Protection Profile for Mobile Device Fundamentals

Version 2.0 17 September 2014

Acknowledgements

This protection profile was developed by the Mobility Technical Community with representatives from industry, U.S. Government agencies, Common Criteria Test Laboratories, and international Common Criteria schemes. The National Information Assurance Partnership wishes to acknowledge and thank the members of this group whose dedicated efforts contributed significantly to the publication. These organizations include:

U.S. Government

Defense Information Systems Agency (DISA)

Information Assurance Directorate (IAD)

National Information Assurance Partnership (NIAP)

National Institute of Standards and Technology (NIST)

International Common Criteria Schemes

Australasian Information Security Evaluation Program (AISEP)

Canadian Common Criteria Evaluation and Certification Scheme (CSEC)

Information-technology Promotion Agency, Japan (IPA)

UK IT Security Evaluation and Certificate Scheme (CESG)

Industry

Apple, Inc.

BlackBerry

LG Electronics, Inc.

Microsoft Corporation

Motorola Solutions

Samsung Electronics Co., Ltd.

Other Members of the Mobility Technical Community

Common Criteria Test Laboratories

EWA-Canada, Ltd.

Gossamer Security Solutions

0. Preface

0.1 Objectives of Document

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

0.2 Scope of Document

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

0.3 Intended Readership

The target audiences of this PP are Mobile Device developers, CC consumers, evaluators and schemes.

0.4 Related Documents

Common Criteria[1]

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

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

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

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

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

0.5 Revision History

Version	Date	Description
1.0	21 October 2013	Initial Release
1.1	12 February 2014	Typographical changes and additional clarifications in application notes. Removed assignment from FCS_TLS_EXT.1 and limited testing to those ciphersuites in both FCS_TLS_EXT.1 and FCS_TLS_EXT.2.
2.0	17 September 2014	Included changes based on Technical Rapid Response Team Decisions. Clarified many requirements and assurance activities. Mandated objective requirements: • Application Access Control (FDP_ACF_EXT.1.2) • VPN Information Flow Control (FDP_IFC_EXT.1) Added new objective requirements: • Suite B cryptography for IEEE 802.11 • Certificate enrollment • Protection of additional key material types • Heap overflow protection • Bluetooth requirements • Cryptographic operation services for applications • Remote Attestation (FPT_NOT_EXT.1) Added transition dates for some objective requirements. Included hardware-isolated REK and key storage selections. Allowed key derivation by REK. Clarified FTP_ITC_EXT.1 and added FDP_UPC_EXT.1. Mandated HTTPS and TLS for application use. (FDP_UPC_EXT.1) Removed Dual_EC_DRBG as an approved DRBG. Adopted new TLS requirements. Mandated TSF Wipe upon authentication failure limit and required number of authentication failures be maintained across reboot. Clarified Management Class. Included more domain isolation discussion and tests. Updated Audit requirements and added Auditable Events table. Added SFR Category Mapping Table. Updated Use Case Templates. Moved Glossary to Introduction.

Contents

Figures / Tables

1. PP Introduction

1.1 PP Reference Identification

PP Reference: Protection Profile for Mobile Device Fundamentals

PP Version: 2.0

PP Date: 17 September 2014

1.2 Glossary

Term	Meaning
Address Space Layout Randomization (ASLR)	An anti-exploitation feature which loads memory mappings into unpredictable locations. ASLR makes it more difficult for an attacker to redirect control to code that they have introduced into the address space of a process or the kernel.
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 likely to be acting remotely and could be the Mobile Device Management (MDM) Administrator acting through an MDM Agent. If the device is unenrolled, the user is the administrator.
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
Auxiliary Boot Modes	Auxiliary boot modes are states in which the device provides p one or more components to provide an interface that enal unauthenticated user to interact with either a specific compo several components that exist outside of the device's fully authen operational state Developer Modes are a subset of Auxiliary Boot Modes but differ, as they specifically target developers and not intended for average user interaction.
CC	Common Criteria
Common Application Developer	Application developers (or software companies) often produce many applications under the same name. Mobile devices often allow shared resources by such applications where otherwise resources would not be shared.
Data	Program/application or data files that are stored or transmitted by a server or Mobile Device (MD).
Data Encryption Key (DEK)	A key used to encrypt data-at-rest.
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 the Mobile Device is managed with active policy settings from the administrator.

Term	Meaning
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.
File Encryption Key (FEK)	A DEK used to encrypt a file when File Encryption is used. FEKs are unique to each encrypted file.
Key Chaining	The method of using multiple layers of encryption keys to protect data. A top layer key encrypts a lower layer key which encrypts the data, this method can have any number of layers.
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)	The individual authorized to physically control and operate the Mobile Device. Depending on the use case, this can be the device owner or an individual authorized by the device owner.
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 OS of the application processor handles most user interaction and provides the execution environment for apps. The OS 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 OS of the application processor.
Password Authentication Factor	A type of authentication factor requiring the user to provide a secret set of characters to gain access.
Powered Off State	The device has been shutdown such that no TOE function can be performed.
PP	Protection Profile
Protected Data	Protected data is all non-TSF data, including all user or enterprise data. Protected data includes all keys in software-based secure key storage. Some or all of this data may be considered sensitive data as well.

Term	Meaning
Rich Operating System (Rich OS)	This term is a synonym used to refer to the primary operating system of the application processor defined above under "Operating System (OS)". This term is used to distinguish the primary operating system from an operating system executing in a smaller, isolated execution environment that may be present on the processor.
Root Encryption Key (REK)	A key tied to the device used to encrypt other keys.
SAR	Security Assurance Requirement
Sensitive data	Sensitive data shall be identified in the TSS section of the Security Target (ST) by the ST author. 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 protected while in the locked state (FDP_DAR_EXT.2). Sensitive data must minimally include some or all keys in software-based key storage.
SFR	Security Functional Requirement
ST	Security Target
Target of Evaluation	A set of software, firmware and/or hardware possibly accompanied by guidance. [CC1]
TOE	Target of Evaluation
TOE Security Functionality (TSF)	A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs. [CC1]
TSS	TOE Summary Specification
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.
Unlocked State	Powered on and device functionality is available for use. Implies user authentication has occurred (when so configured).

See [CC1] for other Common Criteria abbreviations and terminology.

1.3 TOE Overview

This assurance standard specifies information security requirements for Mobile Devices for use in an enterprise. A Mobile Device in the context of this assurance standard is a device which is composed of a hardware platform and its system software. The device typically provides wireless connectivity and may include software for functions like secure messaging, email, web, VPN connection, and VoIP (Voice over IP), for access to the protected enterprise network, enterprise data and applications, and for communicating to other Mobile Devices.

Figure 1 illustrates the network operating environment of the Mobile Device.

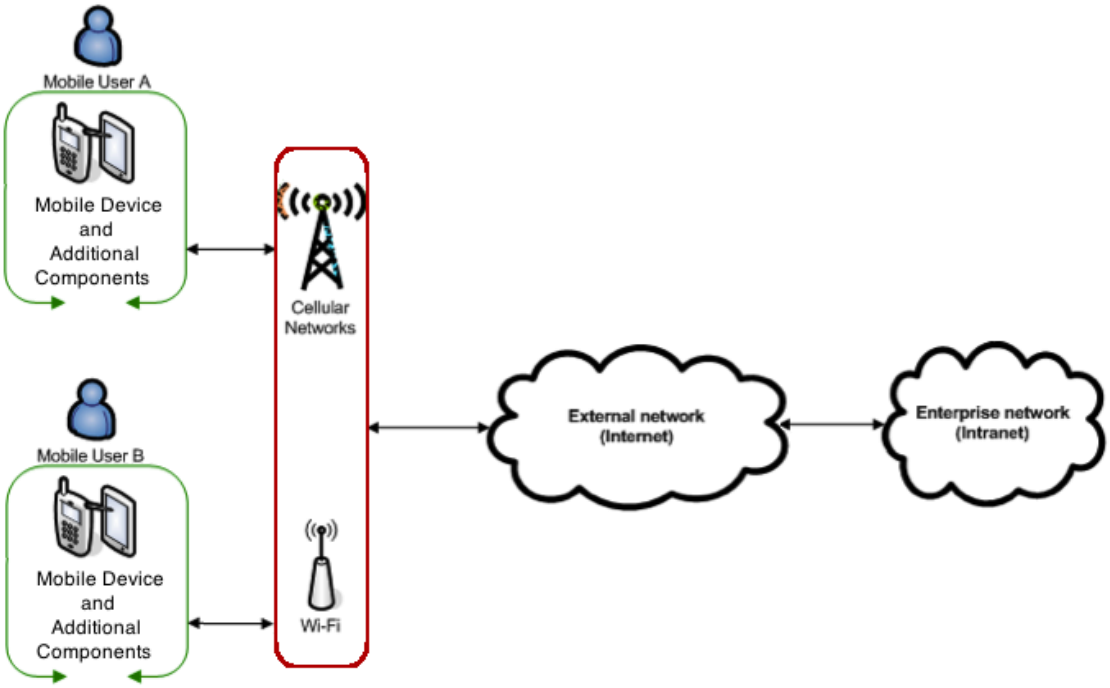

Figure 1: Mobile Device Network Environment

Examples of a "Mobile Device" that should claim conformance to this Protection Profile include smartphones, tablet computers, and other Mobile Devices with similar capabilities.

The Mobile Device provides essential services, such as cryptographic services, data-at-rest protection, and key storage services to support the secure operation of applications on the device. Additional security features such as security policy enforcement, application mandatory access control, anti-exploitation features, user authentication, and software integrity protection are implemented in order to address threats.

This assurance standard describes these essential security services provided by the Mobile Device and serves as a foundation for a secure mobile architecture. As illustrated in Figure 2, it is expected that a typical deployment would also include either third-party or bundled components. Whether these components are bundled as part of the Mobile Device by the manufacturer or developed by a third-party, they must be separately validated against the related assurance standards such as the Protection Profile for Mobile Device Management Systems, Protection Profile for IPsec VPN Clients, and Protection Profile for VoIP Applications. It is the responsibility of the architect of the overall secure mobile architecture to ensure validation of these components. Additional applications that may come pre-installed on the Mobile Device that are not validated are considered to be potentially flawed, but not malicious. Examples include VoIP client, email client, and web browser.

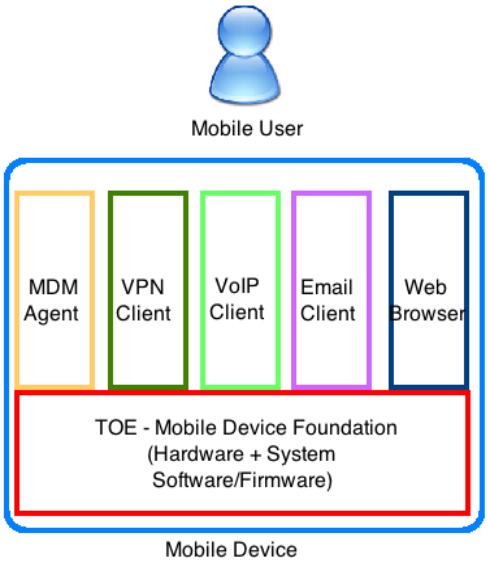

Figure 2: Optional Additional Mobile Device Components

1.4 TOE Usage

The Mobile Device may be operated in a number of use cases. Appendix G provides use case templates that list those selections, assignments, and objective requirements that best support the use cases identified by this Protection Profile. In addition to providing essential security services, the Mobile Device includes the necessary security functionality to support configurations for these various use cases. Each use case may require additional configuration and applications to achieve the desired security. A selection of these use cases is elaborated below.

Several of the use case templates include objective requirements that are strongly desired for the indicated use cases. Readers can expect those requirements to be made mandatory in the next revision of this protection profile, and industry should aim to include that security functionality in products in the near-term.

As of publication of this version of the Protection Profile, meeting the requirements in Section 5 is necessary for all use cases.

[USE CASE 1] Enterprise-owned device for general-purpose enterprise use and limited personal use

An enterprise-owned device for general-purpose business use is commonly called Corporately Owned, Personally Enabled (COPE). This use case entails a significant degree of enterprise control over configuration and, possibly, software inventory. The enterprise elects to provide users with Mobile Devices and additional applications (such as VPN or email clients) in order to maintain control of their Enterprise data and security of their networks. 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.

[USE CASE 2] 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. For example, 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 entail compliance with policies that are more restrictive than those in any general-purpose use case, yet may mitigate risks to highly sensitive information. As in the previous case, the enterprise will look for additional applications providing enterprise connectivity and services to have a similar level of assurance as the platform.

[USE CASE 3] 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, the enterprise is limited in what security policies it can enforce 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 may require assured enterprise software, for example a VPN client, before allowing access. The device may be provisioned for access to enterprise resources after significant personal usage has occurred. Based upon the operational environment and the acceptable risk level of the enterprise, those security functional requirements outlined in Section 5 of this PP are sufficient for the secure implementation of this BYOD use case.

[USE CASE 4] Personally-owned device for personal and limited enterprise use

A personally-owned device may also be given access to limited enterprise services such as enterprise email. Because the user does not have full access to the enterprise or enterprise data, the enterprise may not need to enforce any security policies on the device. However, the enterprise may want secure email and web browsing with assurance that the services being provided to those clients by the Mobile Device are not compromised. Based upon the operational environment and the acceptable risk level of the enterprise, those security functional requirements outlined in Section 5 of this PP are sufficient for the secure implementation of this BYOD use case.

2. CC Conformance

As defined by the references [CC1], [CC2] and [CC3], this cPP conforms to the requirements of Common Criteria v3.1, Revision 4. The methodology applied for the PP evaluation is defined in [CEM].

This cPP satisfies the following Assurance Families: APE_CCL.1, APE_ECD.1, APE_INT.1, APE_OBJ.1, APE_REQ.1 and APE_SPD.1.

3. Security Problem Definition

3.1 Threats

Mobile devices are subject to the threats of traditional computer systems along with those entailed by their mobile nature. The threats considered in this Protection Profile are those of network eavesdropping, network attacks, physical access, and malicious or flawed applications, as detailed in the following sections.

3.1.1 T.EAVESDROP Network Eavesdropping

An attacker is positioned on a wireless communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between the Mobile Device and other endpoints.

3.1.2 T.NETWORK Network Attack

An attacker is positioned on a wireless communications channel or elsewhere on the network infrastructure. Attackers may initiate communications with the Mobile Device or alter communications between the Mobile Device and other endpoints in order to compromise the Mobile Device. These attacks include malicious software update of any applications or system software on the device. These attacks also include malicious web pages or email attachments which are usually delivered to devices over the network.

3.1.3 T.PHYSICAL Physical Access

The loss or theft of the Mobile Device may give rise to loss of confidentiality of user data including credentials. These physical access threats may involve attacks which attempt to access the device through external hardware ports, through its user interface, and also through direct and possibly destructive access to its storage media. The goal of such attacks is to access data from a lost or stolen device which is not expected to return to its user.

Note: Defending against device re-use after physical compromise is out of scope for this protection profile.

3.1.4 T.FLAWAPP Malicious or Flawed Application

Applications loaded onto the Mobile Device may include malicious or exploitable code. This code could be included intentionally by its developer or unknowingly by the developer, perhaps as part of a software library. Malicious apps may attempt to exfiltrate data to which they have access. They may also conduct attacks against the platform’s system software which will provide them with additional privileges and the ability to conduct further malicious activities. Malicious applications may be able to control the device's sensors (GPS, camera, microphone) to gather intelligence about the user's surroundings even when those activities do not involve data resident or transmitted from the device. Flawed applications may give an attacker access to perform network-based or physical attacks that otherwise would have been prevented.

3.1.5 T.PERSISTENT Persistent Presence

Persistent presence on a device by an attacker implies that the device has lost integrity and cannot regain it. The device has likely lost this integrity due to some other threat vector, yet the continued access by an attacker constitutes an on-going threat in itself. In this case the device and its data may be controlled by an adversary at least as well as by its legitimate owner.

3.2 Assumptions

The assumptions for the Mobile Device are defined in Appendix A.1.1.

3.3 Organizational Security Policy

There are no OSPs for the Mobile Device.

4. Security Objectives

4.1 Security Objectives for the TOE

The security objectives for the Mobile Device are defined as follows.

4.1.1 O.COMMS Protected Communications

To address the network eavesdropping and network attack threats described in Section 3.1, concerning wireless transmission of Enterprise and user data and configuration data between the TOE and remote network entities, conformant TOEs will use a trusted communication path. The TOE will be capable of communicating using one (or more) of these standard protocols: IPsec, DTLS, TLS, HTTPS, or Bluetooth. The protocols are specified by RFCs that offer a variety of implementation choices. Requirements have been imposed on some of these choices (particularly those for cryptographic primitives) to provide interoperability and resistance to cryptographic attack.

While conformant TOEs must support all of the choices specified in the ST, they may support additional algorithms and protocols. If such additional mechanisms are not evaluated, guidance must be given to the administrator to make clear the fact that they were not evaluated.

FCS_CKM.1(*), FCS_CKM.2(*), FCS_CKM_EXT.7, FCS_COP.1(*), FCS_DTLS_EXT.1, FCS_HTTPS_EXT.1 FCS_RBG_EXT.1, FCS_SRV_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.2, FDP_BLT_EXT.1, FDP_IFC_EXT.1, FDP_STG_EXT.1, FDP_UPC_EXT.1, FIA_BLT_EXT.1, FIA_BLT_EXT.2, FIA_PAE_EXT.1, FIA_X509_EXT.1, FIA_X509_EXT.2, FIA_X509_EXT.3, FIA_X509_EXT.4, FPT_BLT_EXT.1, FTA_WSE_EXT.1, FTP_ITC_EXT.1

4.1.2 O.STORAGE Protected Storage

To address the issue of loss of confidentiality of user data in the event of loss of a Mobile Device (T.PHYSICAL), conformant TOEs will use data-at-rest protection. The TOE will be capable of encrypting data and keys stored on the device and will prevent unauthorized access to encrypted data.

FCS_CKM_EXT.1, FCS_CKM_EXT.2, FCS_CKM_EXT.3, FCS_CKM_EXT.4, FCS_CKM_EXT.5, FCS_CKM_EXT.6, FCS_COP.1(*), FCS_IV_EXT.1, FCS_RBG_EXT.1, FCS_STG_EXT.1, FCS_STG_EXT.2, FCS_STG_EXT.3, FDP_DAR_EXT.1, FDP_DAR_EXT.2, FIA_UAU_EXT.1, FPT_KST_EXT.1, FPT_KST_EXT.2, FPT_KST_EXT.3

4.1.3 O.CONFIG Mobile Device Configuration

To ensure a Mobile Device protects user and enterprise data that it may store or process, conformant TOEs will provide the capability to configure and apply security policies defined by the user and the Enterprise Administrator. If Enterprise security policies are configured these must be applied in precedence of user specified security policies.

FMT_MOF_EXT.1.1, FMT_MOF_EXT.1.2, FMT_SMF_EXT.1, FMT_SMF_EXT.2, FTA_TAB.1

4.1.4 O.AUTH Authorization and Authentication

To address the issue of loss of confidentiality of user data in the event of loss of a Mobile Device (T.PHYSICAL), users are required to enter an authentication factor to the device prior to accessing protected functionality and data. Some non-sensitive functionality (e.g., emergency calling, text notification) can be accessed prior to entering the authentication factor. The device will automatically lock following a configured period of inactivity in an attempt to ensure authorization will be required in the event of the device being lost or stolen.

Authentication of the endpoints of a trusted communication path is required for network access to ensure attacks are unable to establish unauthorized network connections to undermine the integrity of the device.

Repeated attempts by a user to authorize to the TSF will be limited or throttled to enforce a delay between unsuccessful attempts.

FCS_CKM.2(1), FIA_AFL_EXT.1, FIA_BLT_EXT.1, FIA_BLT_EXT.2, FIA_PMG_EXT.1, FIA_TRT_EXT.1, FIA_UAU_EXT.1, FIA_UAU_EXT.2, FIA_UAU_EXT.3, FIA_UAU.7, FIA_X509_EXT.2, FIA_X509_EXT.4, FTA_SSL_EXT.1

4.1.5 O.INTEGRITY Mobile Device Integrity

To ensure the integrity of the Mobile Device is maintained conformant TOEs will perform self-tests to ensure the integrity of critical functionality, software/firmware and data has been maintained. The user shall be notified of any failure of these self-tests. (This will protect against the threat T.PERSISTENT.)

To address the issue of an application containing malicious or flawed code (T.FLAWAPP), the integrity of downloaded updates to software/firmware will be verified prior to installation/execution of the object on the Mobile Device. In addition, the TOE will restrict applications to only have access to the system services and data they are permitted to interact with. The TOE will further protect against malicious applications from gaining access to data they are not authorized to access by randomizing the memory layout.

FAU_GEN.1, FAU_SAR, FAU_SEL.1, FAU_STG.1, FAU_STG.4, FCS_COP.1(2), FCS_COP.1(3), FDP_ACF_EXT.1, FPT_AEX_EXT.1, FPT_AEX_EXT.2, FPT_AEX_EXT.3, FPT_AEX_EXT.4, FPT_BBD_EXT.1, FPT_NOT_EXT.1, FPT_STM.1, FPT_TST_EXT.1, FPT_TST_EXT.2, FPT_TUD_EXT.1, FPT_TUD_EXT.2

4.2 Security Objectives for the Operational Environment

The objectives that are required to be met by the TOE's operational environment are defined in Appendix A.2.2.

5. Security Functional Requirements

The individual security functional requirements are specified in the sections below. For a complete list of requirements, see Appendix A.3, Security Functional Requirements Category Mapping.

5.1 Conventions

The following conventions are used for the completion of operations:

- [*Italicized text within square brackets*] indicates an operation to be completed by the ST author
- Underlined text indicates additional text provided as a refinement.
- [**Bold text within square brackets**] indicates the completion of an assignment.
- [***Bold-italicized text within square brackets***] indicates the completion of a selection.

5.2 Class: Cryptographic Support (FCS)

5.2.1 Cryptographic Key Management (FCS_CKM)

This section describes how keys are generated, derived, combined, and destroyed. There are two major types of keys: DEKs and KEKs. (A REK is considered a KEK.) DEKs are used to protect data (as in the DAR protection described in Section 0). KEKs are used to protect other keys - DEKs, other KEKs, and other types of keys stored by the user or applications. The following diagram shows an example key hierarchy to illustrate the concepts of this profile. This example is not meant as an approved design, but ST authors will be expected to provide a diagram illustrating their key hierarchy in order to demonstrate that they meet the requirements of this profile.

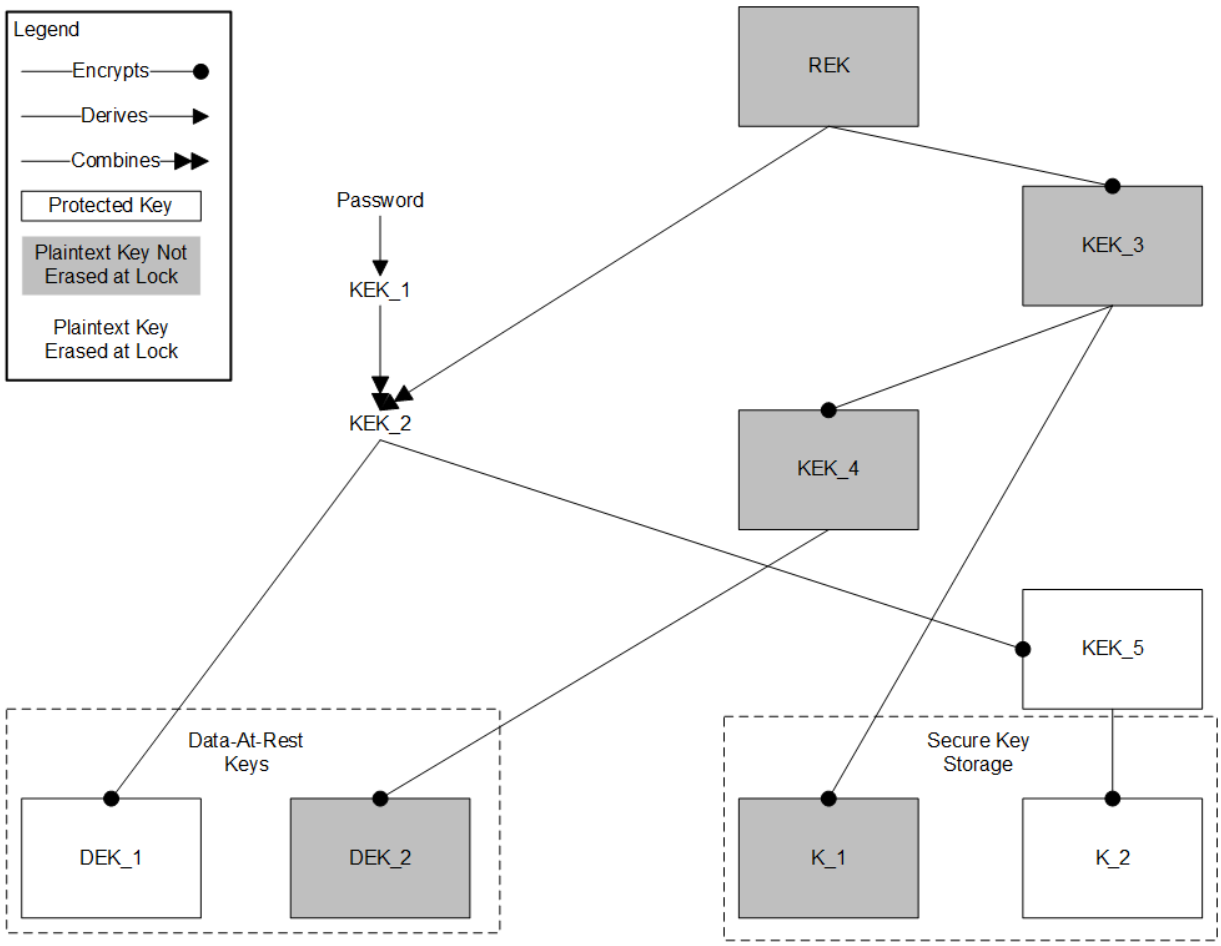

Figure 3: An Illustrative Key Hierarchy

5.2.1.1 Cryptographic Key Generation

FCS_CKM.1(1)	Cryptographic key generation

FCS_CKM.1.1(1) The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [selection:

- ***[RSA schemes]*** *using cryptographic key sizes of [**2048-bit or greater]** that meet the following: [**selection:***
    - ***FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3;***
    - ***ANSI X9.31-1998, Section 4.1];***
- ***[ECC schemes]*** *using [**"NIST curves" P-256, P-384 and [selection: P-521, no other curves]]** that meet the following: [**FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4];*
- ***[FFC schemes]*** *using cryptographic key sizes of [**2048-bit or greater]** that meet the following: [**FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1]*

].

Application Note: The ST author shall select all key generation schemes used for key establishment and entity authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2.1(1) and selected cryptographic protocols must match the selection. When key generation is used for entity authentication, the public key is expected to be associated with an X.509v3 certificate.

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

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.

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

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

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

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

Key Generation for FIPS PUB 186-4 RSA 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:

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

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

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

- n = p*q,
- p and q are probably prime according to Miller-Rabin tests,
- GCD(p-1,e) = 1,
- GCD(q-1,e) = 1,
- 2^16 <= e <= 2^256 and e is an odd integer,
- |p-q| > 2^(nlen/2 - 100),
- p >= squareroot(2)*(2^(nlen/2 -1)),
- q >= squareroot(2)*(2^(nlen/2 -1)),
- 2^(nlen/2) < d < LCM(p-1,q-1),
- e*d = 1 mod LCM(p-1,q-1).

Key Generation for ANSI X9.31-1998 RSA Schemes

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.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

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.

5.2.1.2 Cryptographic Key Generation (WLAN)

FCS_CKM.1(2)	**Cryptographic key generation**

FCS_CKM.1.1(2) The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [***PRF-384***] and specified cryptographic

key sizes [***128 bits***] using a Random Bit Generator as specified in FCS_RBG_EXT.1 that meet the following: [***IEEE 802.11-2012***].

Application Note: This requirement supports FTA_WSE_EXT.1, which requires that the Mobile Device be able to connect to configured Wireless LANs. The cryptographic key derivation algorithm required by IEEE 802.11-2012 (Section 11.6.1.2) and verified in WPA2 certification is PRF-384 which uses the HMAC-SHA-1 function and outputs 384 bits.

This requirement applies only to the keys that are generated/derived for the communications between the access point and the client once the client has been authenticated. It refers to the derivation of the PTK from the PMK, which is done using a random value generated by the RBG specified in this PP, the HMAC function using SHA-1 as specified in this PP, as well as other information. This is specified in IEEE 802.11-2012 primarily in chapter 11.

Assurance Activity:

The cryptographic primitives will be verified through assurance activities specified elsewhere in this PP. The evaluator shall verify that the TSS describes how the primitives defined and implemented by this PP are used by the TOE in establishing and maintaining secure connectivity to the wireless clients. The TSS shall also provide a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed (e.g. WPA2 certification). The evaluator shall ensure that the description of the testing methodology is of sufficient detail to determine the extent to which the details of the protocol specifics are tested.

The evaluator shall also perform the following test using a packet sniffing tool to collect frames between a wireless access point and TOE:

Step 1: The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.

Step 2: The evaluator shall configure the TOE to communicate with the access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-9 or a-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.

Step 3: The evaluator shall start the sniffing tool, initiate a connection between the TOE and access point, and allow the TOE to authenticate, associate and successfully complete the 4-way handshake with the access point.

Step 4: The evaluator shall set a timer for 1 minute, at the end of which the evaluator shall disconnect the TOE from the access point and stop the sniffer.

Step 5: The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and derive the PTK from the 4-way handshake frames and pre-shared key as specified in IEEE 802.11-2012.

Step 6: The evaluator shall select the first data frame from the captured packets that was sent between the access point and TOE after the 4-way handshake successfully completed, and without the frame control value 0x4208 (the first 2 bytes are 08 42). The evaluator shall use the PTK to decrypt the data portion of the packet as specified in IEEE 802.11-2012, and shall verify that the decrypted data contains ASCII-readable text.

Step 7: The evaluator shall repeat Step 7 for the next 2 data frames between the TOE and access point, and without frame control value 0x4208.

