Microsoft Windows

Common Criteria Evaluation

Microsoft Windows 10 November 2015 Update

Security Target

Document Information	
Version Number	1.0
Updated On	June 10, 2016

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The example companies, organizations, products, people and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred.

Microsoft, Active Directory, Visual Basic, Visual Studio, Windows, the Windows logo, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

TABLE OF CONTENTS

LIST OF TABLES

1 Security Target Introduction

This section presents the following information required for a Common Criteria (CC) evaluation:

- Identifies the Security Target (ST) and the Target of Evaluation (TOE);
- Specifies the security target conventions and conformance claims; and,
- Describes the organization of the security target.

1.1 Security Target, TOE, and Common Criteria (CC) Identification

ST Title: Microsoft Windows 10 November 2015 Update Security Target

ST Version: version 1.0, June 10, 2016

TOE Software Identification: The following Windows Operating Systems (OS):

- Microsoft Windows 10 November 2015 Update Pro Edition (64-bit version)
- Microsoft Windows 10 November 2015 Update Enterprise Edition (64-bit version)

The following security updates and patches must be applied to the above Windows products:

- All critical updates as of December 31, 2015

TOE Hardware Identification: The following hardware platforms and components are included in the evaluated configuration:

- **Microsoft Surface Book**, Windows 10 Enterprise and Windows 10 Pro, 64-bit, Intel Core i7, Marvell 8897 Wi-Fi a/b/g/n adapter, Bluetooth 4.0, Bluetooth LE, Intel TPM 2.0

All devices include IEEE 802.11 Wi-Fi and Bluetooth 4.0.

TOE Guidance Identification: The following administrator, user, and configuration guides were evaluated as part of the TOE:

- *Microsoft Windows Common Criteria Evaluation Windows 10 Mobile Device Operational Guidance* along with all the documents referenced therein.

Evaluation Assurance: As specified in section 5.2.1 and specific Assurance Activities associated with the security functional requirements from section 5.2.2.

CC Identification: CC for Information Technology (IT) Security Evaluation, Version 3.1, Revision 4, September 2012.

1.2 CC Conformance Claims

This TOE and ST are consistent with the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 3.1, Revision 4, September 2012, extended (Part 2 extended)
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements Version 3.1, Revision 4 September 2012, extended with ALC_TSU_EXT.1 (Part 3 extended)
- Mobile Device Fundamentals Protection Profile, Version 2.0, September 17, 2014, (MDF PP)
- Evaluation Assurance Activities specified in Section 5.2.1 and CC Part 3 assurance requirements specified in section 5.2.2

1.3 Conventions, Terminology, Acronyms

This section specifies the formatting information used in the security target.

1.3.1 Conventions

The following conventions have been applied in this document:

- Security Functional Requirements (SFRs): Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
  - Iteration: allows a component to be used more than once with varying operations.
  - Assignment: allows the specification of an identified parameter.
  - Selection: allows the specification of one or more elements from a list.
  - Refinement: allows the addition of details.

  The conventions for the assignment, selection, refinement, and iteration operations are described in Section 5.

- Other sections of the security target use a bold font to highlight text of special interest, such as captions.

1.3.2 Terminology

The following terminology is used in the security target:

Term	Definition
Access	Interaction between an entity and an object that results in the flow or modification of data.
Access control	Security service that controls the use of resources[1] and the disclosure and modification of data[2].
Accountability	Tracing each activity in an IT system to the entity responsible for the activity.
Active Directory	Active Directory manages enterprise identities, credentials, information protection, system and application settings through AD Domain Services, Federation Services, Certificate Services and Lightweight Directory Services.

[1] Hardware and software
[2] Stored or communicated

Administrator	An authorized user who has been specifically granted the authority to manage some portion or the entire TOE and thus whose actions may affect the TOE Security Policy (TSP). Administrators may possess special privileges that provide capabilities to override portions of the TSP.
Assurance	A measure of confidence that the security features of an IT system are sufficient to enforce the IT system's security policy.
Attack	An intentional act attempting to violate the security policy of an IT system.
Authentication	A security measure that verifies a claimed identity.
Authentication data	The information used to verify a claimed identity.
Authorization	Permission, granted by an entity authorized to do so, to perform functions and access data.
Authorized user	An authenticated user who may, in accordance with the TOE Security Policy, perform an operation.
Availability	Timely[3], reliable access to IT resources.
Compromise	Violation of a security policy.
Confidentiality	A security policy pertaining to disclosure of data.
Critical cryptographic security parameters	Security-related information appearing in plaintext or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.
Cryptographic boundary	An explicitly defined contiguous perimeter that establishes the physical bounds (for hardware) or logical bounds (for software) of a cryptographic module.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines: • the transformation of plaintext data into ciphertext data • the transformation of ciphertext data into plaintext data • a digital signature computed from data • the verification of a digital signature computed from data • a data authentication code computed from data
Cryptographic module	The set of hardware, software, and/or firmware that implements approved security functions, including cryptographic algorithms and key generation, which is contained within the cryptographic boundary.
Cryptographic module security policy	A precise specification of the security rules under which a cryptographic module must operate.
Defense-in-depth	A security design strategy whereby layers of protection are utilized to establish an adequate security posture for an IT system.
Discretionary Access Control (DAC)	A means of restricting access to objects based on the identity of subjects and groups to which the objects belong. The controls are discretionary meaning that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.
Edition	A distinct variation of a Windows OS version. Examples of editions are Windows Server 2012 [Standard] and Windows Server 2012 Datacenter.
Enclave	A collection of entities under the control of a single authority and having a homogeneous security policy. They may be logical, or based on physical

