events based on the following attributes:

e. **event type;**
f. **success of auditable security events;**
g. **failure of auditable security events; and**
h. **[assignment: other attributes].**

Application Note:

The intent of this requirement is to identify all criteria that can be selected to trigger an audit event. For the ST author, the assignment is used to list any additional criteria or "none". This selection may be configured by the MDM Server.

Assurance Activity:

The evaluator shall review the administrative guidance to ensure that the guidance itemizes all event types, as well as describes all attributes that are to be selectable in accordance with the requirement, to include those attributes listed in the assignment. The administrative guidance shall also contain instructions on how to set the pre-selection as well as explain the syntax (if present) for multi-value pre-selection. The administrative guidance shall also identify those audit records that are always recorded, regardless of the selection criteria currently being enforced.

The evaluator shall also perform the following tests:

Test 1: For each attribute listed in the requirement, the evaluator shall devise a test to show that selecting the attribute causes only audit events with that attribute (or those that are always recorded, as identified in the administrative guidance) to be recorded.

Test 2 [conditional]: If the TSF supports specification of more complex audit pre-selection criteria (e.g., multiple attributes, logical expressions using attributes) then the evaluator shall devise tests showing that this capability is correctly implemented. The evaluator shall also, in the test plan, provide a short narrative justifying the set of tests as representative and sufficient to exercise the capability.

Security Management (FMT)

FMT_POL_EXT.1 Extended: Trusted Policy Update

FMT_POL_EXT.1.2 The MDM Server shall provide digitally signed policies and policy updates to the MDM Agent.

FMT_POL_EXT.1.3 The MDM Agent shall only accept policies and policy updates digitally signed by the Enterprise.

Application Note:

The intent of this requirement is to cryptographically tie the policies to the enterprise that mandated the policy, not to protect the policies in transit (as they are already protected by FPT_ITT.1). This is especially critical for users who connect to multiple enterprises.

Assurance Activity:

Policies must be digitally signed by the enterprise using the algorithms in FCS_COP.1(1). The evaluator shall ensure that the TSS describes how policies are digitally signed by the MDM Server. If applicable, the evaluator shall verify that the AGD guidance instructs administrators on configuring the Enterprise certificate to be used for signing policies or signing the policies before applying them.

The evaluator also ensures that the TSS (or the operational guidance) describes how the candidate policies are obtained by the MDM Agent; the processing associated with verifying the digital signature of the policy updates; and the actions that take place for successful (signature was verified) and unsuccessful (signature could not be verified) cases. The software components that are performing the processing must also be identified in the TSS and verified by the evaluators. The evaluators shall perform the following test:

- *Test 1: The evaluator shall perform a policy update in accordance with FMT_SMF.1(1). The evaluator shall verify the MDM Server signs the update and provides it to the MDM Agent. The evaluator shall verify the MDM Agent accepts the digitally signed policy.*
- *Test 2: The evaluator shall perform a policy update in accordance with FMT_SMF.1(1). The evaluator shall provide an unsigned and an incorrectly signed policy to the MDM Agent. The evaluator shall verify the MDM Agent does not accept the digitally signed policy.*

D.2 Objective MDM Server or MDM Server Platform Requirements

TOE Access (FTA)

FTA_TAB.1 Default TOE Access Banners

FTA_TAB.1.1 Before establishing a user session, the [selection: *MDM Server, MDM Server platform*] shall display an Administrator-specified advisory notice and consent warning message regarding use of the TOE.

Assurance Activity:

The TSS shall describe when the banner is displayed. The evaluator shall also perform the following test:

- *Test 1: The evaluator follows the operational guidance to configure a notice and consent warning message. The evaluator shall then start up or unlock the TSF. The evaluator shall verify that the notice and consent warning message is displayed in each instance described in the TSS.*

D.3 Objective MDM Agent or MDM Agent Platform Requirements

Security Audit (FAU)

FAU_GEN.1.2(2) **Refinement:** The [selection: *MDM Agent, MDM Agent platform*] shall record within each **MDM Agent** audit record at least the following information:

- date and time of the event,

- type of event,
- subject identity**,**
- (if relevant) the outcome (success or failure) of the event,

additional information in

- Table 8,
- [assignment: *other audit relevant information*].