5.2.1.3 Cryptographic Key Establishment

FCS_CKM.2.1(1)	Cryptographic key establishment

FCS_CKM.2.1(1) The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:

- [**RSA-based key establishment schemes**] that meets the following: [**NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography"**];

and [selection:

- *[**Elliptic curve-based key establishment schemes**] that meets the following: [**NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**];*
- *[**Finite field-based key establishment schemes**] that meets the following: [**NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**];*
- *No other schemes*

].

Application Note:

The ST author shall select all key establishment schemes used for the selected cryptographic protocols. FCS_TLSC_EXT.2 requires ciphersuites that use RSA-based key establishment schemes.

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

The elliptic curves used for the key establishment scheme shall correlate with the curves specified in FCS_CKM.1.1(1). Elliptic curve-based schemes will be required for products entering evaluation after Quarter 3, 2015.

The domain parameters used for the finite field-based key establishment scheme are specified by the key generation according to FCS_CKM.1.1(1).

Assurance Activity:

The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1(1). If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

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

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

Key Establishment Schemes
The evaluator shall verify the implementation of the key establishment schemes 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.

Validity Test

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

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

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

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is

incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

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

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

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

5.2.1.4 Cryptographic Key Distribution (WLAN)

FCS_CKM.2.1(2)	Cryptographic key distribution

FCS_CKM.2.1(2) The TSF shall decrypt Group Temporal Key (GTK) in accordance with a specified cryptographic key distribution method [***AES Key Wrap in an EAPOL-Key frame***] that meets the following: [***NIST SP 800-38F, IEEE 802.11-2012 for the packet format and timing considerations***] and does not expose the cryptographic keys.

Application Note: This requirement supports FTA_WSE_EXT.1, which requires that the Mobile Device be able to connect to configured Wireless LANs. This requirement applies to

the GTK that is received by the TOE for use in decrypting broadcast and multicast messages from the Access Point to which it's connected. IEEE 802.11-2012 specifies the format for the transfer as well as the fact that it must be wrapped by the AES Key Wrap method specified in NIST SP 800-38F; the TOE must be capable of unwrapping such keys.

Assurance Activity:

The evaluator shall check the TSS to ensure that it describes how the GTK is unwrapped prior to being installed for use on the TOE using the AES implementation specified in this PP. The evaluator shall also perform the following test using a packet sniffing tool to collect frames between a wireless access point and TOE (which may be performed in conjunction with the assurance activity for FCS_CKM.1.1(2):

Step 1: The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.

Step 2: The evaluator shall configure the TOE to communicate with the access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-9 or a-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.

Step 3: The evaluator shall start the sniffing tool, initiate a connection between the TOE and access point, and allow the TOE to authenticate, associate and successfully complete the 4-way handshake with the access point.

Step 4: The evaluator shall set a timer for 1 minute, at the end of which the evaluator shall disconnect the TOE from the access point and stop the sniffer.

Step 5: The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and derive the PTK and GTK from the 4-way handshake frames and pre-shared key as specified in IEEE 802.11-2012.

Step 6: The evaluator shall select the first data frame from the captured packets that was sent between the access point and TOE after the 4-way handshake successfully completed, and with the frame control value 0x4208 (the first 2 bytes are 08 42). The evaluator shall use the GTK to decrypt the data portion of the selected packet as specified in IEEE 802.11-2012, and shall verify that the decrypted data contains ASCII-readable text.

Step 7: The evaluator shall repeat Step 7 for the next 2 data frames with frame control value 0x4208.

5.2.1.5 Cryptographic Key Support (REK)

FCS_CKM_EXT.1	Extended: Cryptographic Key Support

FCS_CKM_EXT.1.1 The TSF shall support a [selection: *hardware-isolated, hardware-protected*] REK with a key of size [selection: *128 bits, 256 bits*].

FCS_CKM_EXT.1.2 System software on the TSF shall be able only to request [selection: *AES encryption/decryption, NIST SP 800-108 key derivation*] by the key and shall not be able to read, import, or export a REK.

FCS_CKM_EXT.1.3 A REK shall be generated by a RBG in accordance with FCS_RBG_EXT.1.

Application Note: Either 128-bit or 256-bit (or both) are allowed; the ST author makes the selection appropriate for the device. Use of 256-bit key sizes will be required for products entering evaluation after Quarter 3, 2015.

In terms of this requirement, "hardware-isolated" and "hardware-protected" both imply a design such that a REK must be isolated from the rich OS such that the kernel of the OS may only request operations of this key and may not access the raw key material. The distinction between the two regard where the raw key material is accessed. The raw key material of "hardware-isolated" REK(s) is solely available in a separate processor execution environment, isolated from the OS both in terms of processing and memory. The raw key material of "hardware-protected" REK(s) is not available to any software and is computationally processed by hardware. If "hardware-protected" is selected, FCS_CKM_EXT.1.4 must be included in the ST.

The lack of a public/documented API for importing or exporting, when a private/undocumented API exists, is not sufficient to meet this requirement.

The requirements allow for either AES encryption/decryption or key derivation by a mode from SP800-108 by a REK.

The RBG used to generate a REK may be a RBG native to the hardware key container or may be an off-device RBG. If performed by an off-device RBG, the device manufacturer shall not be able to access a REK after the manufacturing process has been completed. The assurance activities for these two cases differ.

Assurance Activity:

The evaluator shall review the TSS to determine that a REK is supported by the product, that the TSS includes a description of the protection provided by the product for a REK, and that the TSS includes a description of the method of generation of a REK.

The evaluator shall verify that the description of the protection of a REK describes how any reading, import, and export of that REK is prevented. (For example, if the hardware protecting the REK is removable, the description should include how other devices are prevented from reading the REK.) The evaluator shall verify that the TSS describes how encryption/decryption/derivation actions are isolated so as to prevent applications and system-level processes from reading the REK while allowing encryption/decryption/derivation by the key.

If "hardware-isolated" is selected and REK(s) are isolated from the rich OS by a separate processor execution environment, the evaluator shall verify that the description includes how the rich OS is prevented from accessing the memory containing REK key material, which software is allowed access to the REK, how any other software in the execution environment is prevented from reading that key material, and what other mechanisms prevent the REK key

material from being written to shared memory locations between the rich OS and the separate execution environment.

If key derivation is performed using a REK, the evaluator shall ensure that the TSS description includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108. (Additional key expansion algorithms are defined in other NIST Special Publications.)

The evaluator shall verify that the generation of a REK meets the FCS_RBG_EXT.1.1 and FCS_RBG_EXT.1.2 requirements:

- If REK(s) is/are generated on-device, the TSS shall include a description of the generation mechanism including what triggers a generation, how the functionality described by FCS_RBG_EXT.1 is invoked, and whether a separate instance of the RBG is used for REK(s).
- If REK(s) is/are generated off-device, the TSS shall include evidence that the RBG meets FCS_RBG_EXT.1.2. This will likely a second set of RBG documentation equivalent to the documentation provided for the RBG assurance activities. In addition, the TSS shall describe the manufacturing process that prevents the device manufacturer from accessing any REKs.

5.2.1.6 Cryptographic Data Encryption Keys

FCS_CKM_EXT.2	Extended: Cryptographic Key Random Generation

FCS_CKM_EXT.2.1 All DEKs shall be randomly generated with entropy corresponding to the security strength of AES key sizes of [selection: *128, 256*] bits.

Application Note: The intent of this requirement is to ensure that the DEK cannot be recovered with less work than a full exhaust of the key space for AES. The key generation capability of the TOE uses a RBG implemented on the TOE device (FCS_RBG_EXT.1). Either 128-bit or 256-bit (or both) are allowed; the ST author makes the selection appropriate for the device. A DEK is used in addition to the KEK so that authentication factors (especially the Password Authentication Factor) can be changed without having to re-encrypt all of the user data on the device.

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

Assurance Activity:

The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked to generate DEKs. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the data.

5.2.1.7 Cryptographic Key Encryption Keys

FCS_CKM_EXT.3	Extended: Cryptographic Key Generation

FCS_CKM_EXT.3.1 All KEKs shall be [selection: *128-bit*, *256-bit*] keys corresponding to at least the security strength of the keys encrypted by the KEK.

Application Note:

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

FCS_CKM_EXT.3.2 The TSF shall generate all KEKs using one of the following methods:

a) derive the KEK from a Password Authentication Factor using PBKDF and

[selection:

b) *generate the KEK using an RBG that meets this profile (as specified in FCS_RBG_EXT.1)*

c) *Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [selection: using an XOR operation, concatenating the keys and use a KDF (as described in SP 800-108), encrypting one key with another]*

].

Application Note:

The PBKDF is performed in accordance with FCS_COP.1(5).

It is expected that each of these methods will be necessary to meet the requirements set out in this document. In particular, Figure 3 has KEKs of each type: KEK_3 is generated, KEK_1 is derived from a Password Authentication Factor, and KEK_2 is combined from two KEKs. If combined, the ST author shall describe which method of combination is used in order to justify that the effective entropy of each factor is preserved.

Assurance Activity:

The evaluator shall examine the key hierarchy TSS to ensure that the formation of all KEKs is described and that the key sizes match that described by the ST author.

- The evaluator shall review the TSS to verify that it contains a description of the PBKDF use to derive KEKs. This description must include the size and storage location of salts. This activity may be performed in combination with that for FCS_COP.1(5).
- If the KEK is generated, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key

size being requested is identical to the key size and mode to be used for the encryption/decryption of the data.

- If the KEK is formed from a combination, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR, a KDF, or encryption. If a KDF is used, the evaluator shall ensure that the TSS description includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108. (Additional key expansion algorithms are defined in other NIST Special Publications.)

5.2.1.8 Cryptographic Key Destruction

FCS_CKM_EXT.4	**Extended: Key Destruction**

FCS_CKM_EXT.4.1 The TSF shall destroy cryptographic keys in accordance with the specified cryptographic key destruction methods:

- by clearing the KEK encrypting the target key,
- in accordance with the following rules:
    - For volatile memory, the destruction shall be executed by a single direct overwrite [selection: *consisting of a pseudo-random pattern using the TSF's RBG*, *consisting of zeroes*] following by a read-verify.
    - For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1), followed a read-verify.
    - For non-volatile flash memory, the destruction shall be executed [selection: *by a single direct overwrite consisting of zeros followed by a read-verify, by a block erase followed by a read-verify*].
    - For non-volatile memory other than EEPROM and flash, the destruction shall be executed by overwriting three or more times with a random pattern that is changed before each write.

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

Because plaintext key material is not allowed to be written to non-volatile memory (FPT_KST_EXT.1), the second selection only applies to key material written to volatile memory.

FCS_CKM_EXT.4.2 The TSF shall destroy all plaintext keying material and critical security parameters when no longer needed.

Application Note: For the purposes of this requirement, plaintext keying material refers to authentication data, passwords, secret/private symmetric keys, private asymmetric keys, data used to derive keys, etc. Key destruction procedures are performed in accordance with FCS_CKM_EXT.4.1.

Plaintext keying material will refer to values derived from passwords for products entering evaluation after Quarter 3, 2015.

There are multiple situations in which plaintext keying material is no longer necessary, including when the TOE is powered off, when the wipe function is performed, when trusted channels are disconnected, when keying material is no longer needed by the trusted channel per the protocol, and when transitioning to the locked state (for those values derived from the Password Authentication Factor or that key material which is protected by the password-derived KEK according to FCS_STG_EXT.2 – see Figure 3). For keys (or key material used to derive those keys) protecting sensitive data received in the locked state, "no longer needed" includes "while in the locked state."

Trusted channels may include TLS, HTTPS, DTLS, IPsec VPNs, IEEE 802.11, EAP-TLS, Bluetooth BR/EDR, and Bluetooth LE. The plaintext keying material for these channels includes (but is not limited to) master secrets, Security Associations (SAs), Pre-Shared Keys (PSKs), Pairwise Master Keys (PMKs), link keys, and Long Term Keys (LTKs).

If REK(s) are processed in a separate execution environment on the same Application Processor as the Rich OS, REK key material must be cleared from RAM immediately after use, and at least, must be wiped when the device is locked, as the REK is part of the key hierarchy protecting sensitive data.

Additionally, although IEEE 802.11-2012 does not specify PMK lifetimes (described in IEEE 802.11-2012 Section 11.6.1.3) for Wireless LAN clients, these lifetimes should be limited, and the PMKSA cleared, in such a way as to prevent continued use of the same PMK for more than 24 hours. Thus, for PMKs, "when no longer needed" is after 24 hours.

Assurance Activity:

The evaluator shall check to ensure the TSS lists each type of plaintext key material (DEKs, software-based key storage, KEKs, trusted channel keys, passwords, etc.) and its origin and storage location.

The evaluator shall verify that the TSS describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state, etc.).

The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write"). For block erases, the evaluator shall also ensure that the block erase

command used is listed and shall verify that the command used also addresses any copies of the plaintext key material and that may be created in order to optimize the use of flash memory.

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

For each software and firmware key clearing situation (including on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state) the evaluator shall repeat the following tests. Note that at this time hardware-bound keys are explicitly excluded from testing.

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:

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

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.

5.2.1.9 TSF Wipe

FCS_CKM_EXT.5	Extended: TSF Wipe

FCS_CKM_EXT.5.1 The TSF shall wipe all protected data by [selection*:*

- *Cryptographically erasing the encrypted DEKs and/or the KEKs in non-volatile memory by following the requirements in FCS_CKM_EXT.4.1;*
- *Overwriting all protected data according to the following rules:*
    - *For EEPROM, the destruction shall be executed by a single direct overwrite consisting of a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1, followed a read-verify.*
    - *For flash memory the destruction shall be executed [selection: by a single direct overwrite consisting of zeros followed by a read-verify, by a block erase followed by a read-verify]*.
    - *For non-volatile memory other than EEPROM and flash, the destruction shall be executed by overwriting three or more times with a random pattern that is changed before each write*.]

FCS_CKM_EXT.5.2 The TSF shall perform a power cycle on conclusion of the wipe procedure.

Application Note: The ST author shall select which method of wipe the TSF performs.

Assurance Activity:

The evaluator shall check to ensure the TSS describes how the device is wiped; and the type of clearing procedure that is performed (cryptographic erase or overwrite) and, if overwrite is performed, the overwrite procedure (overwrite with zeros, overwrite three or more times by a different alternating pattern, overwrite with random pattern, or block erase). If different types of memory are used to store the data to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "data stored on flash are cleared by overwriting once with zeros, while data stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

Assurance Activity Note: The following test may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall perform one of the following tests. The test before and after the wipe command shall be identical. This test shall be repeated for each type of memory used to store the data to be protected.

Method 1 for File-based Methods:

Test: The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a user data (protected data or sensitive data) file, for example, by using an application. The evaluator shall use a tool provided by the developer to examine this data stored in memory (for example, by examining a decrypted files). The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF_EXT.1. The evaluator shall use a tool provided by the developer to examine the same data location in memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).

Method 2 for Volume-based Methods:

Test: The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a unique data string, for example, by using an application. The evaluator shall use a tool provided by the developer to search decrypted data for the unique string. The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF_EXT.1. The evaluator shall use a tool provided by the developer to search for the same unique string in decrypted memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).

5.2.1.10 Cryptographic Salt Generation

FCS_CKM_EXT.6	**Extended: Salt Generation**

FCS_CKM_EXT.6.1 The TSF shall generate all salts using a RBG that meets FCS_RBG_EXT.1.

Application Note: Cryptographic salts are generated for various uses including:

- RSASSA-PSS signature generation
- DSA signature generation
- ECDSA signature generation
- DH static key agreement scheme
- PBKDF
- Key Agreement Scheme in NIST SP 800-56B

Assurance Activity:

The evaluator shall verify that the TSS contains a description regarding the salt generation, including which algorithms on the TOE require salts. The evaluator shall confirm that the salt is generating using an RBG described in FCS_RBG_EXT.1. For PBKDF derivation of KEKs, this assurance activity may be performed in conjunction with FCS_CKM_EXT.3.2.

5.2.2 Cryptographic Operation (FCS_COP)

5.2.2.1 Confidentiality Algorithms

FCS_COP.1(1)	**Cryptographic operation**

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

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

[selection:

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

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

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

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

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

Assurance Activity:

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

AES-CBC Tests

AES-CBC Known Answer Tests

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.

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

AES-GCM 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.

XTS-AES Test

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

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

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

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

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

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

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 (KEKs)

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.

5.2.2.2 Hashing Algorithms

FCS_COP.1(2)	**Cryptographic operation**

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

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

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

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

Assurance Activity:
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.

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

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 $1 \leq i \leq m$. 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 $1 \leq i \leq m/8$. 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.

5.2.2.3 Signature Algorithms

FCS_COP.1(3)	**Cryptographic operation**

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

- [**RSA schemes**] using cryptographic key sizes [**of 2048-bit or greater**] that meet the following: [**FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4**]

and [selection:

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

].

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

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

Assurance Activity:

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

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

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.

5.2.2.4 Keyed Hash Algorithms

FCS_COP.1(4)	**Cryptographic operation**

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

Application Note: The selection in this requirement must be consistent with the key size specified for the size of the keys used in conjunction with the keyed-hash message authentication. HMAC-SHA-1 is currently required in order to comply with FCS_TLSC_EXT.1 and FCS_CKM.1.1(2).

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

Assurance Activity:

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.

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

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

5.2.2.5 Password-Based Key Derivation Functions

FCS_COP.1(5) Cryptographic operation

FCS_COP.1.1(5) The TSF shall perform [***Password-based Key Derivation Functions***] in accordance with a specified cryptographic algorithm [*HMAC-[selection: SHA-1, SHA-256, SHA-384, SHA-512]*], with [assignment: *integer number*] iterations, and output cryptographic key sizes [selection: *128, 256*] that meet the following: [**NIST SP 800-132**].

Application Note: The key cryptographic key sizes in the second selection should be made to correspond to the KEK key sizes selected in FCS_CKM_EXT.3. Use of 256-bit key sizes will be required for products entering evaluation after Quarter 3, 2015.

This password must be conditioned into a string of bits that forms the submask to be used as input into the KEK. Conditioning can be performed using one of the identified hash functions or the process described in NIST SP 800-132; the method used is selected by the ST Author. NIST SP 800-132 requires the use of a pseudo-random function (PRF) consisting of HMAC with an approved hash function. The ST author selects the hash function used, also includes the appropriate requirements for HMAC and the hash function.

Appendix A of NIST SP 800-132 recommends setting the iteration count in order to increase the computation needed to derive a key from a password and, therefore, increase the workload of performing a dictionary attack.

Assurance Activity:

The evaluator shall check that the TSS describes the method by which the password is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_CKM_EXT.3.

For the NIST SP 800-132-based conditioning of the passphrase, the required assurance activities will be performed when doing the assurance activities for the appropriate requirements (FCS_COP.1.1(4)). If any manipulation of the key is performed in forming the submask that will be used to form the KEK, that process shall be described in the TSS.

No explicit testing of the formation of the submask from the input password is required.

The evaluator shall verify that the iteration count for PBKDFs performed by the TOE comply with NIST SP 800-132 by ensuring that the TSS contains a description of the estimated time required to derive key material from passwords and how the TOE increases the computation time for password-based key derivation (including but not limited to increasing the iteration count).

5.2.3 HTTPS Protocol (FCS_HTTPS)

FCS_HTTPS_EXT.1	Extended: HTTPS Protocol

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

FCS_HTTPS_EXT.1.2 The TSF shall implement HTTPS using TLS (FCS_TLSC_EXT.2).

FCS_HTTPS_EXT.1.3 The TSF shall notify the application and [selection: not establish the connection, request application authorization to establish the connection, no other action] if the peer certificate is deemed invalid.

Application Note: Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

Assurance Activity:

Test 1: The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

Other tests are performed in conjunction with FCS_TLSC_EXT.2.

Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:

Test 2: The evaluator shall demonstrate that using a certificate without a valid certification path results in an application notification. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database 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 application is notified of the validation failure.

5.2.4 Initialization Vector Generation (FCS_IV)

FCS_IV_EXT.1	Extended: Initialization Vector Generation

FCS_IV_EXT.1.1 The TSF shall generate IVs in accordance with Table 14: References and IV Requirements for NIST-approved Cipher Modes.

Application Note: Table 14 lists the requirements for composition of IVs according to the 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 and data storage encryption.

Assurance Activity:

The evaluator shall examine the key hierarchy section of the TSS to ensure that the encryption of all keys is described and the formation of the IVs for each key encrypted by the same KEK meets FCS_IV_EXT.1.

5.2.5 Random Bit Generation (FCS_RBG)

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

FCS_RBG_EXT.1.1: The TSF 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); FIPS Pub 140-2 Annex C: X9.31 Appendix 2.4 using AES*].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: *a software-based noise source, TSF-hardware-based noise source*] 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.

FCS_RBG_EXT.1.3: The TSF shall be capable of providing output of the RBG to applications running on the TSF that request random bits.

Application Note: NIST Special Pub 800-90B, Appendix C describes the minimum entropy measurement that should be used immediately and will be required in future publications of this PP.

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

SP 800-90A contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used (if SP 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 CTR_DRBG are allowed.

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, 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. This option will no longer be allowed for products entering evaluation after January 1, 2016.

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

Health testing of the DRBGs is performed in conjunction with the self-tests required in FPT_TST_EXT.1.1.

For the selection in FCS_RBG_EXT.1.2, the ST author selects the appropriate number of bits of entropy that corresponds to the greatest security strength of the algorithms included in the ST. Security strength is defined in Tables 2 and 3 of NIST SP 800-57A. For example, if the implementation includes 2048-bit RSA (security strength of 112 bits), AES 128 (security strength 128 bits), and HMAC-SHA-256 (security strength 256 bits), then the ST author would select 256 bits. Seeding with 256 bits will be required for products entering evaluation after Quarter 3, 2015.

The ST author may select either software or hardware noise sources. A hardware noise source is a component that produces data that cannot be explained by a deterministic rule, due to its physical nature. In other words, a hardware based noise source generates sequences of random numbers from a physical process that cannot be predicted. For example, a sampled ring oscillator consists of an odd number of inverter gates chained into a loop, with an electrical pulse traveling from inverter to inverter around the loop. The inverters are not clocked, so the precise time required for a complete circuit around the loop varies slightly as various physical effects modify the small delay time at each inverter on the line to the next inverter. This variance results in an approximate natural frequency that contains drift and jitter over time. The output of the ring oscillator consists of the oscillating binary value sampled at a constant rate from one of the inverters – a rate that is significantly slower than the oscillator's natural frequency.

Assurance Activity:

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix E and the "Clarification to the Entropy Documentation and Assessment Annex".

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security functions described in FCS_RBG_EXT.1.3.

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

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

Implementations Conforming to FIP 140-2 Annex C

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators 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 evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators 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 evaluators ensure that the values returned by the TSF match the expected values.

The evaluators 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 evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke 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 evaluators ensure 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 RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

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

The following 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 Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.
>
> **Personalization string:** The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the

evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

5.2.6 Cryptographic Algorithm Services (FCS_SRV)

FCS_SRV_EXT.1	Extended: Cryptographic Algorithm Services

FCS_SRV_EXT.1.1 The TSF shall provide a mechanism for applications to request the TSF to perform the following cryptographic operations:

- All mandatory and selected algorithms in FCS_CKM.2(1)
- The following algorithms in FCS_COP.1(1): AES-CBC, [selection: AES Key Wrap, AES Key Wrap with Padding, AES-GCM, AES-CCM, no other modes]
- All mandatory and selected algorithms in FCS_COP.1(3)
- All mandatory and selected algorithms in FCS_COP.1(2)
- All mandatory and selected algorithms in FCS_COP.1(4)

[selection:

- *All mandatory and selected algorithms in FCS_CKM.1(1),*
- *The selected algorithms in FCS_COP.1(5),*
- *No other cryptographic operations*].

Application Note: For each of the listed FCS components in the bulleted list, the intent is that the TOE will make available all algorithms specified for that component in the ST. For example, if for FCS_COP.1(2) the ST author selects SHA-256, then the TOE would have to make available an interface to perform SHA-1 (the "mandatory" portion of FCS_COP.1.1(2)) and SHA-256 (the "selected" portion of FCS_COP.1.1(2)). The exception is for FCS_COP.1(1), which does not require the TOE to make available AES_CCMP, AES_XTS, AES_GCMP-256, or AES_CCMP_256 even though they may be implemented to perform TSF-related functions. However, the ST author is expected to select (for FCS_COP.1(1) selection in this component) algorithms that match those selected in the ST for the FCS_COP.1(1) component.

Assurance Activity:

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security functions (cryptographic algorithms) described in these requirements.

The evaluator shall write, or the developer shall provide access to, an application that requests cryptographic operations by the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. This application may be used to assist in verifying the cryptographic operation assurance activities for the other algorithm services requirements.

5.2.7 Cryptographic Key Storage (FCS_STG)

This section describes how keys are protected. All keys must ultimately be protected by a REK, and may optionally be protected by the user's password. Each key's confidentiality and integrity must be protected. This section also describes the secure key storage services to be

provided by the Mobile Device for use by applications and users, applying the same level of protection for these keys as keys internal to the OS.

5.2.7.1 Secure Key Storage

FCS_STG_EXT.1	Extended: Cryptographic Key Storage

FCS_STG_EXT.1.1 The TSF shall provide [selection: *hardware, hardware-isolated, software-based*] secure key storage for asymmetric private keys and [selection: *symmetric keys, persistent secrets, no other keys*].

Application Note: If this secure key storage is implemented fully in hardware, the ST author shall select "hardware". Key storage is considered implemented fully in hardware if the raw bytes of the key are not available to the any software on the TSF, so that the OS may only request operations by the key. A hardware key store can be exposed to the TSF through a variety of interfaces, including embedded on the motherboard, USB, microSD, and Bluetooth.

If the secure key storage is implemented in a separate processor execution environment, isolated form the rich OS in both processing and memory, the ST author shall select "hardware-isolated."

If the secure key storage is implemented in software that is protected as required by FCS_STG_EXT.2, the ST author shall select "software-based." If "software-based" is selected, the ST author shall select "all software-based key storage" in FCS_STG_EXT.2.

Support for secure key storage for all symmetric keys and persistent secrets will be required in future revisions.

FCS_STG_EXT.1.2 The TSF shall be capable of importing keys/secrets into the secure key storage upon request of [selection: *the user, the administrator*] and [selection: *applications running on the TSF, no other subjects*].

Application Note: If the ST Author selects only user, the ST Author shall select function 11 in FMT_MOF_EXT.1.1.

FCS_STG_EXT.1.3 The TSF shall be capable of destroying keys/secrets in the secure key storage upon request of [selection: *the user, the administrator*].

Application Note: If the ST Author selects only user, the ST Author shall select function 12 in FMT_MOF_EXT.1.1.

FCS_STG_EXT.1.4 The TSF shall have the capability to allow only the application that imported the key/secret the use of the key/secret. Exceptions may only be explicitly authorized by [selection: *the user, the administrator, a common application developer*].

Application Note: If the ST Author selects user or administrator, the ST Author must also select function 34 in FMT_SMF_EXT.1.1. If the ST Author selects only user, the ST Author shall select function 34 in FMT_MOF_EXT.1.1.

FCS_STG_EXT.1.5 The TSF shall allow only the application that imported the key/secret to request that the key/secret be destroyed. Exceptions may only be explicitly authorized by [selection: *the user, the administrator, a common application developer*].

Application Note: If the ST Author selects user or administrator, the ST Author must also select function 35 in FMT_SMF_EXT.1.1. If the ST Author selects only user, the ST Author shall select function 35 in FMT_MOF_EXT.1.1.

Assurance Activity:

The assurance activity for this component entails examination of the ST's TSS to determine that the TOE's implements the required secure key storage. The evaluator shall ensure that the TSS contains a description of the key storage mechanism that justifies the selection of "hardware", "hardware-isolated", or "software-based."

The evaluator shall review the AGD guidance to determine that it describes the steps needed to import or destroy keys/secrets. The evaluator shall also verify that the API documentation provided according to Section 6.2.1 includes the security functions (import, use, and destruction) described in these requirements. The API documentation shall include the method by which applications restrict access to their keys/secrets in order to meet FCS_STG_EXT.1.4.

The evaluator shall test the functionality of each security function:

Test 1: The evaluator shall import keys/secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that generates a key/secret of each supported type and calls the import functions. The evaluator shall verify that no errors occur during import.

Test 2: The evaluator shall write, or the developer shall provide access to, an application that uses an imported key/secret:

- For RSA, the secret shall be used to sign data.
- For ECDSA, the secret shall be used to sign data

In the future additional types will be required to be tested:

- For symmetric algorithms, the secret shall be used to encrypt data.
- For persistent secrets, the secret shall be compared to the imported secret.

The evaluator shall repeat this test with the application-imported keys/secrets and a different application's imported keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to use the key/secret imported by the user or by a different application:

- The evaluator shall deny the approvals to verify that the application is not able to use the key/secret as described.
- The evaluator shall repeat the test, allowing the approvals to verify that the application is able to use the key/secret as described.

If the ST Author has selected "common application developer", this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

Test 3: The evaluator shall destroy keys/secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that destroys an imported key/secret.

The evaluator shall repeat this test with the application-imported keys/secrets and a different application's imported keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to destroy the key/secret imported by the administrator or by a different application:

- The evaluator shall deny the approvals and verify that the application is still able to use the key/secret as described.
- The evaluator shall repeat the test, allowing the approvals and verifying that the application is no longer able to use the key/secret as described.

If the ST Author has selected "common application developer", this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

5.2.7.2 Encryption of Stored Keys

FCS_STG_EXT.2	Extended: Encrypted Cryptographic Key Storage

FCS_STG_EXT.2.1 The TSF shall encrypt all DEKs and KEKs and [selection*: long-term trusted channel key material, all software-based key storage, no other keys*] by KEKs that are [*selection:*

1) *Protected by the REK with [selection:*
   a. *encryption by a REK,*
   b. *encryption by a KEK chaining to a REK],*
2) *Protected by the REK and the password with [selection:*
   a. *encryption by a REK and the password-derived KEK,*
   b. *encryption by a KEK chaining to a REK and the password-derived KEK]*

].

Application Notes: The ST author must select "all software-based key storage" if "software-based" is selected in FCS_STG_EXT.1.1. If the ST author selects "hardware" or "hardware-isolated" in FCS_STG_EXT.1.1, the secure key storage is not subject to this requirement.

REKs are not subject to this requirement.

A REK and the password-derived KEK may be combined to form a combined KEK (as described in FCS_CKM_EXT.3) in order to meet this requirement.