[3] According to a defined metric

	location and proximity.
Entity	A subject, object, user or external IT device.
General-Purpose Operating System	A general-purpose operating system is designed to meet a variety of goals, including protection between users and applications, fast response time for interactive applications, high throughput for server applications, and high overall resource utilization.
Identity	A means of uniquely identifying an authorized user of the TOE.
Integrated Windows authentication	An authentication protocol formerly known as NTLM or Windows NT Challenge/Response.
Named object	• An object that exhibits all of the following characteristics: • The object may be used to transfer information between subjects of differing user identities within the TOE Security Function (TSF). • Subjects in the TOE must be able to request a specific instance of the object. • The name used to refer to a specific instance of the object must exist in a context that potentially allows subjects with different user identities to request the same instance of the object.
Object	An entity under the control of the TOE that contains or receives information and upon which subjects perform operations.
Operating environment	The total environment in which a TOE operates. It includes the physical facility and any physical, procedural, administrative and personnel controls.
Persistent storage	All types of data storage media that maintain data across system boots (e.g., hard disk, removable media).
Public object	An object for which the TSF unconditionally permits all entities “read” access under the Discretionary Access Control SFP. Only the TSF or authorized administrators may create, delete, or modify the public objects.
Resource	A fundamental element in an IT system (e.g., processing time, disk space, and memory) that may be used to create the abstractions of subjects and objects.
SChannel	A security package (SSP) that provides network authentication between clients and servers.
Secure Desktop	The secure desktop is a desktop object associated with a logon session for the SYSTEM account which securely collects the user’s credentials for authentication to Windows by allowing only the logon program, winlogon.exe, to run in this session.
Secure State	Condition in which all TOE security policies are enforced.
Security attributes	TSF data associated with subjects, objects and users that is used for the enforcement of the TSP.
Security-enforcing	A term used to indicate that the entity (e.g., module, interface, subsystem) is related to the enforcement of the TOE security policies.
Security-supporting	A term used to indicate that the entity (e.g., module, interface, subsystem) is not security-enforcing; however, the entity’s implementation must still preserve the security of the TSF.
Security context	The security attributes or rules that are currently in effect. For SSPI, a security context is an opaque data structure that contains security data relevant to a connection, such as a session key or an indication of the

	duration of the session.
Security package	The software implementation of a security protocol. Security packages are contained in security support provider libraries or security support provider/authentication package libraries.
Security principal	An entity recognized by the security system. Principals can include human users as well as autonomous processes.
Security Support Provider (SSP)	A dynamic-link library that implements the SSPI by making one or more security packages available to applications. Each security package provides mappings between an application's SSPI function calls and an actual security model's functions. Security packages support security protocols such as Kerberos authentication and Integrated Windows Authentication.
Security Support Provider Interface (SSPI)	A common interface between transport-level applications. SSPI allows a transport application to call one of several security providers to obtain an authenticated connection. These calls do not require extensive knowledge of the security protocol's details.
Security Target (ST)	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Subject	An active entity within the TOE Scope of Control (TSC) that causes operations to be performed. Subjects can come in two forms: trusted and untrusted. Trusted subjects are exempt from part or all of the TOE security policies. Untrusted subjects are bound by all TOE security policies.
Target of Evaluation (TOE)	An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
Threat	Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.
Unauthorized individual	A type of threat agent in which individuals who have not been granted access to the TOE attempt to gain access to information or functions provided by the TOE.
Unauthorized user	A type of threat agent in which individuals who are registered and have been explicitly granted access to the TOE may attempt to access information or functions that they are not permitted to access.
Universal Unique Identifier (UUID)	UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.
User	Any person who interacts with the TOE.
User Principal Name (UPN)	An identifier used by Microsoft Active Directory that provides a user name and the Internet domain with which that username is associated in an e-mail address format. The format is [*AD username*]@[associated *domain*]; an example would be *john.smith@microsoft.com.*
Uniform Resource Locator (URL)	The address that is used to locate a Web site. URLs are text strings that must conform to the guidelines in RFC 2396.
Version	A Version refers to a release level of the Windows operating system. Windows 7 and Windows 8 are different versions.
Vulnerability	A weakness that can be exploited to violate the TOE security policy.

Table 1 Definitions

1.3.3 Acronyms

The acronyms used in this security target are specified **in Appendix A: List of Abbreviations**

Appendix A: List of Abbreviations.

1.4 ST Overview and Organization

The Windows 10 TOE provides the following security services:

- Cryptographic support
- User data protection
- Identification and Authentication (I&A)
- Protection of the TOE Security Functions (TSF)
- TOE access/session control
- Trusted path/channel
- Security management
- Audit

This security target contains the following additional sections:

- TOE Description (Section 2): Provides an overview of the TSF and boundary.
- Security Problem Definition (Section 3): Describes the threats, organizational security policies and assumptions that pertain to the TOE.
- Security Objectives (Section 4): Identifies the security objectives that are satisfied by the TOE and the TOE operational environment.
- Security Requirements (Section 5): Presents the security functional and assurance requirements met by the TOE.
- TOE Summary Specification (TSS) (Section 6): Describes the security functions provided by the TOE to satisfy the security requirements and objectives.
- Protection Profile Conformance Claim (Section 7): Presents the rationale concerning compliance of the ST with the ***Mobile Device Fundamentals Protection Profile***.
- Rationale for Modifications to the Security Requirements (Section 8): Presents the rationale for the security objectives, requirements, and TOE Summary Specification as to their consistency, completeness and suitability.

2 TOE Description

The mobile device TOE includes the Microsoft Windows 10 operating system, the Microsoft Surface Book, and those applications necessary to manage, support and configure the mobile device.

Microsoft Windows 10 editions are preemptive multitasking, multiprocessor, and multi-user operating systems. In general, operating systems provide users with a convenient interface to manage underlying hardware. They control the allocation and manage computing resources such as processors, memory, and Input/Output (I/O) devices. Windows 10 also referred to as “Windows”, expand these basic operating system capabilities to controlling the allocation and managing higher level IT resources such as security principals (user or machine accounts), files, printing objects, services, window station,

desktops, cryptographic keys, network ports traffic, directory objects, and web content. Multi-user operating systems such as Windows keep track of which user is using which resource, grant resource requests, account for resource usage, and mediate conflicting requests from different programs and users.[4]

Windows 10 is suited for business desktops, notebook, convertible, and tablet computers. It is the workstation product and while it can be used by itself, it is designed to serve as a client within Windows domains.