Application Note:

All audits must contain at least the information mentioned in FAU_GEN.1.2(2), but may contain more information which can be assigned. The ST author shall identify in the TSS which information of the audit record that is performed by the MDM Agent and that which is performed by the MDM Agent's platform.

Assurance Activity:

The evaluator shall check the TSS and ensure that it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field.

When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

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

FAU_CRP_EXT.1 Extended: Support for Compliance Reporting of Mobile Device Configuration

FAU_CRP_EXT.1.1 The MDM Server shall provide [selection: an interface that provide responses to queries about the configuration of enrolled devices, an interface that permits the export of data about the configuration of enrolled devices].

Application Note:

The intent of this requirement is that the MDM provide information about the configuration state of enrolled devices that is sufficient to complete a security audit as defined in the operational user guide. This may take the form of an evaluation and reporting capability integral to the MDM Server, or the MDM Server providing data in a manner that enables external software to do so.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it describes either an interface for processing queries and reporting results about device configuration, or an interface for exporting information about the configuration of all enrolled devices. The evaluator shall check the operational guidance to determine that it defines the items returned/exported, and the instructions for obtaining those data.

Cryptographic Support (FCS)

FCS_CKM.1.1(4) The [selection: MDM Agent, MDM Agent platform] shall generate <u>asymmetric</u> cryptographic keys <u>used for authentication</u> in accordance with a specified cryptographic key generation algorithm [selection:

- *FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 for RSA schemes;*

- *FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4 for ECDSA schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curves];*
- *ANSI X9.31-1998, Appendix A.2.4 Using AES for RSA schemes]*

and specified cryptographic key sizes [***equivalent to, or greater than, a symmetric key strength of 112 bits***].

Application Note:

While it is expected that the public key generated be associated with an identity in an X509v3 certificate, this association is not required to be performed by the TOE, and instead is expected to be performed by a Certificate Authority in the Operational Environment.

For elliptic curve-based schemes, the key size refers to the $\log_2$ of the order of the base point.

The ANSI X9.31-1998 option will be removed from the selection in a future publication of this document. Presently, the selection is not exclusively limited to the FIPS PUB 186-4 options in order to allow industry some further time to complete the transition to the modern FIPS PUB 186-4 standard. As the preferred approach for cryptographic signature, elliptic curves will be required in future publications of this PP.

See NIST Special Publication 800-57, "Recommendation for Key Management" for information about equivalent key strengths.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the key generation function claimed in that platform's ST contains the key generation requirement in the MDM Agent's ST. The evaluator shall also examine the TSS of the MDM Agent's ST to verify that it describes (for each supported platform) how the key generation functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the MDM Agent

If the TSF implements a FIPS 186-4 signature scheme, this requirement is verified under FCS_COP.1.1(5)).

If the TSF implements the ANSI X9.31-1998 scheme, the evaluator shall check to ensure that the TSS describes how the key pairs are generated. In order to show that the TSF implementation complies with ANSI X9.31-1998, the evaluator shall ensure that the TSS contains the following information:

- *The TSS shall list all sections of the standard to which the TOE complies;*
- *For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not"), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;*
- *For each applicable section of Appendix B, any omission of functionality related to "shall" or "should" statements shall be described.*

ANNEX E: ENTROPY DOCUMENTATION AND ASSESSMENT

The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS.

Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the random comes from, where it is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