Sensitive data is protected by the REK and the password. If FDP_DAR_EXT.2 is included in the main body, this sensitive data includes some or all user or enterprise data. The software-based key storage shall either all be sensitive (protected by the REK and the password) or

shall allow users and applications to mark the key as sensitive (protected by the REK and the password) according to FDP_DAR_EXT.2.1.

All keys must ultimately be protected by the REK. Sensitive data must be protected by the password (selection 2). In particular, Figure 3 has KEKs protected according to these requirements: DEK_1 meets 2a and would be appropriate for sensitive data, DEK_2 meets 1b and would not be appropriate for sensitive data, K_1 meets 1a and is not considered a sensitive key, and K_2 meets 2b and is considered a sensitive key.

Long-term trusted channel key material include IPsec and WiFi pre-shared keys and Bluetooth keys, such as link keys, Long Term Keys (LTKs), Connection Signature Resolving Keys (CSRKs), Identity Resolving Keys (IRKs), and Generic AMP Link Keys. These keys shall not be protected by the password, as they may be necessary in the locked state. Encryption of long-term trusted channel key material will be required for products entering into evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall review the TSS to determine that the TSS includes key hierarchy description of the protection of each DEK for data-at-rest, of software-based key storage, of long-term trusted channel keys, and of KEK related to the protection of the DEKs, long-term trusted channel keys, and software-based key storage. This description must include a diagram illustrating the key hierarchy implemented by the TOE in order to demonstrate that the implementation meets FCS_STG_EXT.2. The description shall indicate how the functionality described by FCS_RBG_EXT.1 is invoked to generate DEKs (FCS_CKM_EXT.2), the key size (FCS_CKM_EXT.2 and FCS_CKM_EXT.3) for each key, how each KEK is formed (generated, derived, or combined according to FCS_CKM_EXT.3), the integrity protection method for each encrypted key (FCS_STG_EXT.3), and the IV generation for each key encrypted by the same KEK (FCS_IV_EXT.1). More detail for each task follows the corresponding requirement.

FCS_STG_EXT.2.2 DEKs and KEKs and [selection*: long-term trusted channel key material, all software-based key storage, no other keys*] shall be encrypted using AES in the [selection: *Key Wrap (KW) mode, Key Wrap with Padding (KWP) mode, GCM, CCM, CBC mode*].

Application Note: Either 128-bit or 256-bit (or both) are allowed; the ST author makes the selection appropriate for the device. This requirement refers only to KEKs as defined this PP and does not refer to those KEKs specified in other standards.

Assurance Activity:

The evaluator shall examine the key hierarchy section of the TSS to ensure that each key (DEKs, software-based key storage, and KEKs) is encrypted by keys of equal or greater security strength using one of the selected modes.

The evaluator shall examine the key hierarchy description in the TSS section to verify that each DEK and software-stored key is encrypted according to FCS_STG_EXT.2.

5.2.7.3 Integrity of Stored Keys

FCS_STG_EXT.3	Extended: Integrity of encrypted key storage

FCS_STG_EXT.3.1 The TSF shall protect the integrity of any encrypted DEKs and KEKs and [selection*: long-term trusted channel key material, all software-based key storage, no other keys*] by [selection:

- [*selection: GCM, CCM, Key Wrap, Key Wrap with Padding*] *cipher mode for encryption according to FCS_STG_EXT.2;*
- *a hash (FCS_COP.1(2)) of the stored key that is encrypted by a key protected by FCS_STG_EXT.2;*
- *a keyed hash (FCS_COP.1(4)) using a key protected by a key protected by FCS_STG_EXT.2;*
- *a digital signature of the stored key using an asymmetric key protected according to FCS_STG_EXT.2*].

FCS_STG_EXT.3.2 The TSF shall verify the integrity of the [selection***:*** *hash, digital signature, MAC*] of the stored key prior to use of the key.

Application Note: This requirement is not applicable to derived keys that are not stored.

It is not expected that a single key will be protected from corruption by multiple of these methods; however, a product may use one integrity-protection method for one type of key and a different method for other types of keys. The explicit Assurance Activities for each of the options will be addressed in each of the requirements (FCS_COP.1.1(2), FCS_COP.1.1(4)).

Assurance Activity:

The evaluator shall examine the key hierarchy description in the TSS section to verify that each encrypted key is integrity protected according to one of the options in FCS_STG_EXT.3.

5.2.8 TLS Client Protocol (FCS_TLSC)

5.2.8.1 EAP-TLS Client Protocol

FCS_TLSC_EXT.1	Extended: EAP TLS Protocol

FCS_TLSC_EXT.1.1 The TSF shall implement TLS 1.0 and [selection: *TLS 1.1 (RFC 4346), TLS 1.2 (RFC 5246), no other TLS version*] supporting the following ciphersuites: [

- Mandatory Ciphersuites:
  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
- [selection: Optional Ciphersuites:
  - *TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246*

- *TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246*
- *TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246*
- *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492*
- *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492*
- *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492*
- *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492*
- *TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246*
- *TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246*
- *TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256 as defined in RFC 5246*
- *TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246*
- *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*
- *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*
- *no other ciphersuite*]].

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 listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

If an elliptic-curve ciphersuite is selected, FCS_TLSC_EXT.1.5 shall be included in the ST.

TLS 1.2 is the preferred protocol; however, TLS 1.0 is currently required in order to ensure compliance with RFC 5216. TLS 1.0 and TLS 1.1 are currently allowed due to lack of support for TLS 1.2. TLS 1.0 and TLS 1.1 do not have the extensions necessary to assure a connection with security strength of 112-bits or better.

TLS 1.2 will be required for EAP-TLS for products entering into evaluation after Quarter 3, 2015. These requirements will be revisited as new TLS versions are standardized by the IETF.

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 include 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.

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

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

Test 5: The evaluator shall perform the following modifications to the traffic:

- Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
- Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.
- Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
- Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
- Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

FCS_TLSC_EXT.1.2 The TSF shall verify that the server certificate presented for EAP-TLS [selection: *chains to one of the specified CAs, contains the specified FQDN of the acceptable authentication server certificate*.].

Application Note: The CA or FQDN is specified according to FMT_SMF_EXT.1 function 7.a.

Assurance Activity:

The evaluator shall check that the AGD guidance contains instructions for the administrator to configure the list of Certificate Authorities that are allowed to sign certificates or to configure the FQDN of the authentication server certificate that will be accepted by the TOE in the EAP-TLS exchange.

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

Test 1: Following the guidance provided by the AGD guidance, a CA or an FQDN will be configured as "acceptable" for authentication server certificates and then the evaluator will start a wireless connection and verify that the wireless client is able to successfully connect. The evaluator will then configure the system such that an otherwise valid certificate is signed by a CA that is not allowed by the TOE or presents a FQDN that is not allowed by the TOE. Attempts to authenticate to an authentication server presenting such a certificate should result in the connection being refused. If the TOE supports both methods of limiting the acceptable authentication servers, the evaluator shall repeat this test twice, once with each method.

FCS_TLSC_EXT.1.3 The TSF shall not establish a trusted channel if the peer certificate is invalid.

Application Note: Validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

For TLS connections, this channel shall not be established if the peer certificate is invalid. The HTTPS protocol (FCS_HTTPS_EXT.1) requires different behavior, though HTTPS is implemented over TLS. This element addresses non-HTTPS TLS connections.

Assurance Activity:

The evaluator shall perform the following test:

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 to the Trust Anchor Database 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.

FCS_TLSC_EXT.1.4 The TSF shall support mutual authentication using X.509v3 certificates.

Application Note: The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1. This requirement adds that this use must include the client must be capable of presenting a certificate to a TLS server for TLS mutual authentication.

Assurance Activity:

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

The evaluator shall also perform the following test:

Test 1: The evaluator shall perform the following modification to the traffic:

- Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify the connection is unsuccessful.

5.2.8.2 TLS Client Protocol

FCS_TLSC_EXT.2	Extended: TLS Protocol

FCS_TLSC_EXT.2.1 The TSF shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites: [

- Mandatory Ciphersuites:
  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
- [selection: Optional Ciphersuites:
  - *TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246*
  - *TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246*
  - *TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246*
  - *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492*
  - *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492*
  - *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492*
  - *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492*
  - *TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246*
  - *TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246*
  - *TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256 as defined in RFC 5246*
  - *TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246*
  - *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*
  - *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*
  - *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*
  - *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*
  - *no other ciphersuite*]].

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 listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

If an elliptic-curve ciphersuite is selected, FCS_TLSC_EXT.2.5 shall be included in the ST.

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

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 will be required for products entering evaluation after Quarter 3, 2015.

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 include 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.

The evaluator shall write, or the ST author shall provide, an application for the purposes of testing TLS. 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 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 send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

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

Test 5: The evaluator shall perform the following modifications to the traffic:

- Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
- Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if

using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.
- Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- *(conditional)* If a ECDHE or DHE ciphersuite is selected, modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
- Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
- Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

FCS_TLSC_EXT.2.2 The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

Application Note: The rules for verification of identify are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the application service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server's certificate.

The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged as against best practices but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the assurance activity.

Assurance Activity:

The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported (e.g Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS. In particular, the AGD guidance should describe the API used by applications for configuring the reference identifier.

The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.

Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contains the SAN extension. The evaluator shall verify that the connection succeeds.

Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

- The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.
- The evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails.
- The evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.

Test 6: *[conditional]* If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

Test 7: *[conditional]* If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

FCS_TLSC_EXT.2.3 The TSF shall not establish a trusted channel if the peer certificate is invalid.

Application Note: Validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

For TLS connections, this channel shall not be established if the peer certificate is invalid. The HTTPS protocol (FCS_HTTPS_EXT.1) requires different behavior, though HTTPS is implemented over TLS. This element addresses non-HTTPS TLS connections.

Assurance Activity:

The evaluator shall perform the following test:

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 to the Trust Anchor Database 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.

FCS_TLSC_EXT.2.4 The TSF shall support mutual authentication using X.509v3 certificates.

Application Note: The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1. This requirement adds that this use must include the client must be capable of presenting a certificate to a TLS server for TLS mutual authentication.

Assurance Activity:

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

The evaluator shall also perform the following test:

Test 1: The evaluator shall perform the following modification to the traffic:

- Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify the connection is unsuccessful.

5.3 Class: User Data Protection (FDP)

5.3.1 Access Control (FDP_ACF)

FDP_ACF_EXT.1	Extended: Security access control

FDP_ACF_EXT.1.1 The TSF shall provide a mechanism to restrict the system services that are accessible to an application.

Application Note: Examples of system services to which this requirement applies include:

- obtain data from camera and microphone input devices
- get current GPS location
- retrieve credentials from system-wide credential store
- retrieve contacts list / address book
- retrieve stored pictures
- retrieve text messages
- retrieve emails
- retrieve device identifier information
- obtain network access

Assurance Activity:

The evaluator shall ensure the TSS lists all system services available for use by an application. The evaluator shall also ensure that the TSS describes how applications interface with these system services, and means by which these system services are protected by the TSF.

The TSS shall describe which of the following categories each system service falls in:

1) No applications are allowed access
2) Privileged applications are allowed access
3) Applications are allowed access by user authorization
4) All applications are allowed access

Privileged applications include any applications developed by the TSF developer. The TSS shall describe how privileges are granted to third-party applications. For both types of privileged applications, the TSS shall describe how and when the privileges are verified and how the TSF prevents unprivileged applications from accessing those services.

For any services for which the user may grant access, the evaluator shall ensure that the TSS identifies whether the user is prompted for authorization when the application is installed, or during runtime. The evaluator shall ensure that the operational user guidance contains instructions for restricting application access to system services.

Assurance Activity Note: The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall write, or the developer shall provide, applications for the purposes of the following tests.

Test 1: For each system service to which no applications are allowed access, the evaluator shall attempt to access the system service with a test application and verify that the application is not able to access that system service.

Test 2: For each system service to which only privileged applications are allowed access, the evaluator shall attempt to access the system service with an unprivileged application and verify that the application is not able to access that system service. The evaluator shall attempt to access the system service with a privileged application and verify that the application can access the service.

Test 3: For each system service to which the user may grant access, the evaluator shall attempt to access the system service with a test application. The evaluator shall ensure that either the system blocks such accesses or prompts for user authorization. The prompt for user authorization may occur at runtime or at installation time, and should be consistent with the behavior described in the TSS.

Test 4: For each system service listed in the TSS that is accessible by all applications, the evaluator shall test that an application can access that system service.

FDP_ACF_EXT.1.2 The TSF shall provide an access control policy that prevents [selection: *application processes, groups of application processes*] from accessing [selection: *all, private*] data stored by other [selection: *application processes, groups of application processes*]. Exceptions may only be explicitly authorized for such sharing by [selection*: the user, the administrator, a common application developer*].

Application Note: Application process groups may be designated Enterprise, Managed, Work Environment, Personal, Non-Managed, or Personal Environment. Private data is defined as data that is accessible only by the application which wrote it. Private data is distinguished from data which an application may, by design, write to shared storage areas.

Assurance Activity:

The evaluator shall examine the TSS to verify that it describes which data sharing is permitted between applications, which data sharing is not permitted, and how disallowed sharing is prevented.

Test: The evaluator shall write, or the developer shall provide, two applications, one which saves data containing a unique string and the other which attempts to access that data. If "groups of applications" is selected, the applications shall be placed into different groups. If "private data" is selected, the application shall not write to a designated shared storage area. The evaluator shall verify that the second application is unable to access the stored unique string. The evaluator shall grant access, either as a user, the administrator, or by using a third application with a common application developer to the first, and verify that the application is able to access the stored unique string.

5.3.2 Data-At-Rest Protection (FDP_DAR)

FDP_DAR_EXT.1	**Extended: Protected Data Encryption**

FDP_DAR_EXT.1.1 Encryption shall cover all protected data.

Application Note: As defined in 1.2 Glossary, protected data is all non-TSF data, including all user or enterprise data.

FDP_DAR_EXT.1.2 Encryption shall be performed using DEKs with AES in the [selection: *XTS, CBC, GCM*] mode with key size [selection: *128,256*] bits.

Application Note: Use of 256 bit key sizes will be required for products entering evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall verify that the TSS section of the ST indicates which data is protected by the DAR implementation and what data is considered TSF data. The evaluator shall ensure that this data includes all protected data.

The evaluator shall review the AGD guidance to determine that the description of the configuration and use of the DAR protection does not require the user to perform any actions beyond configuration and providing the authentication credential. The evaluator shall also review the AGD guidance to determine that the configuration does not require the user to identify encryption on a per-file basis.

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

Test 1: The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create user data (non-system) either by creating a file or by using an application. The evaluator shall use a tool provided by the developer to verify that this data is encrypted when the product is powered off, in conjunction with Test 1 for FIA_UAU_EXT.1.

5.3.3 Subset Information Flow Control – VPN (FDP_IFC)

FDP_IFC_EXT.1 Extended: Subset information flow control

FDP_IFC_EXT.1.1 The TSF shall [selection: *provide an interface to VPN clients to enable all IP traffic (other than IP traffic required to establish the VPN connection) to flow through the IPsec VPN client; enable all IP traffic (other than IP traffic required to establish the VPN connection) to flow through the IPsec VPN client*].

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

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

The ST author shall select the second option if the TSF implements a native VPN client (IPsec is selected in FTP_ITC_EXT.1). If the native VPN client is to be validated (IPsec is selected in FTP_ITC_EXT.1 and the TSF is validated against the "Protection Profile for

IPsec Virtual Private Network (VPN) Clients"), the ST author shall also include FDP_IFC_EXT from the VPN Client Protection Profile.

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

Assurance Activity:

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

The evaluator shall verify that one (or more) of the following options is addressed by the documentation:

- The description above indicates that if a VPN client is enabled, all configurations route all Data Plane traffic through the tunnel interface established by the VPN client.
- The AGD guidance describes how the user and/or administrator can configure the TSF to meet this requirement.
- The API documentation includes a security function that allows a VPN client to specify this routing.

Test 1: If the ST author identifies any differences in the routing between WiFi and cellular protocols, the evaluator shall repeat this test with a base station implementing one of the identified cellular protocols.

Step 1 - The evaluator shall enable a WiFi configuration as described in the AGD guidance (as required by FTP_ITC_EXT.1). The evaluator shall use a packet sniffing tool between the wireless access point and an Internet-connected network. The evaluator shall turn on the sniffing tool and perform actions with the device such as navigating to websites, using provided applications, and accessing other Internet resources. The evaluator shall verify that the sniffing tool captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 2 - The evaluator shall configure an IPsec VPN client that supports the routing specified in this requirement, and if necessary, configure the device to perform the routing specified as described in the AGD guidance. The evaluator shall turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step. The evaluator shall verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 3 - The evaluator shall examine the traffic from both step one and step two to verify that all Data Plane traffic is encapsulated by IPsec. The evaluator shall examine the Security

Parameter Index (SPI) value present in the encapsulated packets captured in Step two from the TOE to the Gateway and shall verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from the TOE to the Gateway. The evaluator shall be aware that IP traffic on the cellular baseband outside of the IPsec tunnel may be emanating from the baseband processor and shall verify with the manufacturer that any identified traffic is not emanating from the application processor.

Step 4 - The evaluator shall perform an ICMP echo from the TOE to the IP address of another device on the local wireless network and shall verify that no packets are sent using the sniffing tool. The evaluator shall attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local wireless network, and shall verify that the TOE discards them.

5.3.4 Certificate Data Storage (FDP_STG)

FDP_STG_EXT.1 Extended: User Data Storage

FDP_STG_EXT.1.1 The TSF shall provide protected storage for the Trust Anchor Database.

Assurance Activity:

The evaluator shall ensure the TSS describes the Trust Anchor Database implemented that contain certificates used to meet the requirements of this PP. This description shall contain information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access (for example, unix permissions) in accordance with the permissions established in FMT_SMF_EXT.1 and FMT_MOF_EXT.1.1.

5.3.5 Inter-TSF User Data Protected Channel (FDP_UPC)

FDP_UPC_EXT.1 Extended: Inter-TSF user data transfer protection

FDP_UPC_EXT.1.1 The TSF provide a means for non-TSF applications executing on the TOE to use TLS, HTTPS, Bluetooth BR/EDR, and [selection: *DTLS, Bluetooth LE, no other protocol*] to provide a protected communication channel between the non-TSF application and another IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

Application Note: The intent of this requirement is that one of the selected protocols is available for use by user applications running on the device for use in connecting to distant-end services that are not necessarily part of the enterprise infrastructure. It should be noted that the FTP_ITC_EXT requires that all TSF communications (meaning communications from the device to a gateway) be protected using the protocols indicated in that requirement, so the protocols required by this component ride "on top of" those listed in FTP_ITC_EXT.

It should also be noted that some applications are part of the TSF, and FTP_ITC_EXT requires that TSF applications be protected by at least one of the protocols in first selection in FTP_ITC_EXT.1. It is not required to have two different implementations of a protocol, or

two different protocols, to satisfy both this requirement (for non-TSF apps) and FTP_ITC_EXT (for TSF apps), as long as the services specified are provided.

The ST author shall list which trusted channel protocols are implemented by the Mobile Device for use by non-TSF apps. If the ST author selects IPsec, the TSF shall be validated against the "Protection Profile for IPsec Virtual Private Network (VPN) Clients." Annex B contains the requirements for implementing each of the other optional trusted channel protocols. The ST author must include the security functional requirements for the trusted channel protocol selected in **FDP_UPC_EXT.1** in the main body of the ST.

FDP_UPC_EXT.1.2 The TSF shall permit the non-TSF applications to initiate communication via the trusted channel.

Assurance Activity:

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security functions (protection channel) described in these requirements, and verify that the APIs implemented to support this requirement include the appropriate settings/parameters so that the application can both provide and obtain the information needed to assure mutual identification of the endpoints of the communication as required by this component. The evaluator shall write, or the developer shall provide access to, an application that requests protected channel services by the TSF. The evaluator shall verify that the results from the protected channel match the expected results according to the API documentation. This application may be used to assist in verifying the protected channel assurance activities for the protocol requirements.

The evaluator shall examine the TSS to determine that it describes that all protocols listed in the TSS are specified and included in the requirements in the ST. The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocol(s) selected for use by the applications. The evaluator shall also perform the following tests:

Test 1: The evaluators shall ensure that the application is able to initiate communications with an external IT entity using each protocol specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 2: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext.

5.4 Class: Identification and Authentication (FIA)

5.4.1 Authentication Failures (FIA_AFL)

FIA_AFL_EXT.1	Authentication failure handling

FIA_AFL_EXT.1.1 The TSF shall detect when a configurable positive integer within [assignment: *range of acceptable values*] of unsuccessful authentication attempts occur related to last successful authentication by that user.

Application Note: The positive integer is configured according to FMT_SMF_EXT.1.1 function 2.c. If the TOE implements multiple Password Authentication Factor interfaces (for

example, a DAR decryption interface, a lockscreen interface, an auxiliary boot mode interface), this component applies to all available interfaces.

FIA_AFL_EXT.1.2 When the defined number of unsuccessful authentication attempts has been surpassed, the TSF shall perform wipe of all protected data.

Application Note: Wipe is performed in accordance with FCS_CKM_EXT.5.

FIA_AFL_EXT.1.3 The TSF shall maintain the number of unsuccessful authentication attempts that have occurred upon power off.

Application Note: The TOE may implement a Password Authentication Factor interface that precedes another Password Authentication Factor interface in the boot sequence (for example, a volume DAR decryption interface which precedes the lockscreen interface) before the user can access the device. In this situation, because the user must successfully authenticate to the first interface to access the second, the number of unsuccessful authentication attempts need not be maintained for the second interface.

Assurance Activity:

The evaluator shall ensure that the TSS describes that a value corresponding to the number of unsuccessful authentication attempts since the last successful authentication is kept for each user for each Password Authentication Factor interface. The evaluator shall ensure that this description also includes if and how this value is maintained when the TOE is powered off. The evaluator shall ensure that if the value is not maintained, the interface is after another interface in the boot sequence for which the value is maintained.

The evaluator shall verify that the AGD guidance describes how the administrator configures the maximum number of unsuccessful authentication attempts.

The evaluator shall perform the following tests for each available authentication factor interface:

Test 1: The evaluator shall configure according to the AGD guidance the device with a maximum number of unsuccessful authentication attempts. The evaluator shall enter the locked state and enter incorrect passwords until the wipe occurs. The evaluator shall verify that the number of password entries corresponds to the configured maximum and that the wipe is implemented.

Test 2: The evaluator shall repeat test one, but shall power off (by removing the battery, if possible) the TOE between unsuccessful authentication attempts. The evaluator shall verify that the total number of password entries corresponds to the configured maximum and that the wipe is implemented. Alternatively, if the number of authentication failures is not maintained for the interface under test, the evaluator shall verify that upon booting the TOE between unsuccessful authentication attempts another authentication factor interface is presented before the interface under test.

5.4.2 Bluetooth Authorization and Authentication (FIA_BLT)

FIA_BLT_EXT.1	Extended: Bluetooth User Authorization

FIA_BLT_EXT.1.1 The TSF shall require explicit user authorization before pairing with a remote Bluetooth device.

Application Note: User authorization includes explicit actions like affirming the remote device's name, expressing an intent to connect to the remote device, and entering relevant pairing information (e.g. PINs, numeric codes, or "yes/no" responses). The user must have to explicitly permit all pairing attempts, even when bonding is not taking place.

Because explicit user action must be required to permit pairing, it must not be possible for applications to programmatically enter pairing information (e.g. PINs, numeric codes, or "yes/no" responses) during the pairing process. The absence of public APIs for programmatic authorization is not sufficient to meet this requirement; hidden or private APIs must be absent as well.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it contains a description of when user permission is required for Bluetooth pairing, and that this description mandates explicit user authorization via manual input for all Bluetooth pairing, including application use of the Bluetooth trusted channel and situations where temporary (non-bonded) connections are formed. The evaluator shall examine the API documentation provided according to Section 6.2.1 and verify that this API documentation does not include any API for programmatic entering of pairing information (e.g. PINs, numeric codes, or "yes/no" responses) intended to bypass manual user input during pairing.

The evaluator shall examine the AGD guidance to verify that these user authorization screens are clearly identified and instructions are given for authorizing Bluetooth pairings.

The evaluator shall perform the following test:

Test 1: The evaluator shall perform the following steps:

Step 1 - Initiate pairing with the TOE from a remote Bluetooth device that requests no man-in-the-middle protection, no bonding, and claims to have NoInputNoOutput input-output (IO) capability. (Such a device will attempt to evoke behavior from the TOE that represents the minimal level of user interaction that the TOE supports during pairing.) Step 2 - Verify that the TOE does not permit any Bluetooth pairing without explicit authorization from the user (e.g. the user must have to minimally answer "yes" or "allow" in a prompt).

5.4.3 Port Access Entity Authentication (FIA_PAE)

FIA_PAE_EXT.1	Extended: PAE Authentication

FIA_PAE_EXT.1 The TSF shall conform to IEEE Standard 802.1X for a Port Access Entity (PAE) in the "Supplicant" role.

Application Note: This requirement covers the TSF's role as the supplicant in an 802.1X authentication exchange. If the exchange is completed successfully, the TSF will derive the PMK as a result of the EAP-TLS (or other appropriate EAP exchange) and perform the 4-way handshake with the wireless access system (authenticator) to begin 802.11 communications.

As indicated previously, there are at least two communication paths present during the exchange; one with the wireless access system and one with the authentication server that uses the wireless access system as a relay. The TSF establishes an EAP over LAN (EAPOL) connection with the wireless access system as specified in 802.1X-2010. The TSF and authentication server establish an EAP-TLS session (RFC 5216).

The point of performing 802.1X authentication is to gain access to the network (assuming the authentication was successful and that all 802.11 negotiations are performed successfully); in the terminology of 802.1X, this means the TSF will gain access to the "controlled port" maintained by the wireless access system.

Assurance Activity:

The evaluator shall perform the following tests:

- *Test 1:* The evaluator shall demonstrate that the TOE has no access to the test network. After successfully authenticating with an authentication server through a wireless access system, the evaluator shall demonstrate that the TOE does have access to the test network.
- *Test 2:* The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid client certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.
- *Test 3:* The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid authentication server certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.

5.4.4 Password Management (FIA_PMG)

FIA_PMG_EXT.1	**Extended: Password Management**

FIA_PMG_EXT.1.1 The TSF shall support the following for the Password Authentication Factor:

1. Passwords shall be able to be composed of any combination of [selection: *upper and lower case letters, [assignment: a character set of at least 52 characters*]], numbers, and special characters: [selection: *"!", "@", "#", "$", "%", "^", "&", "*", "(", ")", assignment: other characters*];

2. Password length up to [assignment: *an integer greater than or equal to 14*] characters shall be supported.

Application Note: While some corporate policies require passwords of 14 characters or better, the use of a REK for DAR protection and key storage protection and the anti-hammer requirement (FIA_TRT_EXT.1) addresses the threat of attackers with physical access using much smaller and less complex passwords.

The ST author selects the character set: either the upper and lower case Basic Latin letters or another assigned character set containing at least 52 characters. The assigned character set must be well defined: either according to an international encoding standard (such as Unicode) or defined in the assignment by the ST author. The ST author also selects the special characters that are supported by TOE; they may optionally list additional special characters supported using the assignment.

Assurance Activity:

The evaluator shall examine the operational guidance to determine that it provides guidance to security administrators on the composition of strong passwords, and that it provides instructions on setting the minimum password length. The evaluator shall also perform the following tests. Note that one or more of these tests can be performed with a single test case.

Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

5.4.5 Authentication Throttling (FIA_TRT)

FIA_TRT_EXT.1	Extended: Authentication Throttling

FIA_TRT_EXT.1.1 The TSF shall limit automated user authentication attempts by [selection: *preventing authentication via an external port, enforcing a delay between incorrect authentication attempts*]. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

Application Note: The user authentication attempts in this requirement are attempts to guess the Password Authentication Factor. The developer can implement the timing of the delays in the requirements using unequal or equal timing of delays.

The minimum delay specified in this requirement provides defense against password brute forcing; for example, the expected time to find a randomly generated password of 4 characters (utilizing the minimum character set of 63 characters) is 4 ½ days, and the time for 5 characters is over 287 days.

Assurance Activity:

The evaluator shall verify that the TSS describes the method by which authentication attempts are not able to be automated. The evaluator shall ensure that the TSS describes either how the TSF disables authentication via external interfaces (other than the ordinary user interface) or how authentication attempts are delayed in order to slow automated entry and shall ensure that this delay totals at least 500 milliseconds over 10 attempts.

5.4.6 User Authentication (FIA_UAU)

5.4.6.1 Protected Authentication Feedback

FIA_UAU.7	Protected authentication feedback

FIA_UAU.7.1 The TSF shall provide only [***obscured feedback to the device's display***] to the user while the authentication is in progress.

Application Note: The TSF may briefly (1 second or less) display each character or provide an option to allow the user to unmask the password; however, the password must be obscured by default.

Assurance Activity:

The evaluator shall ensure that the TSS describes the means of obscuring the password entry. The evaluator shall verify that any configuration of this requirement is addressed in the AGD guidance and that the password is obscured by default.

Test: The evaluator shall enter passwords on the device, including at least the Password Authentication Factor at lockscreen, and verify that the password is not displayed on the device.

5.4.6.2 Authentication for Cryptographic Operation

FIA_UAU_EXT.1	Extended: Authentication for Cryptographic Operation

FIA_UAU_EXT.1.1 The TSF shall require the user to present the Password Authentication Factor prior to decryption of protected data and encrypted DEKs, KEKs and [selection: *long-term trusted channel key material, all software-based key storage, no other keys*] at startup.

Application Note: The intent of this requirement is to prevent decryption of protected data before the user has authorized to the device using the Password Authentication Factor. The Password Authentication Factor is also required in order derive the key used to decrypt sensitive data (see 1.2 Glossary and Appendix D.3.3), which includes software-based secure key storage.

The ST author shall select long-term trusted channel key material or software-based key storage in correspondence with FCS_STG_EXT.2.1.

Assurance Activity:

The evaluator shall verify that the TSS section of the ST describes the process for decrypting protected data and keys. The evaluator shall ensure that this process requires the user to enter a Password Authentication Factor and, in accordance with FCS_CKM_EXT.3, derives a KEK which is used to protect the software-based secure key storage and (optionally) DEK(s) for sensitive data, in accordance with FCS_STG_EXT.2.