2.1 Security Environment and TOE Boundary

The TOE includes both physical and logical boundaries. Its operational environment is that of a networked environment with IEEE 802.11 (Wi-Fi) and Bluetooth networks.

2.1.1 Logical Boundaries

The logical boundary of the TOE includes:

- The **Boot Manager**, which is invoked by the computer's bootstrapping code.
- The **Windows Loader** which loads the operating system into the computer's memory.
- **Windows OS Resume** which reloads an image of the executing operating system from a hibernation file as part of resuming from a hibernated state.
- The **Windows Kernel** which contains device drivers for the Windows NT File System, full volume encryption, the crash dump filter, and the kernel-mode cryptographic library.
- The **IPv4 / IPv6 network stack** in the kernel.
- **Windows Explorer** for Windows 10 which can be used to manage the OS and check the integrity of Windows files and updates.
- The **Windows Trusted Installer** which installs updates to the Windows operating system.
- The **Key Isolation Service** which protects secret and private keys.
- The **App Container** which is the execution environment for the Windows Store Applications which are the only applications covered by this evaluation.

2.1.2 Physical Boundaries

Physically, each TOE tablet or convertible consists of an Intel-based computer. Windows 10 can execute on Intel-based processors. Refer to section 1.1 for the specific list of hardware and processors included in the evaluation.

A set of devices may be attached as part of the TOE:

- Display Monitors
- Fixed Disk Drives (including disk drives and solid state drives)
- Removable Disk Drives (including USB storage)

[4] Some of the links in the security target was written for earlier Windows versions however the content in all those documents apply to Windows 10.

- Network Adaptor
- Keyboard
- Mouse
- Printer
- Audio Adaptor
- CD-ROM Drive
- Smart Card Reader
- Trusted Platform Module (TPM)

2.2 TOE Security Services

This section summarizes the security services provided by the TOE:

- **Security Audit:** Windows has the ability to collect audit data, review audit logs, protect audit logs from overflow, and restrict access to audit logs. Audit information generated by the system includes the date and time of the event, the user identity that caused the event to be generated, and other event specific data. Authorized administrators can review audit logs and have the ability to search and sort audit records. Authorized Administrators can also configure the audit system to include or exclude potentially auditable events to be audited based on a wide range of characteristics. In the context of this evaluation, the protection profile requirements cover generating audit events, selecting which events should be audited, and providing secure storage for audit event entries.
- **Cryptographic Support:** Windows provides FIPS CAVP-validated cryptographic functions that support encryption/decryption, cryptographic signatures, cryptographic hashing, cryptographic key agreement, and random number generation. The TOE additionally provides support for public keys, credential management and certificate validation functions and provides support for the National Security Agency's Suite B cryptographic algorithms. Windows also provides extensive auditing support of cryptographic operations and a key isolation service designed to limit the potential exposure of secret and private keys. In addition to using cryptography for its own security functions, Windows offers access to the cryptographic support functions for user-mode and kernel-mode programs. Public key certificates generated and used by Windows authenticate users and machines as well as protect both user and system data in transit.
    - **Software-based disk encryption:** Windows implements BitLocker to provide encrypted data storage for fixed and removable volumes and protects the disk volume's encryption key with one or more intermediate keys and authorization factor
    - **IPsec:** Windows implements IPsec to provide protected, authenticated, confidential, and tamper-proof networking between two peer computers.[5]
- **User Data Protection**: In the context of this evaluation Windows protects user data at rest and provides secure storage of X.509v3 certificates.

[5] Windows implements IPsec however it was not included in the Mobile Device Fundamentals PP evaluation because there is a separate protection profile for IPsec VPN clients.

- **Identification and Authentication**: In the context of this evaluation, Windows provides the ability to use, store, and protect X.509 certificates that are used for TLS and authenticates the user to their mobile device.
- **Protection of the TOE Security Functions**: Windows provides a number of features to ensure the protection of TOE security functions. Windows protects against unauthorized data disclosure and modification by using a suite of Internet standard. Windows ensures process isolation security for all processes through private virtual address spaces, execution context, and security context. The Windows data structures defining process address space, execution context, memory protection, and security context are stored in protected kernel-mode memory. Windows includes self-testing features that ensure the integrity of executable program images and its cryptographic functions. Finally, Windows provides a trusted update mechanism to update Windows binaries itself.
- **TOE Access:** Windows provides the ability for a user to lock their session either immediately or after a defined interval. Windows constantly monitors the mouse, keyboard, and touch display for activity and locks the computer after a set period of inactivity. Windows allows an authorized administrator to configure the system to display a logon banner before the logon dialog.
- **Trusted Path for Communications**: Windows uses a suite of protocols to provide a Virtual Private Network Connection (VPN) between itself, acting as a VPN client, and a VPN gateway in addition to providing protected communications for HTTPS and TLS.
- **Security Management:** Windows includes several functions to manage security policies. Policy management is controlled through a combination of access control, membership in administrator groups, and privileges.

The combination of these services enables Windows 10 to meets “Enterprise-owned device for general-purpose enterprise use” case and the “Enterprise-owned device for specialized, high-security use case” described in the MDF PP.

3 Security Problem Definition

The security problem definition consists of the threats to security, organizational security policies, and usage assumptions as they relate to Windows. The assumptions, threats, and policies are copied from the Mobile Device Fundamentals Protection Profile ("MDF PP").

3.1 Threats to Security

Table 2 presents known or presumed threats to protected resources that are addressed by Windows based on conformance to the Mobile Device Fundamentals Protection Profile.

Threat	Description
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 2 MDF PP Threats Addressed By Windows

3.2 Organizational Security Policies

An organizational security policy is a set of rules or procedures imposed by an organization upon its operations to protect its sensitive data and IT assets. Table 3 describes organizational security policies which are necessary for conformance to the protection profile.

Security Policy	Description
[None]	There are no Organizational Security Policies for the protection profile.