ANNEX F: GLOSSARY

Technical Definitions

Administrator	The Administrator is responsible for management activities, including setting the policy that is applied by the enterprise on the Mobile Device. This administrator is the Mobile Device Management (MDM) Administrator, acting through an MDM Agent.
Data	Program/application or data files that are stored or transmitted by a server or mobile device (MD).
Developer Modes	Developer modes are states in which additional services are available to a user in order to provide enhanced system access for debugging of software. Developer modes are states in which additional services are available to a user in order to provide enhanced system access for debugging of software. For the purpose of this profile, these modes also include boot modes which are not verified according to FPT_TUD_EXT.2.
Enrolled state	The state in which a Mobile Device is managed by a policy from an MDM system.
Enterprise Applications	Applications that are provided and managed by the enterprise.
Enterprise Data	Enterprise data is any data residing in the enterprise servers, or temporarily stored on mobile devices to which the mobile device user is allowed access according to security policy defined by the enterprise and implemented by the administrator.
Key Encryption Key (KEK)	A key used to encrypt other keys, such as DEKs or storage that contains keys.
Locked State	Powered on but most functionality is unavailable for use. User authentication is required to access functionality (when so configured).
MD	Mobile Device
MDM Agent	The MDM Agent is installed on a mobile device as an application or is part of the mobile device's OS. The MDM Agent establishes a secure connection back to the MDM Server controlled by the administrator.
Mobile Device User (User)	This is the person who uses and is held responsible for the mobile device's physical control and operation.
Operating System (OS)	Software which runs at the highest privilege level and can directly control hardware resources. Modern mobile devices typically have at least two primary operating systems: one which runs on the cellular baseband processor and one which runs on the application processor. The platform of the application processor handles most user interaction and provides the execution environment for apps.

	The platform of the cellular baseband processor handles communications with the cellular network and may control other peripherals. The term OS, without context, may be assumed to refer to the platform of the application processor.
Powered-Off State	The device has been shutdown.
Protected Data	Protected data is all non-TSF data on the MD, including all user or enterprise data. Protected data is encrypted while the MD is powered off. Protected data includes all keys in software-based secure key storage. Some or all of this data may be considered sensitive data as well.
Root Encryption Key (REK)	A key tied to the device used to encrypt other keys.
Sensitive data	Sensitive data shall be identified by the MD's TSS. Sensitive data may include all user or enterprise data or may be specific application data such as emails, messaging, documents, calendar items, and contacts. Sensitive data is optionally protected while in the locked state by the MD. Sensitive data must minimally include some or all keys in software-based key storage.
Trust Anchor Database	A list of trusted root Certificate Authority certificates.
TSF Data	Data for the operation of the TSF upon which the enforcement of the requirements relies.
Unenrolled state	The state in which the Mobile Device is not managed by an MDM system.
Unlocked State	Powered on and device functionality is available for use. Implies user authentication has occurred (when so configured).

Common Criteria Definitions

Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
CC	Common Criteria
CM	Configuration Management
PP	Protection Profile
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
Security Target (ST)	Implementation-dependent statement of security needs for a specific identified TOE.

Target of Evaluation (TOE)	Set of software, firmware and hardware under evaluation, possibly accompanied by guidance.
TOE Security Functionality (TSF)	Combined functionality of all hardware, software, and firmware of a TOE that must be relied upon for the correct enforcement of the SFRs.
TOE Summary Specification (TSS)	Documentation which provides evaluators with a description of the implementation of SFRs in the TOE.

ANNEX G: INITIALIZATION VECTOR REQUIREMENTS

Table 9 References and IV Requirements for NIST-approved Cipher Modes

Cipher Mode	Reference	IV Requirements
Electronic Codebook (ECB)	SP 800-38A	No IV
Counter (CTR)	SP 800-38A	"Initial Counter" shall be non-repeating. No counter value shall be repeated across multiple messages with the same secret key.
Cipher Block Chaining (CBC)	SP 800-38A	IVs shall be unpredictable. Repeating IVs leak information about whether the first one or more blocks are shared between two messages, so IVs should be non-repeating in such situations.
Output Feedback (OFB)	SP 800-38A	IVs shall be non-repeating and shall not be generated by invoking the cipher on another IV.
Cipher Feedback (CFB)	SP 800-38A	IVs should be non-repeating as repeating IVs leak information about the first plaintext block and about common shared prefixes in messages.
XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing (XTS)	SP 800-38E	No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer.
Cipher-based Message Authentication Code (CMAC)	SP 800-38B	No IV
Key Wrap and Key Wrap with Padding	SP 800-38F	No IV
Counter with CBC-Message Authentication Code (CCM)	SP 800-38C	No IV. Nonces shall be non-repeating.
Galois Counter Mode (GCM)	SP 800-38D	IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^32 for a given secret key unless an implementation only uses 96-bit IVs (default length).