The following tests may be performed in conjunction with FDP_DAR_EXT.1 and FDP_DAR_EXT.2.

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

Test 1: The evaluator shall enable encryption of protected data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as protected data.

The evaluator shall reboot the device, use a tool provided by developer to search for the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by the developer to access the unique string amongst the application data, and verify that the unique string can be found.

Test 2: [conditional] The evaluator shall require user authentication according to the AGD guidance. The evaluator shall store a key in the software-based secure key storage.

The evaluator shall lock the device, use a tool provided by developer to access the key amongst the stored data, and verify that the key cannot be retrieved or accessed. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the key, and verify that the key can be retrieved or accessed.

Test 3: [conditional] The evaluator shall enable encryption of sensitive data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as sensitive data.

The evaluator shall lock the device, use a tool provided by developer to attempt to access the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the unique string amongst the application data, and verify that the unique string can be retrieved.

5.4.6.3 Timing of Authentication

FIA_UAU_EXT.2	Extended: Timing of Authentication

FIA_UAU_EXT.2.1 The TSF shall allow [selection: [assignment: *list of* actions], *no actions*] on behalf of the user to be performed before the user is authenticated.

FIA_UAU_EXT.2.2 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

Assurance Activity:

The evaluator shall verify that the TSS describes the actions allowed by unauthorized users in the locked state. The evaluator shall attempt to perform some actions not listed in the selection while the device is in the locked state and verify that those actions do not succeed.

5.4.6.4 Re-Authentication

FIA_UAU_EXT.3	Extended: Re-Authentication

FIA_UAU_EXT.3.1: The TSF shall require the user to enter the correct Password Authentication Factor when the user changes the Password Authentication Factor, and following TSF- and user-initiated locking in order to transition to the unlocked state, and [selection:[assignment: *other conditions*], *no other conditions*].

Application Note: TSF- and user-initiated locking is described in FTA_SSL_EXT.1.

Assurance Activity:

Test 1: The evaluator shall configure the TSF to use the Password Authentication Factor according to the AGD guidance. The evaluator shall change Password Authentication Factor according to the AGD guidance and verify that the TSF requires the entry of the Password Authentication Factor before allowing the factor to be changed.

Test 2: The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF_EXT.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.

Test 3: The evaluator shall configure user-initiated locking according to the AGD guidance. The evaluator shall lock the TSF and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.

5.4.7 X509 Certificates (FIA_X509)

5.4.7.1 Validation of Certificates

FIA_X509_EXT.1	Extended: Validation of certificates

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

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a certificate in the Trust Anchor Database.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using [selection: *the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, a Certificate Revocation List (CRL) as specified in RFC 5759*].
- The TSF shall validate the extendedKeyUsage field according to the following rules:

- Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.

- Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

- (Conditional) Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) 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. FIA_X509_EXT.2 requires that certificates are used for EAP-TLS; this use requires that the extendedKeyUsage rules are verified. Certificates may optionally be used for trusted updates of system software and mobile applications (FPT_TUD_EXT.2) and for integrity verification (FPT_TST_EXT.2) and, if implemented, must be validated to contain the Code Signing purpose extendedKeyUsage.

While FIA_X509_EXT.1.1 requires that the TOE perform certain checks on the certificate presented by a TLS server, there are corresponding checks that the authentication server 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. This check is required to support EAP-TLS for the WLAN trusted channel.

In the case of WLAN certificate validation during EAP-TLS, if the TOE verifies certificate revocation status using CRLs, the TOE should use any available stored, valid CRL. If the TOE verifies revocation status using OCSP, the certificate revocation status cannot be verified before establishing a WLAN trusted channel.

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

Application Note: This requirement applies to certificates that are used and processed by the TSF and restricts the certificates that may be added to the Trust Anchor Database.

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 and FIA_X509_EXT.3. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. The evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

Test 1: The evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function (e.g. application validation, trusted channel setup, or trusted software update), and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates--conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the node certificate and revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). For the test of the WLAN use case, only pre-stored CRLs are used. 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 has been 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.

Test 7: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

Test 8: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

Test 9: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

5.4.7.2 X509 Certificate Authentication

FIA_X509_EXT.2 Extended: X509 certificate authentication

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

Application Note: The ST author's selection shall match the selection of FTP_ITC_EXT.1.1. Certificates may optionally be used for trusted updates of system software

(FPT_TUD_EXT.2.3) and mobile applications (FPT_TUD_EXT.2.5) and for integrity verification (FPT_TST_EXT.2). If FPT_TUD_EXT.2.5 is included in the ST, "code signing for mobile applications" must be included in the selection.

FIA_X509_EXT.2.2 When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [selection: *allow the administrator to choose whether to accept the certificate in these cases, allow the user 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 or user-configured option is selected by the ST Author, the ST Author must also select function 30 in FMT_SMF_EXT.1.

The TOE may behave differently depending on the trusted channel; for example, in the case of WLAN where connections are unlikely to be established, the TOE may accept the certificate even though certificates are not accepted for other channels. The ST author should select all applicable behaviors.

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. The evaluator shall verify that any distinctions between trusted channels are described. 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 the following test for each trusted channel:

Test: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

5.4.7.3 Request Validation of Certificates

FIA_X509_EXT.3	Extended: Request Validation of certificates

FIA_X509_EXT.3.1 The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2 The TSF shall respond to the requesting application with the success or failure of the validation.

Application Note: In order to comply with all of the rules in FIA_X509_EXT.1, multiple API calls may be required; all of these calls should be clearly documented.

Assurance Activity:

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security function (certificate validation) described in this requirement. This documentation shall be clear as to which results indicate success and failure.

The evaluator shall write, or the developer shall provide access to, an application that requests certificate validation by the TSF. The evaluator shall verify that the results from the validation match the expected results according to the API documentation. This application may be used to verify that import, removal, modification, and validation are performed correctly according to the tests required by FDP_STG_EXT.1, FDP_ITC_EXT.1, FMT_SMF_EXT.1.1, and FIA_X509_EXT.1.

5.5 Class: Security Management (FMT)

Both the user and the administrator (as defined in the Glossary, Section 1.2) may manage the TOE. This administrator is likely to be acting remotely and could be the Mobile Device Management (MDM) Administrator acting through an MDM Agent.

The Administrator is responsible for management activities, including setting the policy that is applied by the enterprise on the Mobile Device. These management functions are likely to be a different set than those management functions provided to the user. Management functions that are provided to the user and not the administrator are listed in FMT_MOF_EXT.1.1. Management functions for which the administrator may adopt a policy that restricts the user from performing that function are listed in FMT_MOF_EXT.1.2.

Table 1 compares the management functions required by this Protection Profile in the following three requirements (FMT_MOF_EXT.1.1, FMT_MOF_EXT.1.2, and FMT_SMF_EXT.1).

5.5.1 Management of Functions in TSF (FMT_MOF)

FMT_MOF_EXT.1	Extended: Management of security functions behavior

FMT_MOF_EXT.1.1 The TSF shall restrict the ability to perform the functions in column 3 of Table 1 to the user.

Application Note: The functions that have an “M” in the third column are mandatory for this component; the functions that have an “O” in the third column are optional and may be

selected; and those functions with a "-" in the third are not applicable and may not be selected. The ST author should select those security management functions which only the user may perform.

The ST author may not select the same function in both FMT_MOF_EXT.1.1 and FMT_MOF_EXT.1.2.

The ST author should select those security management functions which the administrator may not perform. The ST author may use a table in the ST, indicating with clear demarcations (to be accompanied by an index) those functions that are not implemented with APIs for the administrator and are restricted to the user (as in column 2). The ST author should iterate a row to indicate any variations in the selectable sub-functions or assigned values with respect to the values in the columns.

For functions that are mandatory, any sub-functions not in a selection are also mandatory and any assignments must contain at least one assigned value. For non-selectable sub-functions in an optional function, all sub-functions outside a selection must be implemented in order for the function to be listed.

Assurance Activity:

The evaluator shall verify that the TSS describes those management functions which may only be performed by the user and confirm that the TSS does not include an Administrator API for any of these management functions. This activity will be performed in conjunction with FMT_SMF_EXT.1.

FMT_MOF_EXT.1.2 The TSF shall restrict the ability to perform the functions in column 5 of Table 1 to the administrator when the device is enrolled and according to the administrator-configured policy.

Application Note: As long as the device is enrolled, the administrator (of the enterprise) must be guaranteed that minimum security functions of the enterprise policy are enforced. Further restrictive policies can be applied at any time by the user on behalf of the user or other administrators.

The functions that have an "M" in the fifth column are mandatory for this component; the functions that have an "O" in the fifth column are optional and may be selected; and those functions with a "-" in the fifth are not applicable and may not be selected.

The ST author may not select the same function in both FMT_MOF_EXT.1.1 and FMT_MOF_EXT.1.2.

The ST author should select those security management functions which the administrator may restrict. The ST author may use a table in the ST, indicating with clear demarcations (to be accompanied by an index) those functions that are and are not implemented with APIs for the administrator (as in column 4). Additionally, the ST author should demarcate which functions the user is prevented from accessing or performing (as in column 5). The ST author should iterate a row to indicate any variations in the selectable sub-functions or assigned values with respect to the values in the columns.

For functions that are mandatory, any sub-functions not in a selection are also mandatory and any assignments must contain at least one assigned value. For non-selectable sub-functions in an optional function, all sub-functions outside the selection must be implemented in order for the function to be listed.

Assurance Activity:

The evaluator shall verify that the TSS describes those management functions which may be performed by the Administrator, to include how the user is prevented from accessing, performing, or relaxing the function (if applicable), and how applications/APIs are prevented from modifying the Administrator configuration. The TSS also describes any functionality that is affected by administrator-configured policy and how. This activity will be performed in conjunction with FMT_SMF_EXT.1.

Test 1: The evaluator shall use the ***test environment*** to deploy policies to Mobile Devices.

Test 2: The evaluator shall create policies which collectively include all management functions which are controlled by the (enterprise) administrator and cannot be overridden/relaxed by the user as defined in FMT_MOF_EXT.1.1. The evaluator shall apply these policies to devices, attempt to override/relax each setting both as the user (if a setting is available) and as an application (if an API is available), and ensure that the TSF does not permit it. Note that the user may still apply a more restrictive policy than that of the administrator.

Test 3: Additional testing of functions provided to the administrator are performed in conjunction with the testing activities for FMT_SMF_EXT.1.1.

5.5.2 Specification of Management Functions (FMT_SMF)

5.5.2.1 Specification of Management Functions

FMT_SMF_EXT.1	**Extended: Specification of Management Functions**

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

Management Function **Status Markers:** M – Mandatory O – Optional/Objective	**FMT_SMF_EXT.1**	**FMT_MOF_EXT.1.1**	**Administrator**	**FMT_MOF_EXT.1.2**
1. configure password policy: a. minimum password length b. minimum password complexity c. maximum password lifetime	M	-	M	M
2. configure session locking policy:	M	-	M	M

a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures				
3. enable/disable the VPN protection: a. across device [selection: *b. on a per-app basis* *c. no other method*]	M	O	O	O
4. enable/disable [assignment: *list of radios*]	M	O	O	O
5. enable/disable [assignment: *list of audio or visual collection devices*]: a. across device [selection: *b. on a per-app basis* *c. no other method*]	M	-	M	M
6. specify wireless networks (SSIDs) to which the TSF may connect	M	-	M	O
7. configure security policy for each wireless network: a. [selection: *specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s), specify the FQDN(s) of acceptable WLAN authentication server certificate(s)*] b. security type c. authentication protocol d. client credentials to be used for authentication	M	-	M	O
8. transition to the locked state	M	-	M	-
9. TSF wipe of protected data	M	-	M	-
10. configure application installation policy by [selection: *a. restricting the sources of applications,* *b. specifying a set of allowed applications based on [assignment: application characteristics] (an application whitelist),* c. *denying installation of applications*]	M	-	M	M
11. import keys/secrets into the secure key storage	M	O	O	-
12. destroy imported keys/secrets and [selection: *no other keys/secrets, [assignment: list of other categories of keys/secrets]*] in the secure key storage	M	O	O	-
13. import X.509v3 certificates into the Trust Anchor Database	M	-	M	O
14. remove 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	M	O	O	-
15. enroll the TOE in management	M	M	O	-
16. remove applications	M	-	M	O
17. update system software	M	-	M	O
18. install applications	M	-	M	O
19. remove Enterprise applications	M	-	M	-
20. configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name [selection: *c. allow/disallow additional wireless technologies to be used with Bluetooth,*	M	O	O	O

d. disable/enable Advertising (for LE), *e. disable/enable the Connectable mode* *f. disable/enable the Bluetooth services and/or profiles available on the device,* *g. specify minimum level of security for each pairing ,* *h. configure allowable methods of Out of Band pairing* i. *no other Bluetooth configuration*]				
21. enable/disable 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. *all notifications*]	M	O	O	O
22. enable/disable all data signaling over [assignment: *list of externally accessible hardware ports*]	O	O	O	O
23. enable/disable [assignment: *list of protocols where the device acts as a server*]	O	O	O	O
24. enable/disable developer modes	O	O	O	O
25. enable data-at rest protection	O	O	O	O
26. enable removable media's data-at-rest protection	O	O	O	O
27. enable/disable bypass of local user authentication	O	O	O	O
28. wipe Enterprise data	O	O	O	-
29. approve [selection: *import, removal*] by applications of X.509v3 certificates in the Trust Anchor Database	O	O	O	O
30. configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate	O	O	O	O
31. enable/disable the cellular protocols used to connect to cellular network base stations	O	O	O	O
32. read audit logs kept by the TSF	O	O	O	-
33. configure [selection: *certificate, public-key*] used to validate digital signature on applications	O	O	O	O
34. approve exceptions for shared use of keys/secrets by multiple applications	O	O	O	O
35. approve exceptions for destruction of keys/secrets by applications that did not import the key/secret	O	O	O	O
36. configure the unlock banner	O	-	O	O
37. configure the auditable items	O	-	O	O
38. retrieve TSF-software integrity verification values	O	O	O	O
39. enable/disable [selection: a. *USB mass storage mode,* b. *USB data transfer without user authentication,* c. *USB data transfer without authentication of the connecting system*]	O	O	O	O
40. enable/disable backup to [selection: *locally connected system, remote system*]	O	O	O	O
41. enable/disable [selection:	O	O	O	O

a. *Hotspot functionality authenticated by [selection: pre-shared key, passcode, no authentication]*, b. *USB tethering authenticated by [selection: pre-shared key, passcode, no authentication]*]				
42. approve exceptions for sharing data between [selection: *application processes, groups of application processes*]	O	O	O	O
43. place applications into application process groups based on [assignment: *application characteristics*]	O	O	O	O
44. enable/disable location services: a. across device [selection: *b. on a per-app basis* c. *no other method*]	M	O	O	O
45. [assignment: *list of other management functions to be provided by the TSF*]	O	O	O	O

Table 1: Management Functions

Application Notes:

Table 1 compares the management functions required by this Protection Profile.

The first column lists the management functions identified in the PP.

In the following columns:

- 'M' means Mandatory,
- 'O' means Optional/Objective,

The second column (FMT_SMF_EXT.1) indicates whether the function is to be implemented. The ST author should select which Optional functions are implemented.

The third column (FMT_MOF_EXT.1.1) indicates functions that are to be restricted to the user.

The fourth column (Administrator) indicates functions that are to be always available to the administrator. These may be derived from the second and third columns Thus, the TOE must offer these functions, if included in FMT_SMF_EXT.1, to the administrator to perform.

The fifth column (FMT_MOF_EXT.1.2) indicates whether the function is to be restricted to the administrator when the device is enrolled and the administrator applies the indicated policy.

The ST author may use a table in the ST, listing only those functions that are implemented. For functions that are mandatory, any sub-functions not in a selection are also mandatory and any assignments must contain at least one assigned value. For functions that are optional and contain an assignment or selection, at least one value must be assigned/selected to be included in the ST. For non-selectable sub-functions in an optional function, all sub-functions must be implemented in order for the function to be included. For functions with a "per-app basis" sub function and an assignment, the ST author must indicate which assigned features are manageable on a per-app basis and which are not by iterating the row.

Function-specific Application Notes:

For functions 3, 5 and 44, the function must be implemented on a device-wide basis but may also be implemented on a per-app basis in which the configuration includes the list of applications to which the enable/disable applies.

Function 3 addresses enabling and disabling the IPsec VPN only. The configuration of the VPN Client itself (with information such as VPN Gateway, certificates, and algorithms) is addressed by the Protection Profile for IPsec VPN Clients. The administrator options should only be listed if the administrator can remotely enable/disable the VPN connection.

The assignment in function 4 consists of all radios, such as Wi-Fi, GPS, cellular, NFC, Bluetooth BR/EDR, and Bluetooth LE, which can be enabled and disabled. In the future, if both Bluetooth BR/EDR and Bluetooth LE are supported, they will be required to be enabled and disabled separately. Disablement of the cellular radio does not imply that the radio may not be enabled in order to place emergency phone calls; however, it is not expected that a device in "airplane mode", where all radios are disabled, will automatically (without authorization) turn on the cellular radio to place emergency calls.

The assignment in function 5 consists of all audio and visual devices, such as camera and microphone, which can be enabled and disabled by either the user or administrator. Disablement of the microphone does not imply that the microphone may not be enabled in order to place emergency phone calls.

Regarding functions 4 and 5, disablement of a particular radio or audio/visual device must be effective as soon as the TOE has power. Disablement must also apply when the TOE is booted into auxiliary boot modes, for example, associated with updates or backup. If the TOE supports states in which security management policy is inaccessible, for example, due to data-at-rest protection, it is acceptable to meet this requirement by ensuring that these devices are disabled by default while in these states. That these devices are disabled during auxiliary boot modes does not imply that the device (particularly the cellular radio) may not be enabled in order to perform emergency phone calls.

The security policy in function 7 addresses security types, such as WPA2-Enterprise, and authentication protocols, such as EAP-TLS. The CA or FQDN is specified for comparison according to FCS_TLSC_EXT.1.2.

Wipe of the TSF (function 9) is performed according to FCS_CKM_EXT.5.

The selection in function 10 allows the ST author to select which mechanisms are available to the administrator through the MDM Agent to restrict the applications which the user may install.

- If the administrator can restrict the sources from which applications can be installed, the ST author selects option a.
- If the administrator can specify a whitelist of allowed applications, the ST author selects option b. The ST author should list any application characteristics (e.g. name, version, or developer) based on which the whitelist can be formed.
- If the administrator can prevent the user from installing additional applications, the ST author selects c.

In the future, function 14 may require destruction or disabling 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.

For function 15, the enrollment function may be installing an MDM agent and includes the policies to be applied to the device. It is acceptable for the user approval notice to require the user to intentionally opt to view the policies (for example, by "tapping" on a "View" icon) rather than listing the policies in full in the notice.

For function 17, the administrator capability to update the system software may be limited to causing a prompt to the user to update rather than the ability to initiate the update itself. As the administrator is likely to be acting remotely, he/she would be unaware of inopportune situations, such as low power, which may cause the update to fail and the device to become inoperable. The user can refuse to accept the update in such situations. It is expected that system architects will be cognizant of this limitation and will enforce network access controls in order to enforce enterprise-critical updates.

Function 18 addresses both installation and update. This protection profile does not distinguish between installation and update of applications because mobile devices typically completely overwrite the previous installation with a new installation during an application update.

For function 19, "Enterprise applications" are those applications that are installed by the enterprise administrator.

For function 20, management of the Discoverable mode and of the Bluetooth device name are both mandatory. All other management functions for Bluetooth are currently objective:

- Function 20.c includes disabling WiFi being used as a part of Bluetooth High Speed and disabling NFC as an Out Of Band pairing method for Bluetooth.
- The Bluetooth services and/or profiles that may be disabled (function 20.f) should be listed for the user/administrator either by service and/or profile name or by the types of applications for which the service and/or profile is used.
- Examples of levels of security for function 20.g are the use of legacy pairing and the use of different types of Secure Simple Pairing (for BR/EDR: Just Works, Numeric Comparison, Passkey Entry, Out of Band; for LE: Just Works, Passkey Entry, Out of Band). The level of security may be configurable for each individual pairing or for all Bluetooth pairings.
  For products entering into evaluation after Quarter 3, 2015, if the TSF supports any security modes other than Security Mode 4/Level 3 or higher (for BR/EDR) or Security Mode 1/Level 3 (for LE), the TSF shall provide a mechanism for the user to choose the minimum level of security to enforce for a particular device during the pairing process.
  For BR/EDR, Security Mode 4/Level 3 corresponds to Secure Simple Pairing (SSP) with a requirement for encryption and a requirement for man-in-the-middle (MiTM) protection during pairing. For LE, Security Mode 1/Level 3 corresponds to a requirement to use authenticated pairing with encryption. This requirement could be met through the use of a menu that provides the user with a way to disable legacy

and/or Just Works SSP (for BR/EDR) and to require authenticated pairing and encryption (for LE).

- If Out of Band pairing methods are supported, function 20.h should be selected.

If the display of notifications in the locked state is supported, the configuration of these notifications (function 21) must be included in the selection.

The assignment in function 22 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 either the user or administrator. Disablement of data transfer over an external port must be effective during and after boot into the normal operative mode of the device. If the TOE supports states in which configured security management policy is inaccessible, for example, due to data-at-rest protection, it is acceptable to meet this requirement by ensuring that data transfer is disabled by default while in these states. Each of the ports may be enabled or disabled separately. The configuration policy need not disable all ports together.

The assignment in function 23 consists of all protocols where the TSF acts as a server, such as WiFi tethering (personal hotspot), which can be enabled and disabled by either the user or administrator.

Function 24 must be included in the selection if developer modes are supported by the TSF.

Function 25 must be included in the selection if data-at-rest protection is not natively enabled.

Function 26 should be included in the selection if the device supports removable media.

Function 27 must be included in the selection if bypass of local user authentication, such as a "Forgot Password" feature involving password hints or remote authentication, is supported.

Function 29 must be included in the selection if the TSF allows applications to import or remove X.509v3 certificates from the Trust Anchor Database. These applications do not include MDM Agents. This function does not apply to applications trusting a certificate for its own validations. The function only applies to situations where the application modifies the device-wide Trust Anchor Database, affecting the validations performed by the TSF for other applications. The user or administrator may be provided the ability to globally allow or deny any application requests in order to meet this requirement.

Function 30 must be included in the selection if the "administrator-configured option" is selection in FIA_X509_EXT.2.2.

Function 33 should be included in the selection if FPT_TUD_EXT.2.5 is included in the ST and the configurable options is selected.

Function 34 should be included in the selection if user or administrator is selected in FCS_STG_EXT.1.4.

Function 35 should be included in the selection if user or administrator is selected in FCS_STG_EXT.1.5.

Function 36 must be included in the selection if FTA_TAB.1 is included in the ST.

Function 37 must be included in the selection if FAU_SEL.1 is included in the ST.

For function 41, hotspot functionality refers to the condition in which the mobile device is serving as an access point to other devices, not the connection of the TOE to external hotspots.

Functions 42 and 43 correspond to FDP_ACF_EXT.1.2.

For function 44, location services includes location information gathered from GPS, cellular, and WiFi. Disablement of location services across the device does not imply that the microphone may not be enabled in order to place emergency phone calls.

Assurance Activity:

The evaluator shall verify that the TSS describes all management functions, what role(s) can perform each function, and how these functions are (or can be) restricted to the roles identified by FMT_MOF_EXT.1.

The following activities are organized according to the function number in the table. These activities include TSS assurance activities, AGD assurance activities, and test activities.

Test activities specified below shall take place in the *test environment* described in the Assurance Activity for FPT_TUD_EXT.1.1, FPT_TUD_EXT.1.2, and FPT_TUD_EXT.1.3. The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 1

The evaluator shall verify the TSS defines the allowable policy options: the range of values for both password length and lifetime, and a description of complexity to include character set and complexity policies (e.g., configuration and enforcement of number of uppercase, lowercase, and special characters per password).

Test 1: The evaluator shall exercise the TSF configuration as the administrator and perform positive and negative tests, with at least two values set for each variable setting, for each of the following:

- minimum password length
- minimum password complexity
- maximum password lifetime

Function 2

The evaluator shall verify the TSS defines the range of values for both timeout period and number of authentication failures.

Test 2: The evaluator shall exercise the TSF configuration as the user and the administrator. The evaluator shall perform positive and negative tests, with at least two values set for each variable setting, for each of the following.

- screen-lock enabled/disabled
- screen lock timeout
- number of authentication failures (may be combined with test for FIA_AFL.1)

Function 3

Test 3: The evaluator shall perform the following tests:

Test 3a: The evaluator shall exercise the TSF configuration to enable the VPN protection. These configuration actions must be used for the testing of the FDP_IFC.1.1 requirement.

Test 3b: [conditional] If "per-app basis" is selected, the evaluator shall create two applications and enable one to use the VPN and the other to not use the VPN. The evaluator shall exercise each application (attempting to access network resources; for example by browsing different websites) individually while capturing packets from the TOE. The evaluator shall verify from the packet capture that the traffic from the VPN-enabled application is encapsulated in IPsec and that the traffic from the VPN-disabled application is not encapsulated in IPsec.

Function 4

The evaluator shall verify that the TSS includes a description of each radio and an indication of if the radio can be enabled/disabled along with what role can do so. In addition the evaluator shall verify that the frequency ranges at which each radio operates is included in the TSS. The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function. The spectrum analyzer and Ramsey box used in this test shall be NVLAP approved and calibrated.

Test 4: The evaluator shall exercise the TSF configuration as both the user and administrator to enable and disable the state of each radio (e.g. Wi-Fi, GPS, cellular, NFC, Bluetooth) listed by the ST author. Additionally, the evaluator shall repeat the steps below, booting into any auxiliary boot mode supported by the device. For each radio, the evaluator shall:

Step 1 - Configure spectrum analyzer to sweep desired frequency range for the radio to be tested (based on range provided in the TSS) and place the handset into a Ramsey Box (or other RF-shielding environment) to isolate them from all other RF traffic.

Step 2 - The evaluator shall create a baseline of the expected behaviour of RF signals. If a spike of RF activity for the uplink channel for the specific radio frequency band is observed it is deemed that the radio are enabled. The evaluator shall power on the device, ensure the radio in question is enabled, power off the device, enable "Max Hold" on the spectrum analyzer and power on the device. The evaluator shall observe if any RF spikes are present. The evaluator shall enter any necessary passwords to complete the boot process, waiting 2 minutes and resetting the spectrum analyzer between each step.

Step 3 - The evaluator shall disable the radio in question and complete the above tests, five times per radio. The evaluator shall verify the absence of RF activity for the uplink channel during device reboot and casual usage.

Function 5

The evaluator shall verify that the TSS includes a description of each collection device and an indication of if it can be enabled/disabled along with what role can do so. The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function.

Test 5: The evaluator shall perform the following test(s):

Test 5a: The evaluator shall exercise the TSF configuration as both the user and administrator to enable and disable the state of each audio or visual collection devices (e.g. camera, microphone) listed by the ST author. For each collection device, the evaluator shall disable the device and then attempt to use its functionality. The evaluator shall reboot the TOE and verify that disabled collection devices may not be used during or early in the boot process. Additionally, the evaluator shall boot the device into each available auxiliary boot mode and verify that the collection device cannot be used.

Test 5b: [conditional] If "per-app basis" is selected, the evaluator shall create two applications and enable one to use access the A/V device and the other to not access the A/V device. The evaluator shall exercise each application attempting to access the A/V device individually. The evaluator shall verify that the enabled application is able to access the A/V device and the disabled application is not able to access the A/V device.

Function 6

The evaluator shall create a test environment consisting of a wireless access system and an authentication server for the purpose of tests associated with functions 6 and 7.

Test 6: The evaluator shall specify the wireless network and wireless network settings according to the AGD guidance both as an administrator and as a user. The evaluator shall specify a value for each management function according to the configuration of the test network. Minimally, the evaluator shall construct 2 SSIDs, one corresponding to a WPA2 Enterprise network using EAP-TLS and one corresponding to a disallowed SSID. The evaluator shall verify that the TSF can establish a connection to the allowed SSID, but not to the disallowed SSID.

Function 7

The evaluator shall create a test environment consisting of a wireless access system and an authentication server for the purpose of tests associated with functions 6 and 7.

The evaluator shall verify the TSS describes the configuration and enforcement of the various credential options used in validation of the WLAN authentication server. The evaluator shall review the administrative guidance to determine that it describes how to configure the security type, protocol, and client credentials for each of the credential options described in the TSS.

Test 7: The evaluator shall specify a wireless network with an incorrect value for WLAN authentication server and verify that the Mobile Device cannot connect to the WLAN. The evaluator shall repeat this test, setting incorrect values for the security type and authentication protocol individually and verify that the Mobile Device cannot connect to the WLAN. The evaluator shall then specify, for each credential option claimed in the ST, correct options and demonstrate that the TOE can successfully establish a connection to the WLAN.

Function 8

Test 8: The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to transition to a locked state, and verify that the device transitions to the locked state upon command.

Function 9

Test 9: The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to perform a wipe of protected data. The evaluator must ensure that this management setup is used when conducting the assurance activities in FCS_CKM_EXT.5.

Function 10

The evaluator shall verify the TSS describes the allowable application installation policy options based on the selection included in the ST. If the application whitelist is selected, the evaluator shall verify that the TSS includes a description of each application characteristic upon which the whitelist may be based.

Test 10: The evaluator shall exercise the TSF configuration as the administrator to restrict particular applications, sources of applications, or application installation according to the AGD guidance. The evaluator shall attempt to install unauthorized applications and ensure that this is not possible. The evaluator shall, in conjunction, perform the following specific tests:

Test 10a: [conditional] The evaluator shall attempt to connect to an unauthorized repository in order to install applications.

Test 10b: [conditional] The evaluator shall attempt to install two applications (one whitelisted, and one not) from a known good repository and verify that the application not on the whitelist is rejected. The evaluator shall also attempt to side-load executables or installation packages via USB connections to determine that the white list is still adhered to

Function 11 & Function 12

The evaluator shall verify that the TSS describes each category of keys/secrets that can be imported into the TSF's secure key storage.

Test 11:
&
Test 12: The test of these functions is performed in association with FCS_STG_EXT.1.

Function 13

The evaluator shall review the AGD guidance to determine that it describes the steps needed to import, modify, or remove certificates in the Trust Anchor database, and that the users that have authority to import those certificates (e.g., only administrator, or both administrators and users) are identified.