Table 3 Organizational Security Policies

3.3 Secure Usage Assumptions

Table 4 describes the core security aspects of the environment in which Windows is intended to be used. It includes information about the physical, personnel, procedural, and connectivity aspects of the environment.

The following specific conditions are assumed to exist in an environment where the TOE is employed in order to conform to the protection profile:

Assumption	Description
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 4 Secure Usage Assumptions

4 Security Objectives

This section defines the security objectives of Windows and its supporting environment. Security objectives, categorized as either TOE security objectives or objectives by the supporting environment, reflect the stated intent to counter identified threats, comply with any organizational security policies identified, or address identified assumptions. All of the identified threats, organizational policies, and assumptions are addressed under one of the categories below.

4.1 TOE Security Objectives

Table 5 describes the security objectives for Windows which are needed to comply with the MDF PP.

Security Objective	Source
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 5 Security Objectives for the TOE

4.2 Security Objectives for the Operational Environment

The TOE is assumed to be complete and self-contained and, as such, is not dependent upon any other products to perform properly. However, certain objectives with respect to the general operating environment must be met. Table 6 describes the security objectives for the operational environment as specified in the protection profile.

Environment Objective	Description
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 6 Security Objectives for the Operational Environment

5 Security Requirements

The section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) for the TOE. The requirements in this section have been drawn from the Mobile Device Fundamentals Protection Profile, Version 2.0, September 17, 2014, or the Common Criteria.

Where requirements are drawn from the protection profile, the requirements are copied verbatim, except for some changes to required identifiers to match the iteration convention of this document, from that protection profile and only operations performed in this security target are identified.

The extended requirements, extended component definitions and extended requirement conventions in this security target are drawn from the protection profile; the security target reuses the conventions from the protection profile which include the use of the word “Extended” and the “_EXT” identifier to denote extended functional requirements. The security target assumes that the protection profile correctly defines the extended components and so they are not reproduced in the security target.

Where applicable the following conventions are used to identify operations:

- **Iteration**: Iterated requirements (components and elements) are identified with letter following the base component identifier. For example, iterations of FMT_MOF.1 are identified in a manner similar to FMT_MOF.1(Audit) (for the component) and FCS_COP.1.1(Audit) (for the elements).
- **Assignment**: Assignments are identified in brackets and bold (e.g., **[assigned value]**).
- **Selection**: Selections are identified in brackets, bold, and italics (e.g., ***[selected value]***).
    - Assignments within selections are identified using the previous conventions, except that the assigned value would also be italicized and extra brackets would occur (e.g., ***[selected value [assigned value]]***).
- **Refinement**: Refinements are identified using bold text (e.g., **added text**) for additions and strike-through text (e.g., ~~deleted text~~) for deletions.

5.1 TOE Security Functional Requirements

This section specifies the SFRs for the TOE.

Requirement Class	Requirement Component
Security Audit (FAU)	Audit Data Generation (FAU_GEN.1)
	Audit Review (FAU_SAR.1)
	Security Audit Event Selection (FAU_SEL.1)
	Audit Storage Protection (FAU_STG.1)
	Prevention of Audit Data Loss (FAU_STG.4)
Cryptographic Support (FCS)	Cryptographic Key Generation for Key Establishment (FCS_CKM.1(ASYM KA))
	WLAN Cryptographic Key Generation for WLAN (FCS_CKM.1(WLAN384))
	WLAN Cryptographic Key Generation for WLAN (FCS_CKM.1(WLAN704))
	Cryptographic Key Establishment (FCS_CKM.2(ASYM AU))
	Cryptographic Key Establishment for Group Temporal Key (FCS_CKM.2(GTK))
	Extended: Cryptographic Key Support for Root Encryption Key

	(FCS_CKM_EXT.1)
	Extended: Cryptographic Key Random Generation for Data Encryption Keys (FCS_CKM_EXT.2)
	Extended: Cryptographic Key Generation for Key Encryption Keys (FCS_CKM_EXT.3)
	Extended: Key Destruction (FCS_CKM_EXT.4)
	Extended: TSF Wipe (FCS_CKM_EXT.5)
	Extended: Salt Generation (FCS_CKM_EXT.6)
	Extended: Bluetooth Key Generation (FCS_CKM_EXT.7)
	Cryptographic Operation for Data Encryption/Decryption (FCS_COP.1(SYM))
	Cryptographic Operation for Hashing (FCS_COP.1(HASH))
	Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))
	Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))
	Cryptographic Operation for Password-Based Key Derivation (FCS_COP.1(PBKD))
	Cryptographic Operation for Password Based Key Derivation (FCS_COP.1)
	Extended: Initialization Vector Generation (FCS_IV_EXT.1)
	Extended: Random Bit Generation (FCS_RBG_EXT.1)
	Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1)
	Extended: Cryptographic Key Storage (FCS_STG_EXT.1)
	Extended: Encrypted Cryptographic Key Storage (FCS_STG_EXT.2)
	Extended: Encrypted Integrity of Cryptographic Key Storage (FCS_STG_EXT.3)
	Extended: EAS TLS Protocol (FCS_TLSC_EXT.1)
	Extended: TLS Protocol (FCS_TLSC_EXT.2)
	Extended: HTTPS Protocol (FCS_HTTPS_EXT.1)
User Data Protection (FDP)	Extended: Security Attribute Based Access Control (FDP_ACF_EXT.1)
	Extended: Data at Rest Encryption (FDP_DAR_EXT.1)
	Extended: Subset Information Flow Control (FDP_IFC_EXT.1)
	Extended: User Data Storage (FDP_STG_EXT.1)
	Extended: Inter-TSF User Data Transfer Protection (FDP_UPC_EXT.1)
	Extended: Limitation of Bluetooth Device Access (FDP_BLT_EXT.1)
Identification & Authentication (FIA)	Authorization Failure Handling (FIA_AFL_EXT.1)
	Extended: Bluetooth User Authorization (FIA_BLT_EXT.1)
	Extended: Bluetooth Authentication (FIA_BLT_EXT.1)
	Extended: Bluetooth Authentication (FIA_BLT_EXT.2)
	Extended: Bluetooth Authentication (FIA_BLT_EXT.3)
	Extended: PAE Authentication (FIA_PAE_EXT.1)
	Extended: Password Management (FIA_PMG_EXT.1)
	Extended: Authentication Throttling (FIA_TRT_EXT.1)
	Protected Authorization Feedback (FIA_UAU.7)
	Extended: Authentication for Cryptographic Operation (FIA_UAU_EXT.1)
	Extended: Timing of Authentication (FIA_UAU_EXT.2)
	Extended: Re-Authentication (FIA_UAU_EXT.3)
	Extended: Validation of Certificates (FIA_X509_EXT.1)
	Extended: X.509 Certificate Authentication (FIA_X509_EXT.2)
	Extended: Request Validation of Certificates (FIA_X509_EXT.3)