Test 13: The evaluator shall import certificates according to the AGD guidance as the user and/or as the administrator, as determined by the administrative guidance. The evaluator shall verify that no errors occur during import. The evaluator should perform an action requiring use of the X.509v3 certificate to provide assurance that installation was completed properly.

Function 14

The evaluator shall verify that the TSS describes each additional category of X.509 certificates and their use within the TSF.

Test 14: The evaluator shall remove an administrator-imported certificate and any other categories of certificates included in the assignment of function 14 from the Trust Anchor Database according to the AGD guidance as the user and as the administrator.

Function 15

The evaluator shall examine the TSS to ensure that it contains a description of each management function that will be enforced by the enterprise once the device is enrolled. The evaluator shall examine the AGD guidance to determine that this same information is present.

Test 15: The evaluator shall verify that user approval is required to enroll the device into management.

Function 16

The evaluator shall verify that the TSS includes an indication of what applications (e.g., user-installed applications, Administrator-installed applications, or Enterprise applications) can be removed along with what role can do so. The evaluator shall examine the AGD guidance to determine that it details, for each type of application that can be removed, the procedures necessary to remove those applications and their associated data. For the purposes of this assurance activity, "associated data" refers to data that are created by the app during its operation that do not exist independent of the app's existence, for instance, configuration data, or e-mail information that's part of an e-mail client. It does not, on the other hand, refer to data such as word processing documents (for a word processing app) or photos (for a photo or camera app).

Test 16: The evaluator shall attempt to remove applications according to the AGD guidance and verify that the TOE no longer permits users to access those applications or their associated data.

Function 17

Test 17: The evaluator shall attempt to update the TSF system software following the procedures in the AGD guidance and verify that updates correctly install and that the version numbers of the system software increase.

Function 18

Test 18: The evaluator shall attempt to install a mobile application following the procedures in the AGD guidance and verify that the mobile application is installed and available on the TOE.

Function 19

The evaluator shall verify that the TSS includes an indication of what Enterprise applications are removable, what actions initiate this removal, and what role can do so. This activity can be performed in conjunction with the TSS activity defined for Function 16. The evaluator shall review the AGD guidance to determine that it describes the steps needed to remove Enterprise applications from the device.

Test 19: The evaluator shall attempt to remove any Enterprise applications from the device by following the administrator guidance. The evaluator shall verify that the TOE no longer permits users to access those applications or their associated data.

Function 20

The evaluator shall ensure that the TSS includes a description of the Bluetooth profiles and services supported and the Bluetooth security modes and levels supported by the TOE. If function c is selected, the evaluator shall verify that the TSS describes any additional wireless technologies that may be used with Bluetooth, including WiFi with Bluetooth High Speed and NFC as an Out of Band pairing mechanism. If function f is selected, the evaluator shall verify that all supported Bluetooth services are listed in the TSS as manageable and, if the TOE allows disabling by application rather than by service name, that a list of services for each application is also listed. If function g is selected, the evaluator shall verify that the TSS describes the method by which the level of security for pairings are managed, including whether the setting is performed for each pairing or is a global setting. If function h is selected, the evaluator shall verify that the TSS describes when Out of Band pairing methods are allowed and which ones are configurable.

Test 20: The evaluator shall use a Bluetooth-specific protocol analyzer to perform the following tests of each sub-function:

Test 20a: The evaluator shall disable the Discoverable mode and shall verify that other Bluetooth BR/EDR devices cannot detect the TOE. The evaluator shall use the protocol analyzer to verify that the TOE does not respond to inquiries from other devices searching for Bluetooth devices. The evaluator shall enable Discoverable mode and verify that other devices can detect the TOE and that the TOE sends response packets to inquiries from searching devices.

Test 20b: The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name, change the Bluetooth device name, and verify that the Bluetooth traffic from the device lists the new name.

Test 20c: [conditional] The evaluator shall disable additional wireless technologies for the TOE and verify that the Bluetooth traffic is not able to be sent over WiFi using Bluetooth High Speed, and that NFC cannot be used for pairing. The evaluator shall enable additional wireless technologies and verify that Bluetooth High Speed uses WiFi or that the device can pair using NFC.

Test 20d: [conditional] The evaluator shall enable Advertising for Bluetooth LE, verify that the advertisements are captured by the protocol analyzer, disable Advertising, and verify that no advertisements from the device are captured by the protocol analyzer.

Test 20e: [conditional] The evaluator shall enable Connectable mode and verify that other Bluetooth devices may pair with the TOE and (if the devices were bonded) re-connect after pairing and disconnection. For BR/EDR devices: The evaluator shall use the protocol analyzer to verify that the TOE responds to pages from the other devices and permits pairing and re-connection. The evaluator shall disable Connectable mode and verify that the TOE does not respond to pages from remote Bluetooth devices, thereby not permitting pairing or re-connection. For LE: The evaluator shall use the protocol analyzer to verify that the TOE sends connectable advertising events and responds to connection requests. The evaluator shall disable Connectable mode and verify that the TOE stops sending connectable advertising events and stops responding to connection requests from remote Bluetooth devices.

Test 20f: [conditional] The evaluator shall allow low security modes/levels on the TOE and shall initiate pairing with the TOE from a remote device that allows *only* something other than Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR), or Security Mode 1/Level 3 (for LE). (For example, a remote BR/EDR device may claim Input/Output capability “NoInputNoOutput” and state that man-in-the-middle (MiTM) protection is not

required. A remote LE device may not support encryption.) The evaluator shall verify that this pairing attempt succeeds due to the TOE falling back to the low security mode/level. The evaluator shall then remove the pairing of the two devices, prohibit the use of low security modes/levels on the TOE, then attempt the connection again. The evaluator shall verify that the pairing attempt fails. With the low security modes/levels disabled, the evaluator shall initiate pairing from the TOE to a remote device that supports Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR) or Security Mode 1/Level 3 (for LE). The evaluator shall verify that this pairing is successful and uses the high security mode/level.

Test 20g: [conditional] The evaluator shall attempt to pair using each of the Out of Band pairing methods, verify that the pairing method works, iteratively disable each pairing method, and verify that the pairing method fails.

Function 21

The evaluator shall examine the AGD Guidance to determine that it specifies, for at least each category of information selected for Function 21, how to enable and disable display information for that type of information in the locked state.

Test 21: For each category of information listed in the AGD guidance, the evaluator shall verify that when that TSF is configured to limit the information according to the AGD, the information is no longer displayed in the locked state.

It should be noted that the following functions are optional capabilities, if the function is implemented, then the following assurance activities shall be performed. The notation of "[conditional] beside the function number indicates that if the function is not included in the ST, then there is no expectation that the assurance activity be performed.

Function 22 [conditional]

The evaluator shall verify that the TSS includes a list of each externally accessible hardware port and an indication of if data transfer over that port can be enabled/disabled. AGD guidance will describe how to perform the enable/disable function.

Test 22: The evaluator shall exercise the TSF configuration to enable and disable data transfer capabilities over each externally accessible hardware ports (e.g. USB, SD card, HDMI) listed by the ST author. The evaluator shall use test equipment for the particular interface to ensure that no low-level signalling is occurring on all pins used for data transfer when they are disabled. For each disabled data transfer capability, the evaluator shall repeat this test by rebooting the device into the normal operational mode and verifying that the capability is disabled throughout the boot and early execution stage of the device.

Function 23 [conditional]

The evaluator shall verify that the TSS describes how the TSF acts as a server in each of the protocols listed in the ST, and the reason for acting as a server.

Test 23: The evaluator shall attempt to disable each listed protocol in the assignment, which should include tethering uses. The evaluator shall verify that remote devices can no longer access the TOE or TOE resources using any disabled protocols.

Function 24 [conditional]

Test 24: The evaluator shall exercise the TSF configuration as both the user and administrator to enable and disable any developer mode. The evaluator shall test that developer mode

access is not available when its configuration is disabled. The evaluator shall verify the developer mode remains disabled during device reboot.

Function 25 [conditional]

Test 25: The evaluator shall exercise the TSF configuration as both the user and administrator to enable system-wide data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all assurance activities for DAR (see Section 0) are conducted with the device in this configuration.

Function 26 [conditional]

Test 26: The evaluator shall exercise the TSF configuration as both the user and administrator to enable removable media's data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all assurance activities for DAR (see Section 0) are conducted with the device in this configuration.

Function 27 [conditional]

The evaluator shall examine the AGD guidance to determine that it describes how to enable and disable any "Forgot Password", password hint, or remote authentication (to bypass local authentication mechanisms) capability.

Test 27: For each mechanism listed in the AGD guidance that provides a "Forgot Password" feature or other means where the local authentication process can be bypassed, the evaluator shall disable the feature and ensure that they are not able to bypass the local authentication process.

Function 28 [conditional]

Test 28: The evaluator shall attempt to wipe Enterprise data resident on the device according to the administrator guidance. The evaluator shall verify that the data is no longer accessible by the user.

Function 29 [conditional]

The evaluator shall verify that the TSS describes how approval for an application to perform the selected action (import, removal) with respect to certificates in the Trust Anchor Database is accomplished (e.g., a pop-up, policy setting, etc.).

The evaluator shall also verify that the API documentation provided according to Section 6.2.1 includes any security functions (import, modification, or destruction of the Trust Anchor Database) allowed by applications.

Test 29: The evaluator shall perform one of the following tests:

Test 29a: [Conditional] If applications may import certificates to the Trust Anchor Database, the evaluator shall write, or the developer shall provide access to, an application that imports a certificate into the Trust Anchor Database. The evaluator shall verify that the TOE requires approval before allowing the application to import the certificate:

- The evaluator shall deny the approvals to verify that the application is not able to import the certificate. Failure of import shall be tested by attempting to validate a certificate that chains to the certificate whose import was attempted (as described in the Assurance Activity for FIA_X509_EXT.1).
- The evaluator shall repeat the test, allowing the approval to verify that the application is able to import the certificate and that validation occurs.

Test 29b: [Conditional] If applications may remove certificates in the Trust Anchor Database, the evaluator shall write, or the developer shall provide access to, an application that removes certificates from the Trust Anchor Database. The evaluator shall verify that the TOE requires approval before allowing the application to remove the certificate:

- The evaluator shall deny the approvals to verify that the application is not able to remove the certificate. Failure of removal shall be tested by attempting to validate a certificate that chains to the certificate whose removal was attempted (as described in the Assurance Activity for FIA_X509_EXT.1).

The evaluator shall repeat the test, allowing the approval to verify that the application is able to remove/modify the certificate and that validation no longer occurs.

Function 30 [conditional]

Test 30: The test of this function is performed in conjunction with FIA_X509_EXT.2.2.

Function 31 [conditional]

The evaluator shall ensure that the TSS describes which cellular protocols can be disabled. The evaluator shall confirm that the AGD guidance describes the procedure for disabling each cellular protocol identified in the TSS.

Test 31: The evaluator shall attempt to disable each cellular protocol according to the administrator guidance. The evaluator shall attempt to connect the device to a cellular network and, using network analysis tools, verify that the device does not allow negotiation of the disabled protocols.

Function 32 [conditional]

Test 32: The evaluator shall attempt to read any device audit logs according to the administrator guidance and verify that the logs may be read. This test may be performed in conjunction with the assurance activity of FAU_GEN.1.

Function [conditional]

Test 33: The test of this function is performed in conjunction with FPT_TUD_EXT.2.5.

Function 34 [conditional]

The evaluator shall verify that the TSS describes how the approval for exceptions for shared use of keys/secrets by multiple applications is accomplished (e.g., a pop-up, policy setting, etc.).

Test 34: The test of this function is performed in conjunction with FCS_STG_EXT.1.

Function 35 [conditional]

The evaluator shall verify that the TSS describes how the approval for exceptions for destruction of keys/secrets by applications that did not import the key/secret is accomplished (e.g., a pop-up, policy setting, etc.).

Test 35: The test of this function is performed in conjunction with FCS_STG_EXT.1.

Function 36 [conditional]

The evaluator shall verify that the TSS describes any restrictions in banner settings (e.g., character limitations).

Test 36: The test of this function is performed in conjunction with FTA_TAB.1.

Function 37 [conditional]

Test 37: The test of this function is performed in conjunction with FAU_SEL.1.

Function 38 [conditional]

Test 38: The test of this function is performed in conjunction with FPT_NOT_EXT.1.2.

Function 39 [conditional]

The evaluator shall verify that the TSS includes a description of how data transfers can be managed over USB.

Test 39: The evaluator shall perform the following tests based on the selections in 0.

Test 39a: [conditional] The evaluator shall disable USB mass storage mode, attach the device to a computer, and verify that the computer cannot mount the TOE as a drive. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.

Test 39b: [conditional] The evaluator shall disable USB data transfer without user authentication, attach the device to a computer, and verify that the TOE requires user authentication before the computer can access TOE data. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.

Test 39c: [conditional] The evaluator shall disable USB data transfer without connecting system authentication, attach the device to a computer, and verify that the TOE requires connecting system authentication before the computer can access TOE data. The evaluator shall then connect the TOE to another computer and verify that the computer cannot access TOE data. The evaluator shall then connect the TOE to the original computer and verify that the computer can access TOE data.

Function 40 [conditional]

The evaluator shall verify that the TSS includes a description of available backup methods that can be enabled/disabled.

Test 40: The evaluator shall disable each supported backup location in turn and verify that the TOE cannot complete a backup. The evaluator shall then enable each supported backup location in turn and verify that the TOE can perform a backup.

Function 41 [conditional]

The evaluator shall verify that the TSS includes a description of Hotspot functionality and USB tethering to include any authentication for these.

Test 41: The evaluator shall perform the following tests based on the selections in 0.

Test 41a: [conditional] The evaluator shall enable hotspot functionality with each of the of the support authentication methods. The evaluator shall connect to the hotspot with another device and verify that the hotspot functionality requires the configured authentication method.

Test 41b: [conditional] The evaluator shall enable USB tethering functionality with each of the of the support authentication methods. The evaluator shall connect to the TOE over USB with another device and verify that the tethering functionality requires the configured authentication method.

Function 42 [conditional]

Test 42: The test of this function is performed in conjunction with FDP_ACF_EXT.1.2.

Function 43 [conditional]

Test 43: The test of this function is performed in conjunction with FDP_ACF_EXT.1.2.

Function 44 [conditional]

Test 44: The evaluator shall perform the following tests.

Test 44a: The evaluator shall enable location services device-wide and shall verify that an application (such as a mapping application) is unable to access the TOE's location information.

Test 44b: [conditional] If "per-app basis" is selected, the evaluator shall create two applications and enable one to use access the location services and the other to not access the location services. The evaluator shall exercise each application attempting to access location services individually. The evaluator shall verify that the enabled application is able to access the location services and the disabled application is not able to access the location services.

Function 45 [conditional]

The evaluator shall verify that the TSS describes all assigned security management functions and their intended behavior.

Test 45: The evaluator shall design and perform tests to demonstrate that the function may be configured and that the intended behavior of the function is enacted by the TOE.

5.5.2.2 Specification of Remediation Actions

FMT_SMF_EXT.2	Extended: Specification of Remediation Actions

FMT_SMF_EXT.2.1 The TSF shall offer [selection: *wipe of protected data, wipe of sensitive data, alert the administrator, remove Enterprise applications,* [assignment: *list other available remediation actions*]] upon unenrollment and [selection: [assignment: *other administrator-configured triggers*], *no other triggers*].

Application Note: Unenrollment may consist of removing the MDM agent or removing the administrator's policies. The functions in the selection are remediation actions that TOE provides (perhaps via APIs) to the administrator (perhaps via an MDM agent) that are performed upon unenrollment.

Assurance Activity:

The evaluator shall verify that the TSS describes all available remediation actions, when they are available for use, and any other administrator-configured triggers.

The evaluator shall use the test environment to iteratively configure the device to perform each remediation action in the selection upon unenrollment. The evaluator shall unenroll the device according to AGD guidance and verify that the remediation action configured is performed.

5.6 Class: Protection of the TSF (FPT)

5.6.1 Anti-Exploitation Services (FPT_AEX)

5.6.1.1 Address-Space Layout Randomization

FPT_AEX_EXT.1	Extended: Anti-Exploitation Services (ASLR)

FPT_AEX_EXT.1.1 The TSF shall provide address space layout randomization (ASLR) to applications.

FPT_AEX_EXT.1.2 The base address of any user-space memory mapping will consist of at least 8 unpredictable bits.

Application Notes: The 8 unpredictable bits may be provided by the TSF RBG (as specified in FCS_RBG_EXT.1) but is not required.

Assurance Activity:

The evaluator shall ensure that the TSS section of the ST describes how the 8 bits are generated and provides a justification as to why those bits are unpredictable.

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

Test 1: The evaluator shall select 3 apps included with the TSF. These must include any web browser or mail client included with the TSF. For each of these apps, the evaluator will launch the same app on two separate Mobile Devices of the same type and compare all memory mapping locations. The evaluator must ensure that no memory mappings are placed in the same location on both devices.

If the rare (at most 1/256) chance occurs that two mappings are the same for a single app and not the same for the other two apps, the evaluator shall repeat the test with that app to verify that in the second test the mappings are different.

5.6.1.2 Memory Page Permissions

FPT_AEX_EXT.2	Extended: Anti-Exploitation Services (Memory Page Permissions)

FPT_AEX_EXT.2.1 The TSF shall be able to enforce read, write, and execute permissions on every page of physical memory.

Assurance Activity:

The evaluator shall ensure that the TSS describes of the memory management unit (MMU), and ensures that this description documents the ability of the MMU to enforce read, write, and execute permissions on all pages of virtual memory.

5.6.1.3 Overflow Protection

FPT_AEX_EXT.3	Extended: Anti-Exploitation Services (Overflow Protection)

FPT_AEX_EXT.3.1 TSF processes that execute in a non-privileged execution domain on the application processor shall implement stack-based buffer overflow protection.

Application Note:

A “non-privileged execution domain” refers to the user mode (as opposed to kernel mode, for instance) of the processor. While not all TSF processes must implement such protection, it is expected that most of the processes (to include libraries used by TSF processes) do implement buffer overflow protections.

Assurance Activity:

The evaluator shall determine that the TSS contains a description of stack-based buffer overflow protections implemented in the TSF software which runs in the non-privileged execution mode of the application processor. The exact implementation of stack-based buffer overflow protection will vary by platform. Example implementations may be activated through compiler options such as "-fstack-protector-all", “-fstack-protector”, and “/GS” flags.

The evaluator shall ensure that the TSS contains an inventory of TSF binaries and libraries, indicating those that implement stack-based buffer overflow protections as well as those that do not. The TSS must provide a rationale for those binaries and libraries that are not protected in this manner.

5.6.1.4 Domain Isolation

FPT_AEX_EXT.4	Extended: Domain Isolation

FPT_AEX_EXT.4.1 The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2 The TSF shall enforce isolation of address space between applications.

Application Note: In addition to the TSF software (e.g., kernel image, device drivers, trusted applications) that resides in storage, the execution context (e.g., address space, processor registers, per-process environment variables) of the software operating in a privileged mode of the processor (e.g., kernel), as well as the context of the trusted applications is to be protected. In addition to the software, any configuration information that controls or influences the behavior of the TSF is also to be protected from modification by untrusted subjects.

Configuration information includes, but is not limited to, user and administrative management function settings, WLAN profiles, and Bluetooth data such as the service-level security requirements database.

Untrusted subjects include untrusted applications; unauthorized users who have access to the device while powered off, in a screen-locked state, or when booted into auxiliary boot modes; and, unauthorized users or untrusted software or hardware which may have access to the device over a wired interface, either when the device is in a screen-locked state or booted into auxiliary boot modes.

Assurance Activity:

The evaluator shall ensure that the TSS describes the mechanisms that are in place that prevents non-TSF software from modifying the TSF software or TSF data that governs the behavior of the TSF. These mechanisms could range from hardware-based means (e.g. "execution rings" and memory management functionality); to software-based means (e.g. boundary checking of inputs to APIs). The evaluator determines that the described mechanisms appear reasonable to protect the TSF from modification.

The evaluator shall ensure the TSS describes how the TSF ensures that the address spaces of applications are kept separate from one another.

The evaluator shall ensure the TSS details the USSD and MMI codes available from the dialer at the locked state or during auxiliary boot modes that may alter the behavior of the TSF. The evaluator shall ensure that this description includes the code, the action performed by the TSF, and a justification that the actions performed do not modify user or TSF data. If no USSD or MMI codes are available, the evaluator shall ensure that the TSS provides a description of the method by which actions prescribed by these codes are prevented.

The evaluator shall ensure the TSS documents any TSF data (including software, execution context, configuration information, and audit logs) which may be accessed and modified over a wired interface in auxiliary boot modes. The evaluator shall ensure that the description includes data which is modified in support of update or restore of the device. The evaluator shall ensure that this documentation includes the auxiliary boot modes in which the data may be modified, the methods for entering the auxiliary boot modes, the location of the data, the manner in which data may be modified, the data format and packaging necessary to support modification, and software and/or hardware tools, if any, which are necessary for modifying the data.

The evaluator shall ensure that the TSS provides a description of the means by which unauthorized and undetected modification (that is, excluding cryptographically verified updates per FPT_TUD_EXT.2) of the TSF data over the wired interface in auxiliary boots modes is prevented. (The lack of publically available tools is not sufficient justification. Examples of sufficient justification include auditing of changes, cryptographic verification in the form of a digital signature or hash, disabling the auxiliary boot modes, and access control mechanisms that prevent writing to files or flashing partitions.)

Assurance Activity Note: The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products. In addition, the vendor provides a list of files (e.g., system files, libraries, configuration files, audit logs) that make up the TSF data. This list could be organized by folders/directories (e.g., /usr/sbin, /etc), as well as individual files that may exist outside of the identified directories.

Test 1: The evaluator shall check the "permission settings" for each file in vendor provided list of files that make up the TSF and ensure the settings are appropriate for preventing writing by untrusted applications. The evaluator shall attempt to modify a file of their choosing to ensure the mechanism enforces the permission settings and prevents modification.

Test 2: The evaluator shall create and load an app onto the Mobile Device. This app shall attempt to traverse over all file systems and report any locations to which data can be written or overwritten. The evaluator must ensure that none of these locations are part of the OS software, device drivers, system and security configuration files, key material, or another application's image/data.

Test 3: For each available auxiliary boot mode, the evaluator shall attempt to modify a TSF file of their choosing using the software and/or hardware tools described in the TSS. The evaluator shall verify that the modification fails or that the TSF audits the change as expected according to the description in the TSS.

5.6.2 Key Storage (FPT_KST)

5.6.2.1 Plaintext Key Storage

FPT_KST_EXT.1	Extended: Key Storage

FPT_KST_EXT.1.1 The TSF shall not store any plaintext key material in readable non-volatile memory.

Application Note: The intention of this requirement is that the TOE will not write plaintext keying material to persistent storage. For the purposes of this requirement, plaintext keying material refers to authentication data, passwords, secret/private symmetric keys, private asymmetric keys, data used to derive keys, etc. These values must be stored encrypted.

Additionally, for products entering into evaluation after Quarter 3, 2015, this requirement will also apply to any value derived from passwords. Thus, the TOE cannot store plaintext password hashes for comparison purposes before protected data is decrypted, and the TOE should use key derivation and decryption to verify the Password Authentication Factor.

Assurance Activity:

The evaluator shall consult the TSS section of the ST in performing the assurance activities for this requirement.

In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data.

The evaluator shall ensure that the description also covers how the cryptographic functions in the FCS requirements are being used to perform the encryption functions, including how the KEKs, DEKs, and stored keys are unwrapped, saved, and used by the TOE so as to prevent plaintext from being written to non-volatile storage. The evaluator shall ensure that the TSS describes, for each power-down scenario how the TOE ensures that all keys in non-volatile storage are wrapped with a KEK.

The evaluator shall ensure that the TSS describes how other functions available in the system (e.g., regeneration of the keys) ensure that no unencrypted key material is present in persistent storage.

The evaluator shall review the TSS to determine that it makes a case that key material is not written unencrypted to the persistent storage.

5.6.2.2 No Key Transmission

FPT_KST_EXT.2	Extended: No Key Transmission

FPT_KST_EXT.2.1 The TSF shall not transmit any plaintext key material outside the security boundary of the TOE.

Application Note: For the purposes of this requirement, key material refers to keys, passwords, and other material that is used to derive keys. The intention of this requirement is to prevent the logging of plaintext key information to a service that transmits the information off-device.

In the future, this requirement will apply to symmetric and asymmetric private keys stored in the TOE secure key storage where applications are outside the boundary of the TOE. Thus, the TSF will be required to provide cryptographic key operations (signature, encryption, and decryption) on behalf of applications (FCS_SRV_EXT.1.2) that have access to those keys.

Assurance Activity:

The evaluator shall consult the TSS section of the ST in performing the assurance activities for this requirement. The evaluator shall ensure that the TSS describes the TOE security boundary. The cryptographic module may very well be a particular kernel module, the Operating System, the Application Processor, or up to the entire Mobile Device.

In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data.

The evaluator shall ensure that the TSS describes how other functions available in the system (e.g., regeneration of the keys) ensure that no unencrypted key material is transmitted outside the security boundary of the TOE.

The evaluator shall review the TSS to determine that it makes a case that key material is not transmitted outside the security boundary of the TOE.

5.6.2.3 No Plaintext Key Export

FPT_KST_EXT.3	Extended: No Plaintext Key Export

FPT_KST_EXT.3.1 The TSF shall ensure it is not possible for the TOE user(s) to export plaintext keys.

Application Note: Plaintext keys include DEKs, KEKs, and all keys stored in the secure key storage (FCS_STG_EXT.1). The intent of this requirement is to prevent the plaintext keys from being exported during a backup authorized by the TOE user or administrator.

Assurance Activity:

The ST author will provide a statement of their policy for handling and protecting keys. The evaluator shall check to ensure the TSS describes a policy in line with not exporting either plaintext DEKs, KEKs, or keys stored in the secure key storage.

5.6.3 Self-Test Notification (FPT_NOT)

FPT_NOT_EXT.1	Extended: Self-Test Notification

FPT_NOT_EXT.1.1 The TSF shall transition to non-operational mode and [selection: *log failures in the audit record, notify the administrator, [assignment: other actions], no other actions*] when the following types of failures occur:

- failures of the self-test(s)
- TSF software integrity verification failures
- [selection: *no other failures, [assignment: other failures]*].

Assurance Activity:

The evaluator shall verify that the TSS describes critical failures that may occur and the actions to be taken upon these critical failures.

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

Test 1: The evaluator shall use a tool provided by the developer to modify files and processes in the system that correspond to critical failures specified in the second list. The evaluator shall verify that creating these critical failures causes the device to take the remediation actions specified in the first list.

5.6.4 Reliable Time Stamps (FPT_STM)

FPT_STM.1	Reliable time stamps

FPT_STM.1.1 The TSF shall be able to provide reliable time stamps for its own use.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time. The TSS provides a description of how the time is maintained and considered reliable in the context of each of the time related functions. This documentation must identify whether the TSF uses a NTP server or the carrier's network time as the primary time sources.

The evaluator examines the operational guidance to ensure it describes how to set the time.

Test 1: The evaluator uses the operational guide to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

5.6.5 TSF Functionality Testing (FPT_TST)

5.6.5.1 TSF Cryptographic Functionality Testing

FPT_TST_EXT.1 Extended: TSF Cryptographic Functionality Testing

FPT_TST_EXT.1.1 The TSF shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of all cryptographic functionality.

Application Note: This requirement may be met by performing known answer tests and/or pair-wise consistency tests. The self-tests must be performed before the cryptographic functionality is exercised (for example, during the initialization of a process that utilizes the functionality).

The cryptographic functionality includes the cryptographic operations in FCS_COP, the key generation functions in FCS_CKM, and the random bit generation in FCS_RBG_EXT.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it specifies the self-tests that are performed at start-up. This description must include an outline of the test procedures conducted by the TSF (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 TSS must include any error states that they TSF may enter when self-tests fail, and the conditions and actions necessary to exit the error states and resume normal operation. The evaluator shall verify that the TSS indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user or operator.

The evaluator shall inspect the list of self-tests in the TSS and verify that it includes algorithm self-tests. The algorithm self-tests will typically be conducted using known answer tests.

5.6.5.2 TSF Integrity Testing

FPT_TST_EXT.2 Extended: TSF Integrity Testing

FPT_TST_EXT.2.1 The TSF shall verify the integrity of the bootchain up through the Application Processor OS kernel, and [selection: *all executable code stored in mutable media, [assignment: list of other executable code], no other executable code*], stored in mutable media prior to its execution through the use of [selection: *a digital signature using a hardware-protected asymmetric key, a hardware-protected hash*].

Application Note: The bootchain of the TSF is the sequence of firmware and software, including ROM, bootloader(s), and kernel, which ultimately result in loading the kernel on the Application Processor, regardless of which processor executes that code.

In order to meet this requirement, the hardware protection may be transitive in nature: a hardware-protected public key or hash may be used to verify the mutable bootloader code

which contains a key or hash used by the bootloader to verify the mutable OS kernel code, which contains a key or hash to verify the next layer of executable code, and so on.

The cryptographic mechanism used to verify the (initial) mutable executable code must be protected, such as being implemented in hardware or in read-only memory (ROM). If "all executable code in mutable media" is verified, implementation in hardware or in read-only memory is a natural logical consequence.

At this time, the verification of software executed on other processors stored in mutable media is not required; however, it may be added in the first assignment. If all executable code (including bootloader(s), kernel, device drivers, pre-loaded applications, user-loaded applications, and libraries) is verified, "all executable code stored in mutable media" should be selected.

In the context of this requirement, "hardware-protected" means that a cryptographic value (for example, a public key or a hash) is immutably stored by the device hardware such that if the private key used to digitally sign the software is the manufacturer's private key, the verification of the signature using the public key will succeed. This value need not be protected against unauthorized disclosure, only against unauthorized modification.

Assurance Activity:

The evaluator shall verify that the TSS section of the ST includes a description of the boot procedures, including a description of the entire bootchain, of the software for the TSF's Application Processor. The evaluator shall ensure that before loading the bootloader(s) for the operating system and the kernel, all bootloaders and the kernel software itself is cryptographically verified. For each additional category of executable code verified before execution, the evaluator shall verify that the description in the TSS describes how that software is cryptographically verified.

The evaluator shall verify that the TSS contains a justification for the protection of the cryptographic key or hash, preventing it from being modified by unverified or unauthenticated software. The evaluator shall verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

The evaluator shall verify that the TSS describes each auxiliary boot mode available on the TOE during the boot procedures. The evaluator shall verify that, for each auxiliary boot mode, a description of the cryptographic integrity of the executed code through the kernel is verified before each execution.

The evaluator shall perform the following tests:

Test 1: The evaluator shall perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the TOE properly boots.

Assurance Activity Note: The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 2: The evaluator shall modify a TSF executable that is integrity protected and cause that executable to be successfully loaded by the TSF. The evaluator observes that an integrity violation is triggered and the TOE does not boot. (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that the module was modified so that it was rendered unable to run because its format was corrupt).

[conditional] Test 3: If the ST author indicates that the integrity verification is performed using a public key, the evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1. The evaluator shall digitally sign the TSF executable with a certificate that does not have the Code Signing purpose in the extendedKeyUsage field and verify that an integrity violation is triggered. The evaluator shall repeat the test using a certificate that contains the Code Signing purpose and verify that the integrity verification succeeds. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

5.6.6 Trusted Update (FPT_TUD)

5.6.6.1 Trusted Update: TSF Version Query

FPT_TUD_EXT.1 Extended: Trusted Update: TSF version query

FPT_TUD_EXT.1.1 The TSF shall provide authorized users the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The TSF shall provide authorized users the ability to query the current version of the hardware model of the device.

Application Note: The current version of the hardware model of the device is an identifier that is sufficient to indicate (in tandem with manufacturer documentation) the hardware which comprises the device.

FPT_TUD_EXT.1.3 The TSF shall provide authorized users the ability to query the current version of installed mobile applications.

Application Note: The current version of mobile applications is the name and published version number of each installed mobile application.

Assurance Activity:

The evaluator shall establish a ***test environment*** consisting of the Mobile Device and any supporting software that demonstrates usage of the management functions. This can be test software from the developer, a reference implementation of management software from the developer, or other commercially available software. The evaluator shall set up the Mobile Device and the other software to exercise the management functions according to provided guidance documentation.

Test 1: Using the AGD guidance provided, the evaluator shall test that the administrator and user can query:

- the current version of the TSF operating system and any firmware that can be updated separately
- the hardware model of the TSF
- the current version of all installed mobile applications

The evaluator must review manufacturer documentation to ensure that the hardware model identifier is sufficient to identify the hardware which comprises the device.

5.6.6.2 Trusted Update Verification

FPT_TUD_EXT.2 Extended: Trusted Update Verification

FPT_TUD_EXT.2.1 The TSF shall verify software updates to the Application Processor system software and [selection: *[assignment: other processor system software]*, *no other processor system software*] using a digital signature by the manufacturer prior to installing those updates.

Application Note: The digital signature mechanism is implemented in accordance with FCS_COP.1.1(3).

At this time, this requirement does not required verification of software updates to the software operating outside the Application Processor.

Any change, via a supported mechanism, to software residing in non-volatile storage is deemed a software update. Thus, this requirement applies to TSF software updates regardless of how the software arrives or is delivered to the device. This includes over-the-air (OTA) updates as well as partition images containing software which may be delivered to the device over a wired interface.

FPT_TUD_EXT.2.2 The TSF shall [selection: *never update*, *update only by verified software*] the TSF boot integrity [selection: *key*, *hash*].

Application Note: The key or hash updated via this requirement is used for verifying software before execution in FPT_TST_EXT.2. The key or hash is verified as a part of the digital signature on an update, and the software which performs the update of the key or hash is verified by FPT_TST_EXT.2.

FPT_TUD_EXT.2.3 The TSF shall verify that the digital signature verification key used for TSF updates [selection: *is validated to a public key in the Trust Anchor Database*, *matches a hardware-protected public key*].

Application Note: The ST author shall indicate the method by which the signing key for system software updates is limited and, if selected in FPT_TUD_EXT.2.3, shall indicate how this signing key is protected by the hardware.

If certificates are used, certificates are validated for the purpose of software updates in accordance with FIA_X509_EXT.1 and should be selected in FIA_X509_EXT.2.1. Additionally, FPT_TUD_EXT.2.6 must be included in the ST.

In the context of this requirement, "hardware-protected" means that a cryptographic value (for example, a public key or a hash) is immutably stored by the device hardware such that if the private key used to digitally sign the software is the manufacturer's private key, the verification of the signature using the public key will not succeed. This value need not be protected against unauthorized disclosure, only against unauthorized modification.

Assurance Activity:

The evaluator shall verify that the TSS section of the ST describes all TSF software update mechanisms for updating the system software. The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. The evaluator shall verify that all software and firmware involved in updating the TSF is described and, if multiple stages and software are indicated, that the software/firmware responsible for each stage is indicated and that the stage(s) which perform signature verification of the update are identified.

The evaluator shall verify that the TSS describes the method by which the digital signature is verified and that the public key used to verify the signature is either hardware-protected or is validated to chain to a public key in the Trust Anchor Database. If hardware-protection is selected, the evaluator shall verify that the method of hardware-protection is described and that the ST author has justified why the public key may not be modified by unauthorized parties.

[conditional] If the ST author indicates that software updates to system software running on other processors is verified, the evaluator shall verify that these other processors are listed in the TSS and that the description includes the software update mechanism for these processors, if different than the update mechanism for the software executing on the Application Processor.

[conditional] If the ST author indicates that the public key is used for software update digital signature verification, the evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.

The evaluator shall verify that the developer has provided evidence that the following tests were performed for each available update mechanism:

Test 1: The tester shall try to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.

Test 2: The tester shall digitally sign the update with a key disallowed by the device and verify that installation fails. The tester shall digitally sign the update with the allowed key and verify that installation succeeds.

Test 3: [conditional] The tester shall digitally sign the update with an invalid certificate and verify that update installation fails. The tester shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. The tester shall repeat the test using a valid certificate and a certificate that contains the Code Signing purpose and verify that the application installation succeeds.

Test 4: [conditional] The tester shall repeat this test for the software executing on each processor listed in the first selection. The tester shall attempt to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.

FPT_TUD_EXT.2.4 The TSF shall verify mobile application software using a digital signature mechanism prior to installation.

Application Note: This requirement does not necessitate an X.509v3 certificate or certificate validation. X.509v3 certificates and certificate validation are addressed in FPT_TUD_EXT.2.5.

Assurance Activity:

The evaluator shall verify that the TSS describes how mobile application software is verified at installation. The evaluator shall ensure that this method uses a digital signature.

Test 1: The evaluator shall write, or the developer shall provide access to, an application. The evaluator shall try to install this application without a digitally signature and shall verify that installation fails. The evaluator shall attempt to install a digitally signed application, and verify that installation succeeds.

5.7 Class: TOE Access (FTA)

5.7.1 Session Locking (FTA_SSL)

5.7.1.1 TSF- and User-initiated locked state

FTA_SSL_EXT.1 Extended: TSF- and User-initiated locked state

FTA_SSL_EXT.1.1 The TSF shall transition to a locked state after a time interval of inactivity.

FTA_SSL_EXT.1.2 The TSF shall transition to a locked state after initiation by either the user or the administrator.

FTA_SSL_EXT.1.3 The TSF shall, upon transitioning to the locked state, perform the following operations:

a) clearing or overwriting display devices, obscuring the previous contents;

b) [assignment: *other actions performed upon transitioning to the locked state*].

Application Note: The time interval of inactivity is configured using FMT_SMF_EXT.1 function 2.b. The user-/administrator-initiated lock is specified in FMT_SMF_EXT.1 function 8.

Assurance Activity:

The evaluator shall verify the TSS describes the actions performed upon transitioning to the locked state. The evaluation shall verify that the AGD guidance describes the method of

setting the inactivity interval and of commanding a lock. The evaluator shall verify that the TSS describes the information allowed to be displayed to unauthorized users.

Test 1: The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF_EXT.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.

Test 2: The evaluator shall command the TSF to transition to the locked state according to the AGD guidance as both the user and the administrator. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.

5.7.2 Wireless Network Access (FTA_WSE)

FTA_WSE_EXT.1	Extended: Wireless Network Access

FTA_WSE_EXT.1.1 The TSF shall be able to attempt connections to wireless networks specified as acceptable networks as configured by the administrator in FMT_SMF_EXT.1.

Application Note: The intent of this requirement is to allow the user and administrator to configure the access points to which the TSF may connect and to prevent the TSF from connecting to wireless networks without explicit authorization by the user or the administrator. If, when the device is enrolled, the user is prevented from connecting to wireless networks other than those specified by the enterprise administrator, this management function should be listed in the selection for the requirements FMT_MOF_EXT.1.2. If the management function is not listed in the selection of FMT_MOF_EXT.1.2 requirements, the user may perform the management function as an administrator in order to configure and connect to wireless networks.

Assurance Activity:

The assurance activity for this requirement is performed in conjunction with the assurance activity for FMT_SMF_EXT.1.

5.8 Class: Trusted Path/Channels (FTP)

5.8.1 Trusted Channel Communication (FTP_ITC)

FTP_ITC_EXT.1	Extended: Trusted channel Communication

FTP_ITC_EXT.1.1 The TSF shall use 802.11-2012, 802.1X, and EAP-TLS and [selection, *at least one of: IPsec, TLS, DTLS, HTTPS protocol*] to provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels, provides assured identification of its end points, protects channel data from disclosure, and detects modification of the channel data.

Application Note: The intent of the mandatory portion of the above requirement is to use the cryptographic protocols identified in the requirement to establish and maintain a trusted channel between the TOE and an access point, VPN Gateway, or other trusted IT product.

The ST author shall list which trusted channel protocols are implemented by the Mobile Device. If the ST author selects IPsec, the TSF shall be validated against the "Protection Profile for IPsec Virtual Private Network (VPN) Clients." Annex B contains the requirements for implementing each of the other optional trusted channel protocols. The ST author must include the security functional requirements for the trusted channel protocol selected in **FTP_ITC_EXT.1** in the main body of the ST.

Assured identification of endpoints is performed according to the authentication mechanisms used by the listed trusted channel protocols.

FTP_ITC_EXT.1.2 The TSF shall permit the TSF to initiate communication via the trusted channel.

FTP_ITC_EXT.1.3 The TSF shall initiate communication via the trusted channel for wireless access point connections, administrative communication, configured enterprise connections, and [selection: *OTA updates, no other connections*].

Application Note: In the future, trusted channels will be required for OTA update.

Assurance Activity:

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to access points, VPN Gateways, and other trusted IT products in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specifications. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST. The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to access points, VPN Gateways, and other trusted IT products.

If OTA updates are selected, the TSS shall describe which trusted channel protocol is initiated by the TOE and is used for updates.

The evaluator shall also perform the following tests for each protocol listed:

Test 1: The evaluators shall ensure that the TOE is able to initiate communications with an access point using 802.11-2012 and a pre-shared key, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 2: The evaluators shall ensure that the TOE is able to initiate communications with an access point using 802.11-2012, 802.1x, and EAP-TLS, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 3: [conditional] If IPsec is selected (and the TSF includes a native VPN client), the evaluator shall ensure that the TOE is able to initiate communications with a VPN Gateway, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 4: For any other selected protocol (not tested in Test 1, 2, or 3), the evaluator shall ensure that the TOE is able to initiate communications with a trusted IT product using the protocol, setting up the connection as described in the operational guidance and ensuring that the communication is successful.

Test 5: If OTA updates are selected, the evaluator shall trigger an update request according to the operational guidance and shall ensure that the communication is successful.

Test 6: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext and that a protocol analyzer identifies the traffic as the protocol under testing.

6. Security Assurance Requirements

The Security Objectives for the TOE in Section 4 were constructed to address threats identified in Section 0. The Security Functional Requirements (SFRs) in Section 5 are a formal instantiation of the Security Objectives. The PP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of SARs from CC part 3 that are required in evaluations against this PP. Individual Assurance Activities (Assurance Activities) to be performed are specified both in Section 5 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/user guides for the TOE. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The ITSEF also performs the Assurance Activities contained within Section 5, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The Assurance Activities that are captured in Section 5 also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the PP.

The TOE security assurance requirements are identified in Table 2.

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

Table 2: Security Assurance Requirements

6.1 ASE: Security Target

The ST is evaluated as per ASE activities defined in the CEM. In addition, there may be Assurance Activities specified within Section 5 that call for necessary descriptions to be included in the TSS that are specific to the TOE technology type.

6.2 ADV: Development

The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any additional information required by this PP that is not to be made public (e.g., Entropy Essay).

6.2.1 Basic Functional Specification (ADV_FSP)

The functional specification describes the TOE 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 Assurance Activities for this family 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 in Section 5.

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, AGD_PRE and the API information that is provided to application developers, including the APIs that require privilege to invoke.

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

The API documentation shall include those interfaces required in this profile.

The API documentation shall clearly indicate to which products and versions each available function applies.

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 assurance activities described in Section 5, 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.

6.3 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.

6.3.1 Operational User Guidance (AGD_OPE)

Developer action elements:

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

Application Note: The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages. 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, administrator (e.g., MDM agent), application developer 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 Section 5 and evaluation of the TOE according to the CEM. The following additional information is also required.

The operational guidance shall contain a list of natively installed applications and any relevant version numbers. If any third-parties vendors are permitted to install applications before purchase by the end user or enterprise, these applications shall also be listed.

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. The evaluator shall verify that this process includes the following steps:

46. 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).
47. 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 assurance activities.

6.3.2 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.

6.4 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 is a reflection on the information to be made available for evaluation at this assurance level.

6.4.1 Labelling 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 labelled with its unique reference.

Evaluator action elements:

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

Assurance Activity:

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.

6.4.2 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.1.1D The developer shall provide a configuration list for the TOE.

Content and presentation elements:

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

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

Evaluator action elements:

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

Assurance Activity:

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.

6.4.3 Timely Security Updates (ALC_TSU_EXT)

This component requires the TOE developer, in conjunction with any other necessary parties, to provide information as to how the end-user devices are updated to address security issues in a timely manner. The documentation describes the process of providing updates to the public from the time a security flaw is reported/discovered, to the time an update is released. This description includes the parties involved (e.g., the developer, carriers(s)) and the steps that are performed (e.g., developer testing, carrier testing), including worst case time periods, before an update is made available to the public.

Developer action elements:

ALC_TSU_EXT.1.1D The developer shall provide a description in the TSS of how timely security updates are made to the TOE.

Content and presentation elements:

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

Application Note: The software to be described includes the operating systems of the application processor and the baseband processor, as well as any firmware and applications. The process description includes the TOE developer processes as well as any third-party (carrier) processes. The process description includes each deployment mechanism (e.g., over-the-air updates, per-carrier updates, downloaded updates).

ALC_TSU_EXT.1.2C The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.

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

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

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

Evaluator action elements:

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

Assurance Activity:

The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall

verify that this description addresses the TOE OS, the firmware, and bundled applications, each. The evaluator shall also verify that, in addition to the TOE developer's process, any carrier or other third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days.

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

6.5 Class ATE: Tests

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

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.

6.5.1 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 Section 5 are being met, although some additional testing is specified for SARs in Section 6. 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.

6.6 Class AVA: Vulnerability Assessment

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

6.6.1 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.

A. Rationale

In this PP, the focus in the initial sections of the document is to use a narrative presentation in an attempt to increase the overall comprehensibility of the threats addressed by Mobile Devices; 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 section contains the tabular artefacts that can be used for the assurance activities associated with this document.

A.1 Security Problem Description

A.1.1 Assumptions

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

Assumption Name	Assumption Definition
A.CONFIG	It is assumed that the TOE's security functions are configured correctly in a manner to ensure that the TOE security policies will be enforced on all applicable network traffic flowing among the attached networks.
A.NOTIFY	It is assumed that the mobile user will immediately notify the administrator if the Mobile Device is lost or stolen.
A.PRECAUTION	It is assumed that the mobile user exercises precautions to reduce the risk of loss or theft of the Mobile Device.

Table 3: TOE Assumptions

A.1.2 Threats

The threats listed below are addressed by Mobile Devices and apply to all Mobile Devices.

Threat Name	Threat Definition
T.EAVESDROP	If positioned on a wireless communications channel or elsewhere on the network, attackers may monitor and gain access to data exchanged between the Mobile Device and other endpoints
T.NETWORK	An attacker may initiate communications with the Mobile Device or alter communications between the Mobile Device and other endpoints.
T.PHYSICAL	Loss of confidentiality of user data and credentials may be a result of an attacker gaining physical access to a Mobile Device.
T.FLAWAPP	Malicious or exploitable code could be used knowingly or unknowingly by a developer, possibly resulting in the capability of attacks against the platform's system software.
T.PERSISTENT	An attacker gains and continues to have access the device, resulting it loss of integrity and possible control by both an adversary and legitimate owner.

Table 4: Threats

A.1.3 Organizational Security Policies

No organizational policies have been identified that are specific to Mobile Devices.

A.1.4 Security Problem Definition Correspondence

The following table serves to map the threats and assumptions defined in this PP to the security objectives also defined or identified in this PP.

Threat or Assumption	Security Objectives
A.CONFIG	OE.CONFIG
A.NOTIFY	OE.NOTIFY
A.PRECAUTION	OE.PRECAUTION
T.EAVESDROP	O.COMMS, O.CONFIG, O.AUTH
T.NETWORK	O.COMMS, O.CONFIG, O.AUTH
T.PHYSICAL	O.STORAGE, O.AUTH
T.FLAWAPP	O.COMMS, O.CONFIG, O.AUTH, O.INTEGRITY
T.PERSISTENT	O.INTEGRITY

Table 5: Security Problem Definition Correspondence

A.2 Security Objectives

A.2.1 Security Objectives for the TOE

The following table contains security objectives specific to Mobile Devices.

Security Objective Name	Security Objective Definition
O.COMMS	The TOE will provide the capability to communicate using one (or more) standard protocols as a means to maintain the confidentiality of data that are transmitted outside of the TOE.
O.STORAGE	The TOE will provide the capability to encrypt all user and enterprise data and authentication keys to ensure the confidentiality of data that it stores.
O.CONFIG	The TOE will provide the capability to configure and apply security policies. This ensures the Mobile Device can protect user and enterprise data that it may store or process.
O.AUTH	The TOE will provide the capability to authenticate the user and endpoints of a trusted path to ensure they are communicating with an authorized entity with appropriate privileges.
O.INTEGRITY	The TOE will provide the capability to perform self-tests to ensure the integrity of critical functionality, software/firmware and data has been maintained. The TOE will also provide a means to verify the integrity of downloaded updates.

Table 6: Security Objectives for the TOE

A.2.2 Security Objectives for the Operational Environment

The following table contains security objectives specific to the operational environments for Mobile Devices.

Security Objective Name	Security Objective Definition
OE.CONFIG	TOE administrators will configure the Mobile Device security functions correctly to create the intended security policy
OE.NOTIFY	The Mobile User will immediately notify the administrator if the Mobile Device is lost or stolen.
OE.PRECAUTION	The Mobile User exercises precautions to reduce the risk of loss or theft of the Mobile Device.

Table 7: Security Objectives for the Operational Environment

A.2.3 Security Objective Correspondence

The correspondence between the Security Functional Requirements (SFRs) and Security Objectives identified or defined in this PP is provided in Section 4.

A.3 Security Functional Requirements Category Mapping

Table 9 contains a summary of the SFRs and a category to ease use of the Protection Profile, and Table 8 defines those categories.

Table 8: Category Definitions

Category	Definition
Algorithms	Base cryptographic algorithms.
RBG	Random Bit Generation.
Keys	Cryptographic key management, storage, and generation.
DAR Protection	Data-at-rest protection and wiping.
Authentication	Password Authentication Factor use and locked state.
Certificates	Certificate validation.
Integrity	Software integrity verification.
Access Control	File access control.
Anti-Exploitation Services	TOE services that prevent exploitation.
Auditing	Audit generation and storage.
Management	Settings, policies, commands, and remote management.
Trusted Channel	Authenticated and encrypted protocols and networking.
Bluetooth Trusted Channel	Bluetooth networking security.
WLAN Trusted Channel	WLAN security.

Table 9: SFR Category Mapping

Requirement	Summary	Category
FCS_CKM.1(1)	Key Generation	Algorithms
FCS_CKM.2(1)	Key Establishment	Algorithms
FCS_COP.1(1)	Advanced Encryption Standard	Algorithms
FCS_COP.1(2)	Hashing	Algorithms
FCS_COP.1(3)	Digital Signatures	Algorithms
FCS_COP.1(4)	Keyed-hashes	Algorithms
FCS_COP.1(5)	Password-based Key Derivation Function	Algorithms
FCS_SRV_EXT.1	Cryptographic Services for Apps	Algorithms
FPT_TST_EXT.1	Cryptographic Self-tests	Algorithms
FCS_RBG_EXT.1	Random Bit Generation	RBG
FCS_CKM_EXT.1	Root Encryption Key	Keys
FCS_CKM_EXT.2	Data Encryption Key Generation	Keys
FCS_CKM_EXT.3	Key Encryption Key Generation	Keys
FCS_CKM_EXT.4	Key Destruction	Keys
FCS_CKM_EXT.6	Salt Generation	Keys
FCS_IV_EXT.1	Initialization Vector Generation	Keys
FCS_STG_EXT.1	Key Storage	Keys
FCS_STG_EXT.2	Encryption of Stored Keys	Keys
FCS_STG_EXT.3	Integrity of Stored Keys	Keys
FPT_KST_EXT.1	No Plaintext Key Storage	Keys
FPT_KST_EXT.2	No Plaintext Key Transmission	Keys
FPT_KST_EXT.3	No Plaintext Key Export	Keys
FCS_CKM_EXT.5	TSF Wipe	DAR Protection
FDP_DAR_EXT.1	Data-at-Rest Encryption	DAR Protection
FDP_DAR_EXT.2	Lockscreen Data-at-Rest Encryption	DAR Protection
FIA_UAU_EXT.1	Password Required to Decrypt	DAR Protection
FIA_AFL_EXT.1	Authentication Failure Handling	Authentication
FIA_PMG_EXT.1	Password Size/Complexity Support	Authentication
FIA_TRT_EXT.1	Authentication Throttling	Authentication
FIA_UAU.7	Obscured Password	Authentication
FIA_UAU_EXT.2	Timing of Authentication	Authentication
FIA_UAU_EXT.3	Re-Authentication Conditions	Authentication
FTA_SSL_EXT.1	User- and TSF-Initiated Locking	Authentication
FTA_TAB.1	Banner	Authentication
FDP_STG_EXT.1	Trust Anchor Database	Certificates
FIA_X509_EXT.1	Certificate Validation Rules	Certificates
FIA_X509_EXT.3	Certificate Validation Services to Apps	Certificates
FIA_X509_EXT.4	Certificate Enrollment	Certificates
FPT_NOT_EXT.1	Self-Test Notification and Failure Behavior	Integrity
FPT_TST_EXT.2	Secure Boot	Integrity
FPT_TUD_EXT.2	Trusted Software Updates	Integrity

Requirement	Summary	Category
FDP_ACF_EXT.1	Access Control of System Services & Files	Access Control
FPT_AEX_EXT.1	Address-Space Layout Randomization	Anti-Exploitation Services
FPT_AEX_EXT.2	Memory Page Permissions	Anti-Exploitation Services
FPT_AEX_EXT.3	Overflow Protections	Anti-Exploitation Services
FPT_AEX_EXT.4	Domain Isolation	Anti-Exploitation Services
FPT_BBD_EXT.1	AP Mediation of BP	Anti-Exploitation Services
FPT_STM.1	Timestamps	Auditing
FAU_GEN.1	Audit Log Generation	Auditing
FAU_SAR	Audit Review	Auditing
FAU_SEL.1	Selective Auditing	Auditing
FAU_STG.1	Audit Storage Protection	Auditing
FAU_STG.4	Prevention of Audit Data Loss	Auditing
FMT_MOF_EXT.1.1	Administrator Management Functions	Management
FMT_SMF_EXT.1	All Management Functions	Management
FMT_SMF_EXT.2	Remediation Actions for Unenrollment	Management
FPT_TUD_EXT.1	TSF Version Queries	Management
FCS_HTTPS_EXT.1	HTTPS Protocol	Trusted Channel
FCS_TLSC_EXT.2	TLS Client Protocol	Trusted Channel
FDP_IFC_EXT.1	No Split-Tunneling VPN	Trusted Channel
FDP_UPC_EXT.1	User Data Trusted Channel Communications	Trusted Channel
FIA_X509_EXT.2	Certificate Usage Requirements	Trusted Channel
FTP_ITC_EXT.1	TSF Trusted Channel Communications	Trusted Channel
FCS_DTLS_EXT.1	DTLS	Trusted Channel
FIA_BLT_EXT.1	Bluetooth User Authorization	Bluetooth Trusted Channel
FCS_CKM_EXT.7	Bluetooth Key Generation	Bluetooth Trusted Channel
FDP_BLT_EXT.1	Bluetooth Device Access by Apps	Bluetooth Trusted Channel
FIA_BLT_EXT.2	Bluetooth Authentication	Bluetooth Trusted Channel
FPT_BLT_EXT.1	Bluetooth Profile Limiting	Bluetooth Trusted Channel
FCS_CKM.1(2)	WLAN Key Generation (PTK)	WLAN Trusted Channel
FCS_CKM.2(2)	WLAN Key Distribution (GTK)	WLAN Trusted Channel
FCS_TLSC_EXT.1	EAP-TLS Client Protocol	WLAN Trusted Channel
FIA_PAE_EXT.1	802.1x Protocol	WLAN Trusted Channel
FTA_WSE_EXT.1	WLAN Access	WLAN Trusted Channel
FCS_CKM.1(3)	Suite B WLAN Key Generation (PTK)	WLAN Trusted Channel

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 Mobile Device 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.

No optional items have been identified at this time.

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.

C.1 Cryptographic Key Support (REK)

FCS_CKM_EXT.1	Extended: Cryptographic Key Support

FCS_CKM_EXT.1.4: A REK shall not be able to be read from or exported from the hardware.

Application Note: If "hardware-protected" is selected in FCS_CKM_EXT.1.1, FCS_CKM_EXT.1.4 must be included in the ST.

The lack of a public/documented API for importing or exporting, when a private/undocumented API exists, is not sufficient to meet this requirement.

Assurance Activity:

The assurance activity for this element is performed in conjunction with the assurance activity for the other elements in this component.

C.2 DTLS Protocol (FCS_DTLS)

FCS_DTLS_EXT.1	DTLS Protocol

FCS_DTLS_EXT.1.1 The TSF shall implement the DTLS protocol in accordance with DTLS 1.2 (RFC 6347).

FCS_DTLS_EXT.1.2: The TSF shall implement the requirements in TLS (FCS_TLSC_EXT.2) for the DTLS implementation, except where variations are allowed according to DTLS 1.2 (RFC 6347).

Application Note: Differences between DTLS 1.2 and TLS 1.2 are outlined in RFC 6347; otherwise the protocols are the same. In particular, for the applicable security characteristics defined for the TSF, the two protocols do not differ. Therefore, all application notes and assurance activities that are listed for TLS apply to the DTLS implementation.

FCS_DTLS_EXT.1.3 The TSF shall not establish a trusted communication channel if the peer certificate is deemed invalid.

Application Note: Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

Assurance Activity:

Test 1: The evaluator shall attempt to establish a connection with a DTLS server, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as DTLS.

Other tests are performed in conjunction with the Assurance Activity listed for FCS_TLSC_EXT.2.

Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test.

Test 2: 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 to the Trust Anchor Database 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.

C.3 TLS Client Protocol (FCS_TLSC)

C.3.1 EAP-TLS Protocol

FCS_TLSC_EXT.1	Extended:EAP-TLS Protocol

FCS_TLSC_EXT.1.5 The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: *secp256r1, secp384r1, secp521r1*] and no other curves.

Application Note: This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1(3) and FCS_CKM.1(1) and FCS_CKM.2(1). This extension is required for clients supporting Elliptic Curve ciphersuites.

Assurance Activity:

The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured. If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

The evaluator shall also perform the following test:

Test: The evaluator shall configure the server to perform an ECDHE key exchange message in the TLS connection using a non-supported ECDHE curve (for example, P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

C.3.2 TLS Client Protocol

FCS_TLSC_EXT.2	Extended: TLS Protocol

FCS_TLSC_EXT.2.5 The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [selection: *secp256r1, secp384r1, secp521r1*] and no other curves.

Application Note: This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1(3) and FCS_CKM.1(1) and FCS_CKM.2(1). This extension is required for clients supporting Elliptic Curve ciphersuites.

Assurance Activity:

The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured. If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

The evaluator shall also perform the following test:

Test 1: The evaluator shall configure the server to perform an ECDHE key exchange message in the TLS connection using a non-supported ECDHE curve (for example, P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

C.4 TSF Integrity Testing (FPT_TST)

FPT_TST_EXT.2	Extended: TSF Integrity Testing

FPT_TST_EXT.2.2 The TSF shall not execute code if the code signing certificate is deemed invalid.

Application Note: Certificates may optionally be used for code signing for integrity verification (FPT_TST_EXT.2.1). If "code signing for integrity verification*"* is selected in FIA_X509_EXT.2.1, FPT_TST_EXT.2.2 must be included in the ST.

Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

Assurance Activity:

Testing for this element are performed in conjunction with the assurance activities for FPT_TST_EXT.2.1.

C.5 Trusted Update (FPT_TUD)

FPT_TUD_EXT.2 Extended: Trusted Update Verification

FPT_TUD_EXT.2.6 The TSF shall not install code if the code signing certificate is deemed invalid.

Application Note: Certificates may optionally be used for code signing of system software updates (FPT_TUD_EXT.2.3) and of mobile applications (FPT_TUD_EXT.2.5). This element must be included in the ST if certificates are used for either update element. If either "code signing for system software updates" or "code signing for mobile applications" is selected in FIA_X509_EXT.2.1, FPT_TUD_EXT.2.6 must be included in the ST.

Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

Assurance Activity:

Testing for this element are performed in conjunction with the assurance activities for FPT_TUD_EXT.2.3 and FPT_TUD_EXT.2.5.

D. Objective Requirements

This Appendix includes requirements that specify security functionality which also addresses threats. The requirements are not currently mandated in the body of this PP as they describe security functionality not yet widely-available in commercial technology. However, these requirements may be included in the ST such that the TOE is still conformant to this PP, and it is expected that they be included as soon as possible.