Security Management (FMT)	Extended: Management of Security Functions Behavior (FMT_MOF_EXT.1)
	Extended: Specification of Management Functions (FMT_SMF_EXT.1)
	Extended: Specification of Remediation Actions (FMT_SMF_EXT.2)
Protection of the TSF (FPT)	Extended: Anti-Exploitation Services for Address Space Layout Randomization (FPT_AEX_EXT.1)
	Extended: Anti-Exploitation Services for Memory Page Permissions (FPT_AEX_EXT.2)
	Extended: Anti-Exploitation Services for Stack Overflow Protection (FPT_AEX_EXT.3)
	Extended: Domain Isolation (FPT_AEX_EXT.4)
	Extended: Application Processor Mediation (FPT_BBD_EXT.1)
	Extended: Limitation of Bluetooth Profile Support (FPT_BLT_EXT.1)
	Extended: Key Storage (FPT_KST_EXT.1)
	Extended: No Key Transmission (FPT_KST_EXT.2)
	Extended: No Plaintext Key Export (FPT_KST_EXT.3)
	Extended: Self-Test Event Notification (FPT_NOT_EXT.1(AUDIT))
	Extended: Self-Test Event Notification (FPT_NOT_EXT.1(ATTEST))
	Reliable Time Stamps (FPT_STM.1)
	Extended: TSF Cryptographic Functionality Testing (FPT_TST_EXT.1)
	Extended: TSF Integrity Testing (FPT_TST_EXT.2)
	Extended: Trusted Update: TSF Version Query (FPT_TUD_EXT.1)
	Extended: Trusted Update Verification (FPT_TUD_EXT.2)
TOE Access (FTA)	Extended: TSF- and User-initiated Locked State (FTA_SSL_EXT.1)
	Extended: Wireless Network Access (FTA_WSE_EXT.1)
	Default TOE Access Banners (FTA_TAB.1)
Trusted Path/Channels (FTP)	Extended: Trusted Channel Communication (FTP_ITC_EXT.1)

Table 7 TOE Security Functional Requirements

5.1.1 Security Audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)

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 8** ~~Table 10~~;
7. [***Audit records reaching an administrator-configurable percentage of audit capacity, [none***]].

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 8** ~~Table 10~~.

Requirement	Auditable Events	Additional 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	[***generation of a REK***]	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.	
FCS_CKM.1(3)	None.	
FCS_CKM.2(1)	None.	
FCS_CKM.2(2)	None.	
FCS_COP.1	None.	
FCS_HTTPS_EXT.1	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate. [***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. [***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_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_BLT_EXT.3	None.	
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 Factor.	No additional information.
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~~ ~~Success or failure of enrollment.~~ ~~Update of EST Trust Anchor Database~~	~~Issuer and Subject name of EST Server. Method of authentication. Issuer and Subject name of certificate used to authenticate. Content of~~

		~~Certificate Request Message Issuer and Subject name of added certificate or reason for failure.~~ ~~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(AUDIT)	[***Measurement of TSF software***].	[***Integrity verification value***].
FPT_NOT_EXT.1(ATTEST)	[***Measurement of TSF software***].	[***Integrity verification value***].
FPT_STM.1	None.	
FPT_TST_EXT.1	Initiation of self-test. Failure of self-test.	None
FPT_TST_EXT.2	Start-up of TOE.	Boot Mode.
	[***Detected integrity violations***].	[***The TSF code that caused the integrity violation***].
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.
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.

Table 8 Auditable Events

5.1.1.2 Audit Review (FAU_SAR.1)

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.

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

5.1.1.3 Selective Audit (FAU_SEL.1)

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) [**subject or user identity**].

5.1.1.4 Audit Storage Protection (FAU_STG.1)

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.

5.1.1.5 Prevention of Audit Data Loss (FAU_STG.4)

FAU_STG.4.1 The TSF shall overwrite the oldest stored audit records if the audit trail is full.

5.1.2 Cryptographic Support (FCS)

5.1.2.1 Cryptographic Key Generation (FCS_CKM.1(ASYM KA))

Application Note: FCS_CKM.1(ASYM KA) corresponds to FCS_CKM.1(1) in the MDF protection profile.

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

- [***RSA schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: [***
  - ***FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3;***
  - ***];***
- ***[ECC schemes] using ["NIST curves" P-256, P-384 and [P-521]] 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]

]

5.1.2.2 WLAN Cryptographic Key Generation (FCS_CKM.1(WLAN384))

Application Note: FCS_CKM.1(WLAN384) corresponds to FCS_CKM.1(2) in the MDF protection profile.

FCS_CKM.1.1(WLAN384) 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].

5.1.2.3 WLAN Cryptographic Key Generation (FCS_CKM.1(WLAN704))

Application Note: FCS_CKM.1(WLAN) corresponds to FCS_CKM.1(3) in the MDF protection profile.

FCS_CKM.1.1(WLAN704) The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [PRF-704] and specified cryptographic key sizes [256 bits] using a Random Bit Generator as specified in FCS_RBG_EXT.1 that meet the following: [IEEE 802.11ac-2013].

5.1.2.4 Cryptographic Key Establishment (FCS_CKM.2(ASYM AU))

Application Note: FCS_CKM.2(ASYM AU) corresponds to FCS_CKM.2(1) in the MDF protection profile.

FCS_CKM.2.1(ASYM AU) 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 [

- ***[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"];***

].

5.1.2.5 Cryptographic Key Establishment for Group Temporal Key (FCS_CKM.2(GTK))

Application Note: FCS_CKM.2(GTK) corresponds to FCS_CKM.2(2) in the MDF protection profile.

FCS_CKM.2.1(GTK) 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.

5.1.2.6 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)[6]

FCS_CKM_EXT.1.1 The TSF shall support a [***hardware-isolated***] REK with a [***symmetric***] key of strength [***256 bits***].

FCS_CKM_EXT.1.2 System software on the TSF shall be able only to request [***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.

5.1.2.7 Extended: Cryptographic Key Random Generation (FCS_CKM_EXT.2)

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

5.1.2.8 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)[7]

FCS_CKM_EXT.3.1 The TSF shall use [***asymmetric KEKs of [112 bits of security strength] security strength, [256-bit] symmetric KEKs***] corresponding to at least the security strength of the keys encrypted by the KEK.

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

[

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

c) generate the KEK using a key generation scheme that meets this profile (as specified in FCS_CKM.1(1))[8]

d) ***Combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by [using an XOR operation, encrypting one key with another]***

].

5.1.2.9 Extended: Key Destruction (FCS_CKM_EXT.4)[9]

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 [***consisting of zeroes***].
  - 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 that is not wear-leveled, the destruction shall be executed [***by a block erase that erases the reference to memory that stores data as well as the data itself***].

[6] This protection profile requirement was modified as part of NIAP Technical Decision 38.
[7] This protection profile requirement was modified as part of NIAP Technical Decision 38.
[8] FCS_CKM.1(ASYM KA) corresponds to FCS_CKM.1(1) in the MDF protection profile.
[9] This protection profile requirement was modified as part of NIAP Technical Decisions 28, 47, and 57.

- For non-volatile flash memory that is wear-leveled, the destruction shall be executed [***by a block erase***].
- 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.4.2 The TSF shall destroy all plaintext keying material and critical security parameters when no longer needed.

5.1.2.10 Extended: TSF Wipe (FCS_CKM_EXT.5)

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

- ***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 [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.

5.1.2.11 Extended: Salt Generation (FCS_CKM_EXT.6)

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

5.1.2.12 Extended: Bluetooth Key Generation (FCS_CKM_EXT.7)

FCS_CKM_EXT.7.1 The TSF shall randomly generate public/private ECDH key pairs every [**new pairing**].

5.1.2.13 Cryptographic Operation for Data Encryption/Decryption (FCS_COP.1(SYM))

Application Note: FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the MDF protection profile.

FCS_COP.1.1(SYM) 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

[

- ***AES Key Wrap (KW) (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,***

]

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

5.1.2.14 Cryptographic Operation for Hashing (FCS_COP.1(HASH))

Application Note: FCS_COP.1(HASH) corresponds to FCS_COP.1(2) in the MDF protection profile.

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

5.1.2.15 Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))

Application Note: FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the MDF protection profile.

FCS_COP.1.1(SIGN) 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 [

- ***[ECDSA schemes] using ["NIST curves" P-256, P-384 and [P-521]] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5];***

].

5.1.2.16 Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))

Application Note: FCS_COP.1(HMAC) corresponds to FCS_COP.1(4) in the MDF protection profile.

FCS_COP.1.1(HMAC) The TSF shall perform [keyed-hash message authentication] in accordance with a specified cryptographic algorithm HMAC-SHA-1 and [***HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512***] and cryptographic key sizes [**128 and 256 bits**] and message digest sizes 160 and [***256, 384, 512***] bits that meet the following: [FIPS Pub 198-1, "The Keyed-Hash Message Authentication Code, and FIPS Pub 180-4, "Secure Hash Standard].

5.1.2.17 Cryptographic Operation for Password-Based Key Derivation (FCS_COP.1(PBKD))

Application Note: FCS_COP.1(PBKD) corresponds to FCS_COP.1(5) in the MDF protection profile.

FCS_COP.1.1(PBKD) The TSF shall perform [Password-based Key Derivation Functions] in accordance with a specified cryptographic algorithm [HMAC-[***SHA-1, SHA-256, SHA-384, SHA-512***]], with [**8,800**] iterations, and output cryptographic key sizes [***128, 256***] that meet the following: [NIST SP 800-132]

5.1.2.18 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

FCS_IV_EXT.1.1 The TSF shall generate IVs in accordance with Table 14: References and IV Requirements for NIST-approved Cipher Modes.[10]

5.1.2.19 Extended: Random Bit Generation (FCS_RBG_EXT.1)

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

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates

[10] This refers to Table 14 of the MDF PP.

entropy from [***a software-based noise source, TSF-hardware-based noise source***] with a minimum of ***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.

5.1.2.20 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1)

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(**ASYM AU** ~~1~~)
- The following algorithms in FCS_COP.1(**SYM** ~~1~~): AES-CBC, [***AES Key Wrap, AES-GCM, AES-CCM***
- All mandatory and selected algorithms in FCS_COP.1(**SIGN** ~~3~~)
- All mandatory and selected algorithms in FCS_COP.1(**HASH** ~~2~~)
- All mandatory and selected algorithms in FCS_COP.1(**HMAC** ~~4~~)

[

- ***All mandatory and [selected algorithms] in FCS_CKM.1(ASYM KA ~~1~~),***
- ***The selected algorithms in FCS_COP.1(PBKD ~~5~~)***

].

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(**SYM** ~~1~~)
- Algorithms in FCS_COP.1(**SIGN** ~~3~~)

by keys stored in the secure key storage.

5.1.2.21 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

FCS_STG_EXT.1.1 The TSF shall provide [***hardware-isolated, software-based***] secure key storage for asymmetric private keys and [***symmetric keys, persistent secrets***].