D.1 Class: Security Audit (FAU)

D.1.1 Audit Data Generation (FAU_GEN)

FAU_GEN.1	Audit Data Generation

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

1. Start-up and shutdown of the audit functions;
2. All administrative actions;
3. Start-up and shutdown of the OS and kernel;
4. Insertion or removal of removable media;
5. Establishment of a synchronizing connection;
6. Specifically defined auditable events in Table 10;
7. [selection: *Audit records reaching [assignment: integer value less than 100] percentage of audit capacity, [assignment: other auditable events derived from this profile]*].

Application Note: Audit data generation will be required for products entering into evaluation after Quarter 3, 2015.

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

1. Date and time of the event;
2. type of event;
3. subject identity;
4. the outcome (success or failure) of the event; and
5. additional information in Table 10.

Application Note: The subject identity is usually the process name/ID. The event type is often indicated by a severity level, for example ‘info’, ‘warning’, or ‘error’.

Assurance Activity:

The evaluator shall check the administrative guide and ensure that it lists all of the auditable events and 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 check to make sure that every audit event type mandated by the PP is described and that the description of the fields contains the information required in FAU_GEN.1.2.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP including those listed in the Management section. The evaluator shall examine the administrative guide and make a determination of which administrative commands are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are security relevant with respect to this PP. The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.

The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the provided table and administrative actions. This should include all instances of an event. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. For administrative actions, the evaluator shall test that each action determined by the evaluator above to be security relevant in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

The evaluator shall test the TOE's ability to correctly generate audit records in each of the supported auxiliary modes, exercising as much of the TOE functionality as is available in that mode and ensuring that the audit records are correctly generated.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.

Table 10: Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.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.1	None.	
FAU_STG.4	None.	
FCS_CKM_EXT.1	[selection: generation of a REK, none]	No additional information.
FCS_CKM_EXT.2	None.	
FCS_CKM_EXT.3	None.	
FCS_CKM_EXT.4	None.	
FCS_CKM_EXT.5	Success or failure of the wipe.	No additional information.
FCS_CKM_EXT.6	None.	
FCS_CKM_EXT.7	None.	
FCS_CKM.1(1)	Failure of key generation activity for authentication keys.	No additional information.
FCS_CKM.1(2)	None.	

Requirement	Auditable Events	Additional Audit Record Contents
FCS_CKM.1(3)	None.	
FCS_CKM.2(1)	None.	
FCS_CKM.2(2)	None.	
FCS_COP.1	None.	
FCS_DTLS_EXT.1	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate.
FCS_HTTPS_EXT.1	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate. [selection: User's authorization decision, no additional information].
FCS_IV_EXT.1	None.	
FCS_RBG_EXT.1	Failure of the randomization process.	No additional information.
FCS_SRV_EXT.1	None.	
FCS_STG_EXT.1	Import or destruction of key. [selection: Exceptions to use and destruction rules, No other events]	Identity of key. Role and identity of requestor.
FCS_STG_EXT.2	None.	
FCS_STG_EXT.3	Failure to verify integrity of stored key.	Identity of key being verified.
FCS_TLSC_EXT.1	Failure to establish an EAP-TLS session.	Reason for failure.
	Establishment/termination of an EAP-TLS session.	Non-TOE endpoint of connection.
FCS_TLSC_EXT.2	Failure to establish a TLS session.	Reason for failure.
	Failure to verify presented identifier.	Presented identifier and reference identifier.
	Establishment/termination of a TLS session.	Non-TOE endpoint of connection.
FDP_ACF_EXT.1	None.	
FDP_BLT_EXT.1	None.	
FDP_DAR_EXT.1	Failure to encrypt/decrypt data.	No additional information.
FDP_DAR_EXT.2	Failure to encrypt/decrypt data.	No additional information.
FDP_IFC_EXT.1	None.	
FDP_STG_EXT.1	Addition or removal of certificate from Trust Anchor Database.	Subject name of certificate.
FDP_UPC_EXT.1	Application initiation of trusted channel.	Name of application. Trusted channel protocol. Non-TOE endpoint of connection.
FIA_AFL_EXT.1	Excess of authentication failure limit.	No additional information.
FIA_BLT_EXT.1	User authorization of Bluetooth device. User authorization for local Bluetooth service.	User authorization decision. Bluetooth address and name of device. Bluetooth profile. Identity of local service.
FIA_BLT_EXT.2	Initiation of Bluetooth connection.	Bluetooth address and name of device.
	Failure of Bluetooth connection.	Reason for failure.
FIA_PAE_EXT.1	None.	
FIA_PMG_EXT.1	None.	
FIA_TRT_EXT.1	None.	
FIA_UAU_EXT.1	None.	
FIA_UAU_EXT.2	Action performed before authentication.	No additional information.
FIA_UAU_EXT.3	User changes Password Authentication	No additional information.

Requirement	Auditable Events	Additional Audit Record Contents
	Factor.	
FIA_UAU.7	None.	
FIA_X509_EXT.1	Failure to validate X.509v3 certificate.	Reason for failure of validation.
FIA_X509_EXT.2	Failure to establish connection to determine revocation status.	No additional information.
FIA_X509_EXT.3	None.	
FIA_X509_EXT.4	Generation of Certificate Enrollment Request.	Issuer and Subject name of EST Server. Method of authentication. Issuer and Subject name of certificate used to authenticate. Content of Certificate Request Message.
	Success or failure of enrollment.	Issuer and Subject name of added certificate or reason for failure.
	Update of EST Trust Anchor Database	Subject name of added Root CA.
FMT_MOF_EXT.1.1	None.	
FMT_MOF_EXT.1.2	None.	
FMT_SMF_EXT.1	Change of settings.	Role of user that changed setting. Value of new setting.
	Success or failure of function.	Role of user that performed function. Function performed. Reason for failure.
	Initiation of software update.	Version of update.
	Initiation of application installation or update.	Name and version of application.
FMT_SMF_EXT.2	Unenrollment.	Identity of administrator. Remediation action performed.
FPT_AEX_EXT.1	None.	
FPT_AEX_EXT.2	None.	
FPT_AEX_EXT.3	None.	
FPT_AEX_EXT.4	Blocked attempt to modify TSF data.	Identity of subject. Identity of TSF data.
FPT_BBD_EXT.1	None.	
FPT_BLT_EXT.1	None.	
FPT_KST_EXT.1	None.	
FPT_KST_EXT.2	None.	
FPT_KST_EXT.3	None.	
FPT_NOT_EXT.1	[selection: Measurement of TSF software, none].	[selection: integrity verification value, no additional data].
FPT_STM.1	None.	
FPT_TST_EXT.1	Initiation of self-test. Failure of self-test.	Algorithm that caused failure.
FPT_TST_EXT.2	Start-up of TOE.	Boot Mode.
	[selection: detected integrity violation, none].	[selection: The TSF code file that caused the integrity violation, no additional information].
FPT_TUD_EXT.1	None.	
FPT_TUD_EXT.2	Success or failure of signature verification for software updates.	
	Success or failure of signature verification for applications.	
FTA_SSL_EXT.1	None.	
FTA_TAB.1	Change in banner setting.	No additional information.

Requirement	Auditable Events	Additional Audit Record Contents
FTA_WSE_EXT.1	All attempts to connect to access points.	Identity of access point.
FTP_ITC_EXT.1	Initiation and termination of trusted channel.	Trusted channel protocol. Non-TOE endpoint of connection.

D.1.2 Security Audit Review (FAU_SAR)

FAU_SAR.1	Audit Review

FAU_SAR.1.1 The TSF shall provide [**the administrator**] with the capability to read [**all audited events and record contents**] from the audit records.

Application Note: The administrator shall have access to read the audit record, perhaps through an API or via an MDM Agent which transfers the local records stored on the TOE to the MDM Server where the enterprise administrator may view them. If this requirement is included in the ST, function 32 shall be included in the selection of FMT_SMF_EXT.1.

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

Assurance Activity:

The assurance activity for this requirement is performed in conjunction with test 32 of FMT_SMF_EXT.1.

D.1.3 Security Audit Event Selection (FAU_SEL)

FAU_SEL.1	Selective Audit

FAU_SEL.1.1 The TSF 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. This can be configured through an interface on the TSF for a user/administrator to invoke. 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.

D.1.4 Security Audit Event Storage (FAU_STG)

FAU_STG.1	Audit Storage Protection

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

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

Application Note: Audit storage will be required for products entering into evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall ensure that the TSS lists the location of all logs and the access controls of those files such that unauthorized modification and deletion are prevented.

Test 1: The evaluator shall attempt to delete the audit trail as an unauthorized user and shall verify that the attempt fails.

Test 2: The evaluator shall attempt to modify the audit trail as an unauthorized application and shall verify that the attempt fails.

FAU_STG.4	Prevention of Audit Data Loss

FAU_STG.4.1 The TSF shall overwrite the oldest stored audit records if the audit trail is full.

Application Note: Audit storage will be required for products entering into evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall examine the TSS to ensure that it describes the size limits on the audit records, the detection of a full audit trail, and the action(s) taken by the TSF when the audit trail is full. The evaluator shall ensure that the action(s) results in the deletion or overwrite of the oldest stored record.

D.2 Class: Cryptographic Services (FCS)

D.2.1 Cryptographic Key Management (FCS_CKM)

D.2.1.1 Cryptographic Key Generation (WLAN)

FCS_CKM.1(3)	Cryptographic key generation

FCS_CKM.1.1(3) The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [***PRF-704***] and specified cryptographic key size [***256 bits***] using a Random Bit Generator as specified in FCS_RBG_EXT.1 that meet the following: [***IEEE 802.11ac-2013***].

Application Note: The cryptographic key derivation algorithm required by IEEE 802.11ac-2013 (Section 11.6.1.2) and verified in WPA2 certification is PRF-704 which uses the HMAC-SHA-384 function and outputs 704 bits.

This requirement applies only to the keys that are generated/derived for the communications between the access point and the client once the client has been authenticated. It refers to the derivation of the PTK from the PMK, which is done using a random value generated by the RBG specified in this PP, the HMAC function using SHA-384 as specified in this PP, as well as other information. This is specified in IEEE 802.11ac-2013 primarily in chapter 11.

Assurance Activity:

The cryptographic primitives will be verified through assurance activities specified elsewhere in this PP. The evaluator shall verify that the TSS describes how the primitives defined and implemented by this PP are used by the TOE in establishing and maintaining secure connectivity to the wireless clients. The TSS shall also provide a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed (e.g. WPA2 certification). The evaluator shall ensure that the description of the testing methodology is of sufficient detail to determine the extent to which the details of the protocol specifics are tested.

The evaluator shall also perform the following test:

Step 1 - The evaluator shall use a packet sniffing tool between the wireless access point and TOE. The evaluator shall turn on the sniffing tool and successfully connect the TOE to the access point.

Step 2 – The evaluator shall verify the TOE advertises 00-0F-AC:12 as a supported Authentication and Key Management (AKM) suite and either 00-0F-AC:9 or 00-0F-AC:10 as a supported cipher suite in capture 802.11 beacon and probe response messages.

D.2.1.2 Cryptographic Key Generation (Bluetooth)

FCS_CKM_EXT.7	Extended: Bluetooth Key Generation

FCS_CKM_EXT.7.1 The TSF shall randomly generate public/private ECDH key pairs every [assignment: *frequency of and/or criteria for new key pair generation*].

Application Note: There are likely multiple acceptable ways of keeping ECDH key pairs adequately fresh, including a time-based approach such that the same key pairs will not be used for more than, for instance, 24 hours. Alternatively, the criteria might be linked to the number of passed or failed authentication attempts. As a starting point to determine reasonable authentication attempt-based replacement criteria, note that the Bluetooth specification (v4.1, Vol. 2, 5.1) suggests mitigating repeated authentication attempts by changing a device's private key after three failed authentication attempts from any BD_ADDR, after ten successful pairings from any BD_ADDR, or after a combination of these such that any three successful pairings count as one failed pairing.

This requirement will be moved to Section 5 and will be mandatory for products entering into evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall ensure that the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs. In particular, the evaluator shall ensure that the implementation does not permit the use of static ECDH key pairs.

The evaluator shall perform the following test:

Test 1: The evaluator shall perform the following steps:

Step 1 - Pair the TOE to a remote Bluetooth device and record the public key currently in use by the TOE. (This public key can be obtained using a Bluetooth protocol analyzer to inspect packets exchanged during paring.)

Step 2 - Perform necessary actions to generate new ECDH public/private key pairs. (Note that this test step depends on how the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs.)

Step 3 - Pair the TOE to a remote Bluetooth device and again record the public key currently in use by the TOE.

Step 4 - Verify that the public key in Step 1 differs from the public key in Step 3.

D.2.2 Random Bit Generation (FCS_RBG)

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

FCS_RBG_EXT.1.4 The TSF shall allow applications to add data to the deterministic RBG using the Personalization String as defined in SP 800-90A.

Application Note: As specified in SP 800-90A the TSF shall not count data input from an application towards the entropy required by FCS_RBG_EXT.1. Thus, the TSF shall not allow the only input to the RBG seed to be from an application.

Assurance Activity: The evaluator shall verify that this function is included as an interface to the RBG in the documentation required by Appendix E and that the behaviour of the RBG following a call to this interface is described. The evaluator shall also verify that the documentation of the RBG describes the conditions of use and possible values for the

Personalization String input to the SP 800-90A specified DRBG. The evaluator shall also perform the following test.

Test 1: The evaluator shall write, or the developer shall provide, an application that adds data to the RBG via the Personalization String. The evaluator shall verify that the request succeeds.

FCS_RBG_EXT.1.5 The TSF shall save the state of the deterministic RBG at power-off, and shall use this state as input to the deterministic RBG at startup.

Application Note: The capability to add the state saved at power-off as input to the RBG prevents an RBG that is slow to gather entropy from producing the same output regularly and across reboots. Since there is no guarantee of the protections provided when the state is stored (or a requirement for any such protection), it is assumed that the state is 'known', and therefore cannot contribute entropy to the RBG, but can introduce enough variation that the initial RBG values are not predictable and exploitable.

Assurance Activity:

The assurance activity for this requirement is captured in the RBG documentation for Appendix E. The evaluator shall verify that the documentation describes how the state is generated so as to be available for the next startup, how the state is used as input to the DRBG, and any protection measures used for the state while the TOE is powered off.

D.2.3 Cryptographic Algorithm Services (FCS_SRV)

FCS_SRV_EXT.1	Extended: Cryptographic Algorithm Services

FCS_SRV_EXT.1.2 The TSF shall provide a mechanism for applications to request the TSF to perform the following cryptographic operations:

- Algorithms in FCS_COP.1(1)
- Algorithms in FCS_COP.1(3)

by keys stored in the secure key storage.

Application Note: In the future, the TOE will not be permitted to transmit plaintext key material beyond the TOE, including to applications running on the TOE. The TOE will, therefore, be required to perform cryptographic operations on behalf of applications using the keys stored in the TOE's secure key storage.

Assurance Activity:

The evaluator shall verify that the API documentation for the secure key storage includes the cryptographic operations by the stored keys.

The evaluator shall write, or the developer shall provide access to, an application that requests cryptographic operations of stored keys by the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. The evaluator shall use these APIs to test the functionality of the secure key storage according to the Assurance Activities in FCS_STG_EXT.1.

D.2.4 TLS Client Protocol (FCS_TLSC)

D.2.4.1EAP-TLS Client Protocol

FCS_TLSC_EXT.1	Extended: EAP-TLS Protocol

FCS_TLSC_EXT.1.6 The TSF shall present the signature_algorithms extension in the Client Hello with the supported_signature_algorithms value containing the following hash algorithms: [selection: *SHA256, SHA384, SHA512*] and no other hash algorithms.

Application Note: This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the client and limits the server to the supported hashes for the purpose of digital signature generation by the server. The signature_algorithm extension is only supported by TLS 1.2.

Support for the signature_algorithms extension will be required for products entering into evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall verify that TSS describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured. If the TSS indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension.

The evaluator shall also perform the following test:

- *Test:* The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's HashAlgorithm enumeration within the signature_algorithms extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

FCS_TLSC_EXT.1.7 The TSF shall support secure renegotiation through use of the "renegotiation_info" TLS extension in accordance with RFC 5746.

FCS_TLSC_EXT.1.8 The TSF shall include [selection: choose only one of: *renegotiation_info extension, TLS_EMPTY_RENEGOTIATION_INFO_SCSV ciphersuite*] in the ClientHello message.

Application Note: RFC 5746 defines an extension to TLS that binds renegotiation handshakes to the cryptography in the original handshake.

The ciphersuite included in the selection is a means for clients to be compatible with servers that don't support the extension. It is recommended that client implementations support both the ciphersuite and the extension.

Assurance Activity:

The evaluator shall perform the following tests:

Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the "renegotiation_info" field or the SCSV ciphersuite is included in the ClientHello packet during the initial handshake.

Test 2: The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the "renegotiation_info" extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

Test 3: The evaluator shall verify that ServerHello messages received during secure renegotiation contain the "renegotiation_info" extension. The evaluator shall modify either the "client_verify_data" or "server_verify_data" value and verify that the client terminates the connection.

D.2.4.2TLS Client Protocol

FCS_TLSC_EXT.2	Extended: TLS Protocol

FCS_TLSC_EXT.2.6 The TSF shall present the signature_algorithms extension in the Client Hello with the supported_signature_algorithms value containing the following hash algorithms: [selection: *SHA256, SHA384, SHA512*] and no other hash algorithms.

Application Note: This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the client and limits the server to the supported hashes for the purpose of digital signature generation by the server. The signature_algorithm extension is only supported by TLS 1.2.

Support for the signature_algorithms extension will be required for products entering into evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall verify that TSS describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured. If the TSS indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension.

The evaluator shall also perform the following test:

- *Test:* The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's HashAlgorithm enumeration within the signature_algorithms extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

FCS_TLSC_EXT.2.7 The TSF shall support secure renegotiation through use of the "renegotiation_info" TLS extension in accordance with RFC 5746.

FCS_TLSC_EXT.2.8 The TSF shall include [selection: choose only one of: *renegotiation_info extension, TLS_EMPTY_RENEGOTIATION_INFO_SCSV ciphersuite*] in the ClientHello message.

Application Note: RFC 5746 defines an extension to TLS that binds renegotiation handshakes to the cryptography in the original handshake.

The ciphersuite included in the selection is a means for clients to be compatible with servers that don't support the extension. It is recommended that client implementations support both the ciphersuite and the extension.

Assurance Activity:

The evaluator shall perform the following tests:

Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the "renegotiation_info" field or the SCSV ciphersuite is included in the ClientHello packet during the initial handshake.

Test 2: The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the "renegotiation_info" extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

Test 3: The evaluator shall verify that ServerHello messages received during secure renegotiation contain the "renegotiation_info" extension. The evaluator shall modify either the "client_verify_data" or "server_verify_data" value and verify that the client terminates the connection.

D.3 Class: User Data Protection (FDP)

D.3.1 Access Control (FDP_ACF)

FDP_ACF_EXT.1	Extended: Security attribute based access control

FDP_ACF_EXT.1.3 The TSF shall enforce an access control policy that prohibits an application from granting both write and execute permission to a file on the device.

Assurance Activity:

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

Test 1: The evaluator shall write, or the developer shall provide, an application which attempts to store a file with both write and execute permissions. The evaluator shall verify that this action fails and that the permissions on the file are not simultaneously write and execute.

Test 2: The evaluator shall traverse the file system examining the permission on each TSF file to verify that no file has both write and execute permissions set.

D.3.2 Application Bluetooth Device Access (FDP_BLT)

FDP_BLT_EXT.1	Extended:	Limitation of Bluetooth Device Access

FDP_BLT_EXT.1.1 The TSF shall limit the applications that may communicate with a particular paired Bluetooth device.

Application Note: Not every application with privileges to use Bluetooth should be permitted to communicate with every paired Bluetooth device. For example, the TSF may choose to require that only the application that initiated the current connection may communicate with the device, or it may strictly tie the paired device to the first application that makes a socket connection to the device following initial pairing. Additionally, for more flexibility, the TSF may choose to provide the user with a way to select which applications on the device may communicate with or observe communications with each paired Bluetooth device.

Assurance Activity:

The evaluator shall ensure that the TSS describes the mechanism used to prevent unrestricted access to paired Bluetooth devices (and/or their communication data) by every application with access to the Bluetooth system service on the TOE (as listed in FDP_ACF_EXT.1). The evaluator shall verify that this method either restricts access to a single application or provides explicit control of the applications that may communicate with the paired Bluetooth device.

D.3.3 Data-At-Rest Protection (FDP_DAR)

In the current version of the mandatory requirements in Section 0, only two levels of data-at-rest protection are addressed: TSF data and Protected Data (and keys). For products entering into evaluation after Quarter 3, 2015, an additional level of data-at-rest protection will be added: sensitive data. Table 11 addresses the level of protection required for each level of data-at-rest. If the FDP_DAR_EXT.2 requirements are not included in the body, all non-TSF data, including data that may otherwise be considered sensitive data is treated at the protected data level. Additional information about these data levels can be found in the glossary (Section 1.2).

Data Level	Protection Required
TSF Data	TSF data does not require confidentiality, but does require integrity protection (FPT_TST_EXT.2).
Protected Data	Protected data is encrypted while powered off. (FDP_DAR_EXT.1)
Sensitive Data	Sensitive data is encrypted while in the locked state. (FDP_DAR_EXT.2)

Table 11: Protection of Data Levels

All keys, protected data, and sensitive data must ultimately be protected by the REK. Sensitive data must be protected by the password in addition to the REK. In particular, Figure 3 has KEKs protected according to these requirements: DEK_1 would be appropriate for sensitive data, DEK_2 would not be appropriate for sensitive data, K_1 is not considered a sensitive key, and K_2 is considered a sensitive key.

These requirements include a capability for encrypting sensitive data received while in the locked state, which may be considered a separate sub-category of sensitive data. This capability may be met by a key transport scheme (RSA) by using a public key to encrypt the DEK while protecting the corresponding private key with a password-derived KEK.

This capability may also be met by a key agreement scheme. To do so, the device generates a device-wide sensitive data asymmetric pair (the private key of which is protected by a password-derived KEK) and an asymmetric pair for the received sensitive data to be stored. In order to store the sensitive data, the device-wide public key and data private key are used to generate a shared secret which can be used as a KEK or a DEK. The data private key and shared secret are cleared after the data is encrypted and the data public key stored. Thus, no key material is available in the locked state to decrypt the newly stored data. Upon unlock, the device-wide private key is decrypted and is used with each data public key to regenerate the shared secret and decrypt the stored data. Figure 4, below, illustrates this scheme.

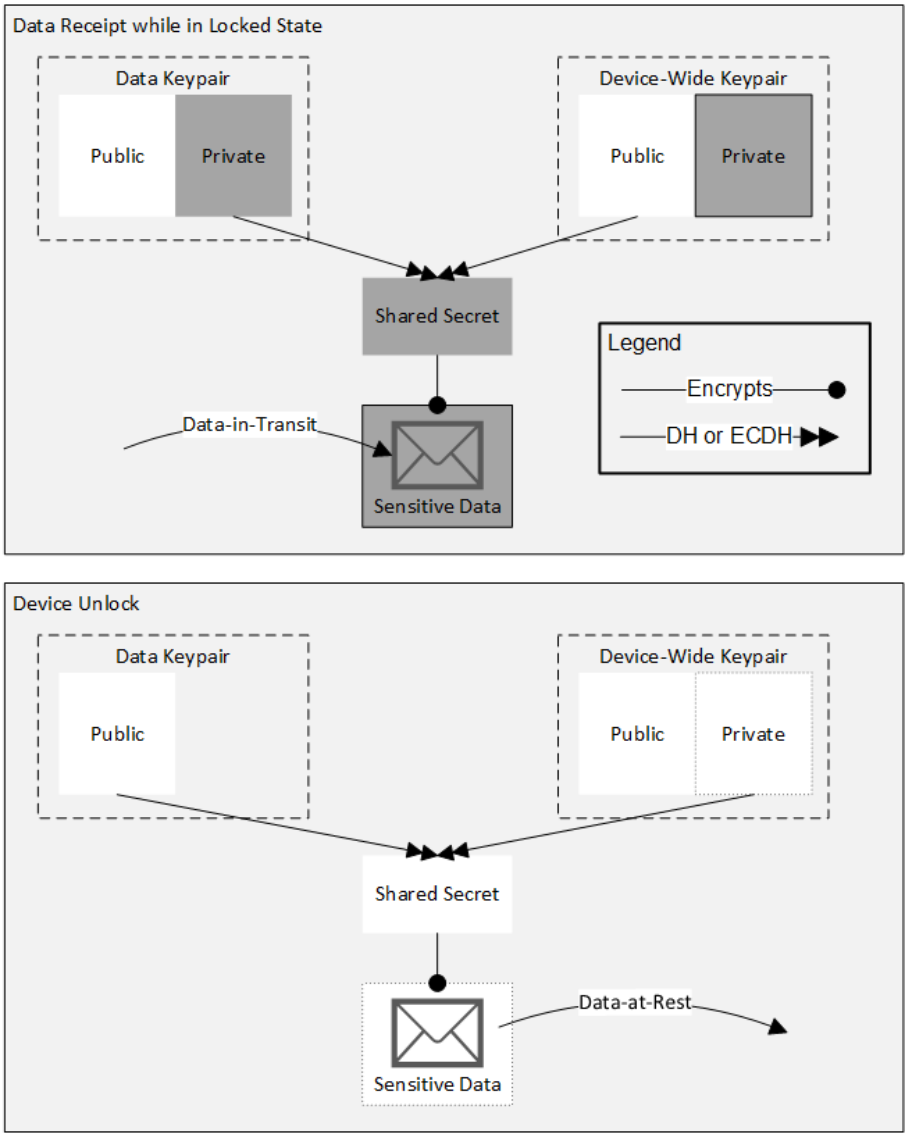

Figure 4: Key Agreement Scheme for Encrypting Received Senstive Data in the Locked State

FDP_DAR_EXT.2	**Extended: Sensitive Data Encryption**

FDP_DAR_EXT.2.1 The TSF shall provide a mechanism for applications to mark data and keys as sensitive.

Application Note: Data and keys that have been marked as sensitive will be subject to certain restrictions (through other requirements) in both the locked and unlocked states of the Mobile Device. This mechanism allows an application to choose those data and keys under its control to be subject to those requirements.

In the future, this PP may require that all data and key created by applications will default to the "sensitive" marking, requiring an explicit "non-sensitive" marking rather than an explicit "sensitive" marking.

Assurance Activity:

The evaluator shall verify that the TSS includes a description of which data stored by the TSF (such as by native applications) is treated as sensitive. This data may include all or some user or enterprise data and must be specific regarding the level of protection of email, contacts, calendar appointments, messages, and documents.

The evaluator shall examine the TSS to determine that it describes the mechanism that is provided for applications to use to mark data and keys as sensitive. This description shall also contain information reflecting how data and keys marked in this manner are distinguished from data and keys that are not (for instance, tagging, segregation in a "special" area of memory or container, etc.).

Test 1: The evaluator shall enable encryption of sensitive data and require user authentication according to the AGD guidance. The evaluator shall try to access and create sensitive data (as defined in the ST and either by creating a file or using an application to generate sensitive data) in order to verify that no other user interaction is required.

FDP_DAR_EXT.2.2 The TSF shall use an asymmetric key scheme to encrypt and store sensitive data received while the product is locked.

Application Note: Sensitive data is encrypted according to FDP_DAR_EXT.1.2. The asymmetric key scheme must be performed in accordance with FCS_CKM.2(1).

The intent of this requirement is to allow the device to receive sensitive data while locked and to store the received data in such a way as to prevent unauthorized parties from decrypting it while in the locked state. If only a subset of sensitive data may be received in the locked state, this subset must be described in the TSS.

Key material must be cleared when no longer needed according to FCS_CKM_EXT.4. For keys (or key material used to derive those keys) protecting sensitive data received in the locked state, “no longer needed” includes “while in the locked state.” For example, in the first key scheme, this includes the DEK protecting the received data as soon as the data is encrypted. In the second key scheme this includes the private key for the data asymmetric pair, the generated shared secret, and any generated DEKs. Of course, both schemes require that a private key of an asymmetric pair (the RSA private key and the device-wide private key, respectively) be cleared when transitioning to the locked state.

Assurance Activity:

The evaluator shall review the TSS section of the ST to determine that the TSS includes a description of the process of receiving sensitive data while the device is in a locked state. The evaluator shall also verify that the description indicates if sensitive data that may be received in the locked state is treated differently than sensitive data that cannot be received in the locked state. The description shall include the key scheme for encrypting and storing the received data, which must involve an asymmetric key and must prevent the sensitive data-at-rest from being decrypted by wiping all key material used to derive or encrypt the data (as described in the application note). The introduction to this section provides two different schemes that meet the requirements, but other solutions may address this requirement.

The evaluator shall perform the tests in FCS_CKM_EXT.4 for all key material no longer needed while in the locked state and shall ensure that keys for the asymmetric scheme are addressed in the tests performed when transitioning to the locked state.

FDP_DAR_EXT.2.3 The TSF shall encrypt any stored symmetric key and any stored private key of the asymmetric key(s) used for the protection of sensitive data according to FCS_STG_EXT.2.1 selection 2.

Application Note: Symmetric keys used to encrypt sensitive data while the TSF is in the unlocked state must be encrypted with (or chain to a KEK encrypted with) the REK and password-derived KEK. A stored private key of the asymmetric key scheme for encrypting data in the locked state must be encrypted with (or chain to a KEK encrypted with) the REK and password-derived KEK.

Assurance Activity:

The evaluator shall verify that the key hierarchy section of the TSS required for FCS_STG_EXT.2.1 includes the symmetric encryption keys (DEKs) used to encrypt sensitive data. The evaluator shall ensure that these DEKs are encrypted by a key encrypted with (or chain to a KEK encrypted with) the REK and password-derived KEK.

The evaluator shall verify that the TSS section of the ST that describes the asymmetric key scheme includes the protection of any private keys of the asymmetric pairs. The evaluator shall ensure that any private keys that are not wiped and are stored by the TSF are stored encrypted by a key encrypted with (or chain to a KEK encrypted with) the REK and password-derived KEK.

FDP_DAR_EXT.2.4 The TSF shall decrypt the sensitive data that was received while in the locked state upon transitioning to the unlocked state using the asymmetric key scheme and shall re-encrypt that sensitive data using the symmetric key scheme.