FCS_STG_EXT.1.2 The TSF shall be capable of importing keys/secrets into the secure key storage upon request of [***the user, the administrator***] and [***applications running on the TSF***].

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

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 [***the administrator***].

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 [***the user, the administrator***].

5.1.2.22 Extended: Encrypted Cryptographic Key Storage (FCS_STG_EXT.2)[11]

FCS_STG_EXT.2.1 The TSF shall encrypt all DEKs and KEKs and [***long-term trusted channel key material, all software-based key storage***] by KEKs that are

[

1) Protected by the REK with [

[11] This protection profile requirement was modified as part of NIAP Technical Decision 38.

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

FCS_STG_EXT.2.2 DEKs and KEKs and [***long-term trusted channel key material, all software-based key storage***] shall be encrypted using one of the following methods: [***using a SP800-56B key establishment scheme, using AES in the [GCM, CCM, CBC mode***]].

5.1.2.23 Extended: Encrypted Integrity of Cryptographic Key Storage (FCS_STG_EXT.3)

FCS_STG_EXT.3.1 The TSF shall protect the integrity of any encrypted DEKs and KEKs and [***long-term trusted channel key material, all software-based key storage***] by [

- ***[GCM, CCM, Key Wrap] 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 [***hash, digital signature, MAC***] of the stored key prior to use of the key.

5.1.2.24 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1 The TSF shall implement TLS 1.0 and [***TLS 1.1 (RFC 4346), TLS 1.2 (RFC 5246)***] supporting the following ciphersuites:

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

].

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

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

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

FCS_TLSC_EXT.1.5 The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1, secp521r1***] and no other curves.

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: [***SHA256, SHA384, SHA512***] and no other hash algorithms.

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 [***renegotiation_info extension***] in the ClientHello message.

5.1.2.25 Extended: TLS Protocol (FCS_TLSC_EXT.2)

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
- [***Optional Ciphersuites:***
  - ***TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246***
  - ***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_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***

].

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

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

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

FCS_ TLSC_EXT.2.5 The TSF shall present the Supported Elliptic Curves Extension in the Client

Hello with the following NIST curves: [***secp256r1, secp384r1, secp521r1***] and no other curves.

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: [***SHA256, SHA384, SHA512***] and no other hash algorithms.

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 [***renegotiation_info extension***] in the ClientHello message.

5.1.2.26 Extended: HTTPS Protocol (FCS_HTTPS_EXT.1)

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 [***not establish the connection, request application authorization to establish the connection***] if the peer certificate is deemed invalid.

5.1.3 User Data Protection (FDP)

5.1.3.1 Extended: Security Access Control (FDP_ACF_EXT.1)

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

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

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.

5.1.3.2 Extended: Protected Data Encryption (FDP_DAR_EXT.1)

FDP_DAR_EXT.1.1 Encryption shall cover all protected data.

FDP_DAR_EXT.1.2 Encryption shall be performed using DEKs with AES in the [***CBC***] mode with key size [***256***] bits.

5.1.3.3 Extended: Subset Information Flow Control (FDP_IFC_EXT.1)

FDP_IFC_EXT.1.1 The TSF shall [***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***].

5.1.3.4 Extended: User Data Storage (FDP_STG_EXT.1)

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

5.1.3.5 Extended: Inter-TSF User Data Transfer Protection (FDP_UPC_EXT.1)

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 [***Bluetooth LE***] 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.

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

5.1.3.6 Extended: Limitation of Bluetooth Device Access (FDP_BLT_EXT.1)

FDP_BLT_EXT.1.1 The TSF shall limit the applications that may communicate with a particular paired Bluetooth device.

5.1.4 Identification and Authentication (FIA)

5.1.4.1 Authentication Failure Handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1 The TSF shall detect when a configurable positive integer within [**a range of acceptable values from 1 to 999**] of unsuccessful authentication attempts occur related to last successful authentication by that user.

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.

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

5.1.4.2 Extended: Bluetooth User Authorization (FIA_BLT_EXT.1)

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

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: [**all Bluetooth profiles**], and shall require explicit user authorization before granting untrusted remote devices access to services associated with the following Bluetooth profiles: [**all Bluetooth profiles**].

5.1.4.3 Extended: Bluetooth Authentication (FIA_BLT_EXT.2)[12]

FIA_BLT_EXT.2.1 The TSF shall require Bluetooth mutual authentication between devices prior to any data transfer over the Bluetooth link.

5.1.4.4 Extended: Bluetooth Authentication (FIA_BLT_EXT.3)[13]

FIA_BLT_EXT.3.1 The TSF shall discard connection attempts from a Bluetooth device address (BD_ADDR) to which a current connection already exists.

5.1.4.5 Extended: PAE Authentication (FIA_PAE_EXT.1)

FIA_PAE_EXT.1.1 The TSF shall conform to IEEE Standard 802.1X for a Port Access Entity (PAE) in the "Supplicant" role.

5.1.4.6 Extended: Password Management (FIA_PMG_EXT.1)

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 [***upper and lower case letters***], numbers, and special characters: [***"!", "@",***

[12] This protection profile requirement was modified as part of NIAP Technical Decision 30.
[13] This protection profile requirement was modified as part of NIAP Technical Decision 30.

"#", "$", "%", "^", "&", "*", "(", ")"];

2. Password length up to [**at least 14**] characters shall be supported.

5.1.4.7 Extended: Authentication Throttling (FIA_TRT_EXT.1)

FIA_TRT_EXT.1.1 The TSF shall limit automated user authentication attempts by [***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.

5.1.4.8 Protected Authentication Feedback (FIA_UAU.7)

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.

5.1.4.9 Extended: Authentication for Cryptographic Operation (FIA_UAU_EXT.1)

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 [***long-term trusted channel key material, all software-based key storage***] at startup.

5.1.4.10 Extended: Timing of Authentication (FIA_UAU_EXT.2)

FIA_UAU_EXT.2.1 The TSF shall allow [***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.

5.1.4.11 Extended: Re-Authentication (FIA_UAU_EXT.3)

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 [***no other conditions***].