Assurance Activity:

The evaluator shall verify that the TSS section of the ST that describes the asymmetric key scheme includes a description of the actions taken by the TSF for the purposes of DAR upon transitioning to the unlocked state. These actions shall minimally include decrypting all received data using the asymmetric key scheme and re-encrypting with the symmetric key scheme used to store data while the device is unlocked.

D.4 Class: Identification and Authentication (FIA)

D.4.1 Bluetooth Authorization and Authentication (FIA_BLT)

D.4.1.1 Bluetooth User Authorization

FIA_BLT_EXT.1	Extended: Bluetooth User Authorization

FIA_BLT_EXT.1.2 The TSF shall require explicit user authorization before granting trusted remote devices access to services associated with the following Bluetooth profiles: [assignment: *list of Bluetooth profiles*], and shall require explicit user authorization before granting untrusted remote devices access to services associated with the following Bluetooth profiles: [assignment: *list of Bluetooth profiles*].

Application Note: In addition to pairing, it may be appropriate to require explicit user action to authorize a particular remote device to access certain Bluetooth services. The TSF may

choose to require this additional action for all devices, or only for those devices that do not have a required level of trust.

The TSF might designate certain devices as having a trusted device relationship with the TOE and granting them "blanket" access to all services. However, it is strongly preferable that instead, for each service, the TSF maintain a list of devices trusted to use that particular service.

Furthermore, it may be the case that the TSF allows movement of devices from the untrusted to the trusted category for a particular service after the user provides explicit authorization for the device to use the service. For example, it may be appropriate to require that the user provide explicit, manual authorization before a remote device may use the OBEX service for an object transfer the first time. The user might be given the option to permit future connections to that service by the particular device without requiring explicit authorization each time.

The ST author shall list all Bluetooth profiles and services for which explicit user authorization is required before a remote device can gain access. Any difference in behavior based on whether or not the device has a trusted relationship with the TOE for that service must be specified.

Assurance Activity:

The evaluator shall perform the following tests for each service protected according to this requirement:

Test 1: While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a "protected" Bluetooth service (from the second list in the requirement) from a remote device that does not have the required level of trust to use the service. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.

Test 2: The evaluator shall repeat Test 1, allow the authorization, and verify that the remote device successfully accesses the service. (Note that this connection may involve pairing, if the untrusted remote device has not yet paired with the TOE.)

Test 3: If the TSF implementation differentiates between trusted and untrusted devices when determining if user authorization is required, repeat Test 1with a service that appears in the second list in the requirement (but not in the first list) and a device that has the required level of trust to use the service. The evaluator shall verify that the user is not prompted for explicit authorization and the connection to the service succeeds.

Test 4: If the TSF implementation differentiates between trusted and untrusted devices when determining if user authorization is required, repeat Test 1 with a service that appears in the first list in the requirement and a device that has the required level of trust to use the service. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the

authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.

Test 5: If the TSF implementation differentiates between trusted and untrusted devices when determining if user authorization is required, repeat Test 2 with a service that appears in the first list in the requirement and a device that has the required level of trust to use the service. The evaluator shall verify that the remote device successfully accesses the service if the user explicitly provides authorization.

D.4.1.2Bluetooth Authentication

FIA_BLT_EXT.2	Extended: Bluetooth Authentication

FIA_BLT_EXT.2.1 The TSF shall require Bluetooth mutual authentication between devices prior to any data transfer over the Bluetooth link.

Application Note: If devices are not already paired, the pairing process must be initiated. If the devices are already paired, mutual authentication based on the current link key must succeed before any data passes over the link.

Assurance Activity:

The evaluator shall ensure that the TSS describes how data transfer of any type is prevented before the Bluetooth pairing is completed. The TSS shall specifically call out any supported RFCOMM and L2CAP data transfer mechanisms. The evaluator shall ensure that the data transfers are only completed after the Bluetooth devices are paired and mutually authenticated

The evaluator shall perform the following test:

Test 1: The evaluator shall use a Bluetooth tool to attempt to access TOE files using the OBEX Object Push service and verify that pairing and mutual authentication are required by the TOE before allowing access. (If the OBEX Object Push service is unsupported on the TOE, a different service that transfers data over Bluetooth L2CAP and/or RFCOMM may be used in this test.)

FIA_BLT_EXT.2.2 The TSF shall discard connection attempts from a Bluetooth device address (BD_ADDR) to which a current connection already exists.

Application Note: If the TOE already has a connection to a remote Bluetooth device, a new connection attempt from a device claiming the same Bluetooth device address may be malicious and should be rejected/ignored. Only one connection to a single remote BD_ADDR may be supported at a time.

This requirement will be moved to Section 5 and will be mandatory for products entering into evaluation after Quarter 3, 2015.

Assurance Activity:

The evaluator shall ensure that the TSS describes how Bluetooth connections are maintained such that two devices with the same Bluetooth device address are not simultaneously

connected and such that the initial connection is not superseded by any following connection attempts.

The evaluator shall perform the following test:

Test 1: The evaluator shall perform the following steps:

Step 1 - Make a Bluetooth connection between the TOE and a remote Bluetooth device with address a known address (BD_ADDR1).

Step 2 - Attempt a connection to the same TOE from a second remote Bluetooth device claiming to have a Bluetooth device address matching BD_ADDR1.

Step 3 - Using a Bluetooth protocol analyzer, verify that the second connection attempt is ignored by the TOE and that the initial connection to the device with BR_ADDR1 is unaffected.

D.4.2 X509 Certificates (FIA_X509)

D.4.2.1 X509 Certificate Authentication

FIA_X509_EXT.2 Extended: X509 certificate authentication

FIA_X509_EXT.2.3 The TSF shall generate a Certificate Request Message as specified in RFC 2986 and be able to provide the following information in the request: public key and [selection: device-specific information, Common Name, Organization, Organizational Unit, Country].

Application Note: The public key referenced in FIA_X509_EXT.2.3 is the public key portion of the public-private key pair generated by the TOE as specified in FCS_CKM.1(1). The trusted channel requirements do not apply to communication with the CA for the certificate request/response messages.

As Enrollment over Secure Transport (EST) is a new standard that has not yet been widely adopted, this requirement is included as an interim objective requirement in order to allow developers to distinguish those products which have do have the ability to generate Certificate Request Messages but do not yet implement EST.

FIA_X509_EXT.2.4 The TSF shall validate the chain of certificates from the Root CA upon receiving the CA Certificate Response.

Assurance Activity:

If the ST author selects "device-specific information", the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

The evaluator shall check to ensure that the operational guidance contains instructions on generating a Certificate Request Message. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this

guidance includes instructions for establishing these fields before creating the certificate request message.

The evaluator shall also perform the following tests:

Test 1: The evaluator shall use the operational guidance to cause the TOE to generate a certificate request message. The evaluator shall capture the generated message and ensure that it conforms with the format specified. The evaluator shall confirm that the certificate request provides the public key and other required information, including any necessary user-input information.

Test 2: The evaluator shall demonstrate that validating a certificate response message without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate response message, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.

D.4.2.2 X509 Certificate Enrollment

FIA_X509_EXT.4 Extended: X509 certificate enrollment

FIA_X509_EXT.4.1 The TSF shall use the Enrollment over Secure Transport (EST) protocol as specified in RFC 7030 to request certificate enrollment using the simple enrollment method described in RFC 7030 Section 4.2.

FIA_X509_EXT.4.2 The TSF shall be capable of authenticating EST requests using an existing certificate and corresponding private key as specified by RFC 7030 Section 3.3.2.

FIA_X509_EXT.4.3 The TSF shall be capable of authenticating EST requests using HTTP Basic Authentication with a username and password as specified by RFC 7030 Section 3.2.3.

FIA_X509_EXT.4.4 The TSF shall perform authentication of the EST server using an Explicit Trust Anchor following the rules described in RFC 7030, section 3.6.1.

Application Note: EST also uses HTTPS as specified in FCS_HTTPS_EXT.1 to establish a secure connection to an EST server. The separate Trust Anchor Database dedicated to EST operations is described as Explicit Trust Anchors in RFC 7030.

FIA_X509_EXT.4.5 The TSF shall be capable of requesting server-provided private keys as specified in RFC 7030 Section 4.4.

FIA_X509_EXT.4.6 The TSF shall be capable of updating its EST-specific Trust Anchor Database using the "Root CA Key Update" process described in RFC 7030 Section 4.1.3.

FIA_X509_EXT.4.7 The TSF shall generate a Certificate Request Message for EST as specified in RFC 2986 and be able to provide the following information in the request: public key and [selection: device-specific information, Common Name, Organization, Organizational Unit, Country].

FIA_X509_EXT.4.8 The TSF shall validate the chain of certificates from the Root CA certificate in the Trust Anchor Database to the EST Server CA certificate upon receiving a CA Certificates Response.

Application Note: The public key referenced in FIA_X509_EXT.4.7 is the public key portion of the public-private key pair generated by the TOE as specified in FCS_CKM.1(1).

Assurance Activity:

The evaluator shall check to ensure that the operational guidance contains instructions on requesting certificates from an EST server, including generating a Certificate Request Message.

The evaluator shall also perform the following tests. Other tests are performed in conjunction with the Assurance Activity listed for FCS_TLSC_EXT.2.

Test 1: The evaluator shall use the operational guidance to cause the TOE to request certificate enrollment from an EST server using the simple enrollment method described in RFC 7030 Section 4.2, authenticating the certificate request to the server using an existing certificate and private key as described by RFC 7030 Section 3.3.2. The evaluator shall confirm that the resulting certificate is successfully obtained and installed in the TOE key store.

Test 2: The evaluator shall use the operational guidance to cause the TOE to request certificate enrollment from an EST server using the simple enrollment method described in RFC 7030 Section 4.2, authenticating the certificate request to the server using a username and password as described by RFC 7030 Section 3.2.3. The evaluator shall confirm that the resulting certificate is successfully obtained and installed in the TOE key store.

Test 3: The evaluator shall modify the EST server to return a certificate containing a different public key than the key included in the TOE's certificate request. The evaluator shall use the operational guidance to cause the TOE to request certificate enrollment from an EST server. The evaluator shall confirm that the TOE does not accept the resulting certificate since the public key in the issued certificate does not match the public key in the certificate request.

Test 4: The evaluator shall configure the EST server or use a man-in-the-middle tool to present a server certificate to the TOE that is present in the TOE general Trust Anchor Database but not its EST-specific Trust Anchor Database. The evaluator shall cause the TOE to request certificate enrollment from the EST server. The evaluator shall verify that the request is not successful.

Test 5: The evaluator shall configure the EST server or use a man-in-the-middle tool to present an invalid certificate. The evaluator shall cause the TOE to request certificate enrollment from the EST server. The evaluator shall verify that the request is not successful The evaluator shall configure the EST server or use a man-in-the-middle tool to present a certificate that does not have the CMC RA purpose and verify that requests to the EST server fail. The tester shall repeat the test using a valid certificate and a certificate that contains the CMC RA purpose and verify that the certificate enrolment requests succeed.

Test 6: The evaluator shall use a packet sniffing tool between the TOE and an EST server. The evaluator shall turn on the sniffing tool and cause the TOE to request certificate enrollment from an EST server. The evaluator shall verify that the EST protocol interaction occurs over a Transport Layer Security (TLS) protected connection. The evaluator is not expected to decrypt the connection but rather observe that the packets conform to the TLS protocol format.

Test 7: The evaluator shall use the operational guidance to cause the TOE to request a server-provided private key and certificate from an EST server. The evaluator shall confirm that the resulting private key and certificate are successfully obtained and installed in the TOE key store.

Test 8: The evaluator shall modify the EST server to, in response to a server-provided private key and certificate request, return a private key that does not correspond with the public key in the returned certificate. The evaluator shall use the operational guidance to cause the TOE to request a server-provided private key and certificate. The evaluator shall confirm that the TOE does not accept the resulting private key and certificate since the private key and public key do not correspond.

Test 9: The evaluator shall configure the EST server to provide a "Root CA Key Update" as described in RFC 7030 Section 4.1.3. The evaluator shall cause the TOE to request CA certificates from the EST server and shall confirm that the EST-specific Trust Anchor Database is updated with the new trust anchor.

Test 10: The evaluator shall configure the EST server to provide a "Root CA Key Update" as described in RFC 7030 Section 4.1.3, but shall modify part of the NewWithOld certificate's generated signature. The evaluator shall cause the TOE to request CA certificates from the EST server and shall confirm that the EST-specific Trust Anchor Database is not updated with the new trust anchor since the signature did not verify.

Test 11: The evaluator shall use the operational guidance to cause the TOE to generate a certificate request message. The evaluator shall capture the generated message and ensure that it conforms with the format specified by RFC 2986. The evaluator shall confirm that the certificate request provides the public key and other required information, including any necessary user-input information.

D.5 Class: Protection of the TSF (FPT)

D.5.1 Anti-Exploitation Services (FPT_AEX)

D.5.1.1Address-Space Layout Randomization

FPT_AEX_EXT.1	Extended: Anti-Exploitation Services (ASLR)

FPT_AEX_EXT.1.3 The TSF shall provide [***address space layout randomization (ASLR) to the kernel***].

FPT_AEX_EXT.1.4 The base address of any kernel-space memory mapping will consist of at least 4 unpredictable bits.

Application Notes: The 4 unpredictable bits may be provided by the TSF RBG (as specified in FCS_RBG_EXT.1).

Assurance Activity:

The evaluator shall ensure that the TSS section of the ST describes how the 4 bits are generated and provides a justification as to why those bits are unpredictable.

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

Test 1: The evaluator shall reboot the TOE at least five times. For each of these reboots, the evaluator shall examine memory mapping locations of the kernel. The evaluator must ensure that no memory mappings are placed in the same location on both devices.

D.5.1.2Memory Page Permissions

FPT_AEX_EXT.2 Extended: Anti-Exploitation Services (Memory Page Permissions)

FPT_AEX_EXT.2.2 The TSF shall prevent write and execute permissions from being simultaneously granted to any page of physical memory [selection: *with no exceptions, assignment: [specific exceptions]*].

Application Note: Memory used for just-in-time (JIT) compilation is anticipated as an exception in this requirement; if so, the ST author must address how this exception is permitted. It is expected that the memory management unit will transition the system to a non-operational state if any violation is detected in kernel memory space.

Assurance Activity:

The evaluator shall ensure that the TSS describes how the operating system of the application processor prevents all processes executing in a non-privileged execution domain from achieving write and execute permissions on any page of memory (with only specified exceptions). The evaluator shall ensure that the TSS describes how such processes are unable to request pages of memory with such permissions, and how they are unable to change permissions to both write and execute on any pages already allocated to them.

D.5.1.3Overflow Protection

FPT_AEX_EXT.3 Extended: Anti-Exploitation Services (Overflow Protection)

FPT_AEX_EXT.3.2 The TSF shall include heap-based buffer overflow protections in the runtime environment it provides to processes that execute on the application processor.

Application Note: These heap-based buffer overflow protections are expected to ensure the integrity of heap metadata such as memory addresses or offsets recorded by the heap implementation to manage memory blocks. This includes chunk headers, look-aside lists, and other data structures used to track the state and location of memory blocks managed by the heap.

Assurance Activity:

The evaluator shall verify that the TSS enumerates the heap implementations provided to userspace processes. The evaluator shall ensure that the TSS lists all types of heap metadata and identifies how the integrity of each type of metadata is ensured. The evaluator shall ensure that the TSS identifies all memory address or offset fields within each type of metadata and identifies how the integrity of these addresses or fields is ensured. The evaluator shall verify that the TSS identifies the manner in which an error condition is entered when a heap overflow is detected and the resulting actions taken by the TSF.

For each heap implementation, the evaluator shall write, or the developer shall provide access to, an application which allocates memory from the heap and then writes arbitrary data significantly beyond the end of the allocated buffer. The evaluator shall attempt to execute this application and verify that the write is not allowed.

D.5.2 Isolation of Baseband (FPT_BBD)

Mobile devices are becoming increasingly complex having an *application processor* that runs a rich operating system and user applications and separate *baseband processor(s)* that handle *cellular and other wireless* network connectivity.

- The application processor within most modern Mobile Devices is a *system on a chip* (SoC) that integrates, for example, CPU/GPU cores and memory interface electronics into a single, power-efficient package.
- Baseband processors are becoming increasingly complex themselves delivering voice encoding alongside *multiple* independent radios (LTE, WiFi, Bluetooth, FM, GPS) in a single package containing multiple CPUs and DSPs.

Thus, the baseband processor(s) in these requirements include such integrated SoCs and include any radio processors (integrated or not) on the Mobile Device.

All other requirements mostly, except where noted, apply to firmware/software on the application processor, but future requirements (notably, all Integrity, Access Control, and Anti-Exploitation requirements) will apply to application processors and baseband processors.

FPT_BBD_EXT.1 Application Processor Mediation

FPT_BBD_EXT.1.1 The TSF shall prevent code executing on any baseband processor (BP) from accessing application processor (AP) resources except when mediated by the AP.

Application Note: These resources include:

- Volatile and non-volatile memory
- Control of and data from integrated and non-integrated peripherals (e.g. USB controllers, touch screen controllers, LCD controller, codecs)
- Control of and data from integrated and non-integrated I/O sensors (e.g. camera, light, microphone, GPS, accelerometers, geomagnetic field sensors)

Assurance Activity:

The evaluator shall ensure that the TSS section of the ST describes at a high level how the processors on the Mobile Device interact, including which bus protocols they use to communicate, any other devices operating on that bus (peripherals and sensors), and identification of any shared resources. The evaluator shall verify that the design described in the TSS does not permit any BPs from accessing any of the peripherals and sensors or from accessing main memory (volatile and non-volatile) used by the AP. In particular, the evaluator shall ensure that the design prevents modification of executable memory of the AP by the BP.

D.5.3 Bluetooth Profile Limiting (FPT_BLT)

FPT_BLT_EXT.1	**Extended: Limitation of Bluetooth Profile Support**

FPT_BLT_EXT.1.1 The TSF shall disable support for [assignment: *list of Bluetooth profiles*] Bluetooth profiles when they are not currently being used by an application on the Mobile Device, and shall require explicit user action to enable them.

Application Note: Some Bluetooth services incur more serious consequences if unauthorized remote devices gain access to them. Such services should be protected by measures like disabling support for the associated Bluetooth profile unless it is actively being used by an application on the Mobile Device (in order to prevent discovery by a Service Discovery Protocol search), and then requiring explicit user action to enable those profiles in order to use the services. It may be further appropriate to require additional user action before granting a remote device access to that service (FIA_BLT_EXT.1.2).

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

The ST author shall list all Bluetooth profiles which are disabled while not in use by an application and which need explicit user action in order to become enabled.

Assurance Activity:

The evaluator shall perform the following tests:

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

Test 2: The evaluator shall repeat Test 1 with a device that currently has a trusted device relationship with the TOE and verify that the same behavior is exhibited.

D.5.4 Self-Test Notification (FPT_NOT)

FPT_NOT_EXT.1	Extended: Self-Test Notification

FPT_NOT_EXT.1.2 The TSF shall [selection: *log, provide the administrator with*] TSF-software integrity verification values.

Application Note: These notifications are typically called remote attestation and these integrity values are typically called measurements. The integrity values are calculated from hashes of critical memory and values, including executable code. The ST author shall select whether these values are logged as a part of FAU_GEN.1.1 or are provided to the administrator.

Assurance Activity:

The evaluator shall verify that the TSS describes which critical memory is measured for these integrity values and how the measurement is performed (including which TOE software performs these generates these values, how that software accesses the critical memory, and which algorithms are used)

If the integrity values are provided to the administrator, the evaluator shall verify that the AGD guidance contains instructions for retrieving these values and information for interpreting them. (For example, if multiple measurements are taken, what those measurements are and how changes to those values relate to changes in the device state.)

Assurance Activity Note: The following test may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall repeat the following test for each measurement:

Test: The evaluator shall boot the device in an approved state and record the measurement taken (either from the log or by using the administrative guidance to retrieve the value via an MDM Agent). The evaluator shall modify the critical memory or value that is measured. The evaluator shall boot the device and verify that the measurement changed.

FPT_NOT_EXT.1.3 The TSF shall cryptographically sign all integrity verification values.

Application Note: The intent of this requirement is to provide assurance to the administrator that the responses provided are from the TOE and have not been modified or spoofed by a man-in-the-middle such as a network-based adversary or a malicious MDM Agent.

Assurance Activity:

The evaluator shall verify that the TSS describes which key the TSF uses to sign the responses to queries and the certificate used to prove ownership of the key. The evaluator shall perform the following test.

Test: The evaluator shall write, or the developer shall provide, a management application that queries either the audit logs or the measurements. The evaluator shall verify that the responses to these queries are signed and verify the signatures against the TOE's certificate.

D.5.5 Trusted Update (FPT_TUD)

FPT_TUD_EXT.2 Extended: Trusted Update Verification

FPT_TUD_EXT.2.5 The TSF shall by default only install mobile applications cryptographically verified by [selection: *a built-in X.509v3 certificate, a configured X.509v3 certificate*].

Application Note: The built-in certificate is installed by the manufacturer either at time of manufacture or as a part of system updates. The configured certificate used to verify the signature is set according to FMT_SMF_EXT.1 function 33.

Assurance Activity:

The evaluator shall verify that the TSS describes how mobile application software is verified at installation. The evaluator shall ensure that this method uses a digital signature by a code signing certificate.

Test 1: The evaluator shall write, or the developer shall provide access to, an application. The evaluator shall try to install this application without a digitally signature and shall verify that installation fails. The evaluator shall attempt to install an application digitally signed with an appropriate certificate, and verify that installation succeeds.

Test 2: The evaluator shall digitally sign the application with an invalid certificate and verify that application installation fails. The evaluator shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. This test may be performed in conjunction with the assurance activities for FIA_X509_EXT.1.

Test 3: If necessary, the evaluator shall configure the device to limit the public keys that can sign application software according to the AGD guidance. The evaluator shall digitally sign the application with a certificate disallowed by the device or configuration and verify that application installation fails. The evaluator shall attempt to install an application digitally signed with an authorized certificate and verify that application installation succeeds.

FPT_TUD_EXT.2.7 The TSF shall verify that software updates to the TSF are a current or later version than the current version of the TSF.

Application Note: A later version has a larger version number. The method for distinguishing newer software versions from older versions is determined by the manufacturer.

Assurance Activity:

The evaluator shall verify that the TSS describes the mechanism that prevents the TSF from installing software updates that are an older version that the currently installed version.

The evaluator shall repeat the following tests to cover all allowed software update mechanisms as described in the TSS. For example, if the update mechanism replaces an entire partition containing many separate code files, the evaluator does not need to repeat the test for each individual file.

Test 1: The evaluator shall attempt to install an earlier version of software (as determined by the manufacturer). The evaluator shall verify that this attempt fails by checking the version identifiers or cryptographic hashes of the privileged software against those previously recorded and checking that the values have not changed.

Test 2: The evaluator shall attempt to install a current or later version and shall verify that the update succeeds.

D.6 Class: TOE Access (FTA)

D.6.1 Default TOE Access Banners (FTA_TAB)

FTA_TAB.1	Default TOE Access Banners

FTA_TAB.1.1 Before establishing a user session, the TSF shall display an advisory warning message regarding unauthorized use of the TOE.

Application Note: This requirement may be met with the configuration of either text or an image containing the text of the desired message. The TSF shall minimally display this information at startup, but may also display the information at every unlock. The banner is configured according to FMT_SMF_EXT.1 function 36.

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.

E. Entropy Documentation And Assessment

The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS.

E.1 Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the random comes from, where it is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

If implemented, the design description shall include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on shall be included.

E.2 Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

The entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

E.3 Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

E.4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in

the entropy source.

F. Acronyms

F.1 Acronyms

Acronym	Meaning
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
AP	Application Processor
API	Application Programming Interface
ASLR	Address Space Layout Randomization
BP	Baseband Processor
BR/EDR	(Bluetooth) Basic Rate/Enhanced Data Rate
CA	Certificate Authority
CBC	Cipher Block Chaining
CCM	Counter with CBC-Message Authentication Code
CCMP	CCM Protocol
CMC	Certificate Management over Cryptographic Message Syntax (CMS)
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSP	Critical Security Parameters
DAR	Data At Rest
DEK	Data Encryption Key
DEP	Data Execution Prevention
DH	Diffie-Hellman
DNS	Domain Name System
DSA	Digital Signature Algorithm
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
EAPOL	EAP Over LAN
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory
EST	Enrollment over Secure Transport
FIPS	Federal Information Processing Standards
FM	Frequency Modulation
FQDN	Fully Qualified Domain Name
GCM	Galois Counter Mode
GPS	Global Positioning System
GPU	Graphics Processing Unit
GTK	Group Temporal Key
HDMI	High Definition Multimedia Interface
HMAC	Keyed-Hash Message Authentication Code
HTTPS	HyperText Transfer Protocol Secure
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPC	Inter-Process Communication
IPsec	Internet Protocol Security
KEK	Key Encryption Key
LE	(Bluetooth) Low Energy
LTE	Long Term Evolution
MD	Mobile Device
MDM	Mobile Device Management
MMI	Man-Machine Interface

Acronym	Meaning
MMS	Multimedia Messaging Service
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
NX	Never Execute
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OS	Operating System
OTA	Over the Air
PAE	Port Access Entity
PBKDF	Password-Based Key Derivation Function
PMK	Pairwise Master Key
PP	Protection Profile
PTK	Pairwise Temporal Key
RA	Registration Authority
RBG	Random Bit Generator
REK	Root Encryption Key
ROM	Read-only memory
RSA	Rivest Shamir Adleman Algorithm
SHA	Secure Hash Algorithm
SMS	Short Messaging Service
SPI	Security Parameter Index
SSH	Secure Shell
SSID	Service Set Identifier
ST	Security Target
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
URI	Uniform Resource Identifier
USB	Universal Serial Bus
USSD	Unstructured Supplementary Service Data
VPN	Virtual Private Network
WiFi	Wireless Fidelity
XCCDF	eXtensible Configuration Checklist Description Format
XTS	XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing

G. Use Case Templates

The following use case templates list those selections, assignments, and objective requirements that best support the use cases identified by this Protection Profile. Note that the templates assume that all SFRs listed in Section 5 are included in the ST, not just those listed in the templates. These templates and deviations from the template should be identified in the Security Target to assist customers with making risk-based purchasing decisions. Products that do not meet these templates are not precluded from use in the scenarios identified by this Protection Profile.

Several of the use cases templates include objective requirements that are strongly desired for the indicated use cases. Readers can expect those requirements to be made mandatory in the next revision of this protection profile, and industry should aim to include that security functionality in products in the near-term.

Where selections for a particular requirement are not identified in a use case template, all available selections are equally applicable to the use case.

G.1 [USE CASE 1] Enterprise-owned device for general-purpose enterprise use

Requirement	Action
FCS_STG_EXT.1.4	Do not select "the user."
FMT_MOF_EXT.1.2 Function 4	Assign GPS.
FMT_MOF_EXT.1.2 Function 23	Include in ST. Assign personal Hotspot connections.
FMT_MOF_EXT.1.2 Function 36	Include in ST.
FMT_MOF_EXT.1.2 Function 39	Include in ST. Select "USB Mass storage mode."
FMT_MOF_EXT.1.2 Function 41	Include in selection. Select "USB tethering."
FMT_SMF_EXT.1.1 Function 4	Assign GPS.
FMT_SMF_EXT.1.1 Function 23	Include in ST. Assign personal Hotspot connections.
FMT_SMF_EXT.1.1 Function 36	Include in ST.
FMT_SMF_EXT.1.1 Function 39	Include in ST. Select "USB Mass storage mode."
FMT_SMF_EXT.1.1 Function 41	Include in ST. Select both options.
FPT_BBD_EXT.1.1	Include in ST.
FPT_TST_EXT.2.1	Select "all executable code stored in mutable media."
FPT_TUD_EXT.2.5	Include in ST.
FTA_TAB.1.1	Include in ST.

Table 12: Enterprise-Owned Template

G.2 [USE CASE 2] Enterprise-owned device for specialized, high-security use

Requirement	Action
FCS_CKM.1.1(1)	Select ECC schemes.
FCS_CKM.2.1(1)	Select ECC schemes.
FCS_CKM_EXT.1.1	Select 256 bits.
FCS_CKM_EXT.2.1	Select 256 bits.
FCS_CKM_EXT.3.1	Select 256 bits.
FCS_COP.1.1(1)	Select 256 bits.

Requirement	Action
FCS_COP.1.1(2)	Select SHA-256 and SHA-384.
FCS_COP.1.1(3)	Selection ECDSA schemes.
FCS_COP.1.1(5)	Select 256 bits.
FCS_RBG_EXT.1.2	Select 256 bits.
FCS_TLSC_EXT.2.1	Select TLS_ECDHE_ECDSA_WITH AES_256_GCM_SHA384.
FCS_TLSC_EXT.2.5	Include in ST. Select secp384.
FCS_TLSC_EXT.2.6	Include in ST. Select SHA-384
FDP_DAR_EXT.1.2	Select 256 bits.
FIA_X509_EXT.2.2	Select either “allow the administrator to choose...” or “not accept the certificate”.
FMT_MOF_EXT.1.2 Function 3	Include in ST.
FMT_MOF_EXT.1.2 Function 6	Include in ST.
FMT_SMF_EXT.1.1 Function 4	Assign all radios on TSF.
FMT_SMF_EXT.1.1 Function 5	Assign all audio or visual collection devices on TSF.
FMT_SMF_EXT.1.1 Function 14	Assign all X.509v3 certificates in the Trust Anchor Database.
FMT_SMF_EXT.1.1 Function 23	Include in ST. Assign all protocols where the TSF acts as a server.
FMT_SMF_EXT.1.1 Function 36	Include in ST.
FPT_BBD_EXT.1	Include in ST.
FTA_TAB.1.1	Include in ST.

Table 13: High- Security Template

G.3 [USE CASE 3] Personally-owned device for personal and enterprise use

At this time no requirements or selections are recommended for this use case; however, it should be noted that FDP_ACF_EXT.1.2 and FMT_SMF_EXT.1 Functions 19 and 28 play a critical role in separating enterprise data from the user’s personal data in this use case.

G.4 [USE CASE 4] Personally-owned device for personal and limited enterprise use

At this time no requirements or selections are recommended for this use case.

H. Initialization Vector 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).

Table 14: References and IV Requirements for NIST-approved Cipher Modes