5.1.4.12 Extended: Validation of Certificates (FIA_X509_EXT.1)

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 [***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 extendedeyUsage 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.

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.

5.1.4.13 Extended: X509 Certificate Authentication (FIA_X509_EXT.2)

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 [***TLS, HTTPS***]], and [***code signing for system software updates, code signing for mobile applications, code signing for integrity verification***, ***no additional uses***].[14]

FIA_X509_EXT.2.2 When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [***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, not accept the certificate***].

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 [***device-specific information, Common Name, Organization, Organizational Unit, Country***].

FIA_X509_EXT.2.4 The TSF shall validate the chain of certificates from the Root CA upon receiving the CA Certificate Response.

5.1.4.14 Extended: Request Validation of Certificates (FIA_X509_EXT.3)

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.

5.1.5 Security Management (FMT)

5.1.5.1 Extended: Management of Security Functions Behavior (FMT_MOF_EXT.1)

FMT_MOF_EXT.1.1 The TSF shall restrict the ability to perform the functions in column 3 of Table **9** ~~1~~ to the user.

FMT_MOF_EXT.1.2 The TSF shall restrict the ability to perform the functions in column 5 of Table **9** ~~1~~ to the administrator when the device is enrolled and according to the administrator-configured policy.

5.1.5.2 Extended: Specification of Management Functions (FMT_SMF_EXT.1)[15]

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

M: Mandatory

O: Optional / Objective

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2

[14] Windows implements IPsec however it was not included in the Mobile Device Fundamentals PP evaluation because there is a separate protection profile for IPsec VPN clients.

[15] This protection profile requirement was modified as part of NIAP Technical Decisions 44, 58, and 64.

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	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: a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures	M	-	M	M
3. enable/disable the VPN protection: a. across device [~~b. on a per-app basis~~ ***c. no other method***]	M	O	O	O
4. enable/disable [**Wi-Fi, Bluetooth,**]	M	O	O	O
5. enable/disable **camera, microphone**]: a. across device [~~b. on a per-app basis~~ ***c. no other method***]	M	O	O	O
6. specify wireless networks (SSIDs) to which the TSF may connect	M	-	M	O
7. configure security policy for each wireless network: a. [***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 [***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 [***[any***	M	O	O	-

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
other keys/secrets] in the secure key storage				
13. import X.509v3 certificates into the Trust Anchor Database	M	-	M	O
14. remove imported X.509v3 certificates and [***[all X.509v3 certificates]***] in the Trust Anchor Database	M	O	O	-
15. enroll the TOE in management	M	M	-	-
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 [~~c. allow/disallow additional wireless technologies to be used with Bluetooth,~~ ***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~~]	M	O	O	O
21. enable/disable display notification in the locked state of: [***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 [**USB hardware ports**]	O	O	O	O
23. enable/disable [**none**]	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

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
~~27. enable/disable bypass of local user authentication~~	~~O~~	~~O~~	~~O~~	~~O~~
28. wipe Enterprise data	O	O	O	-
~~29. approve [***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	**M**	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 [***certificate***] used to validate digital signature on applications	O	O	O	O
34. approve exceptions for shared use of keys/secrets by multiple applications	**M**	O	O	O
35. approve exceptions for destruction of keys/secrets by applications that did not import the key/secret	**M**	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 [***remote system***]	O	O	O	O
~~41. enable/disable [selection:~~ ~~a. Hotspot functionality authenticated by [selection: pre-shared key, passcode, no authentication],~~ ~~b. USB tethering authenticated by [selection: pre-shared key, passcode, no authentication]]~~	~~O~~	~~O~~	~~O~~	~~O~~
~~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~~	O	O	O	O

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
~~characteristics]~~				
~~44. enable/disable location services:~~ ~~a. across device~~ ~~[~~ ~~b. on a per-app basis~~ ~~c. no other method]~~	~~M~~	~~O~~	~~O~~	~~O~~
45. [**none**]	O	O	O	O

Table 9 Management Functions

5.1.5.3 Extended: Specification of Remediation Actions (FMT_SMF_EXT.2)

FMT_SMF_EXT.2.1 The TSF shall offer [***alert the administrator, remove Enterprise applications,***] upon unenrollment and [***no other triggers***].

5.1.6 Protection of the TSF (FPT)

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

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.

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.

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

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

FPT_AEX_EXT.2.2 The TSF shall prevent write and execute permissions from being simultaneously granted to any page of physical memory [***with no exceptions***].

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

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.

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.

5.1.6.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

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.

5.1.6.5 Extended: Application Processor Mediation (FPT_BBD_EXT.1)

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

5.1.6.6 Extended: Limitation of Bluetooth Profile Support (FPT_BLT_EXT.1)

FPT_BLT_EXT.1.1 The TSF shall disable support for [**all**] 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.

5.1.6.7 Extended: Key Storage (FPT_KST_EXT.1)

FPT_KST_EXT.1.1 The TSF shall not store any plaintext key material in readable nonvolatile memory.

5.1.6.8 Extended: No Key Transmission (FPT_KST_EXT.2)

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

5.1.6.9 Extended: No Plaintext Key Export (FPT_KST_EXT.3)

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

5.1.6.10 Extended: Self-Test Notification (FPT_NOT_EXT.1(AUDIT))

FPT_NOT_EXT.1.1(AUDIT) The TSF shall transition to non-operational mode and [***log failures in the audit record***] when the following types of failures occur:

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

5.1.6.11 Extended: Self-Test Notification (FPT_NOT_EXT.1(ATTEST))

FPT_NOT_EXT.1.1(ATTEST) The TSF shall transition to non-operational mode and [***log failures in the audit record, notify the administrator***] when the following types of failures occur:

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

FPT_NOT_EXT.1.2(ATTEST) The TSF shall [***log, provide the administrator with***] TSF-software integrity verification values.[16]

FPT_NOT_EXT.1.3(ATTEST) The TSF shall cryptographically sign all integrity verification values.[17]

5.1.6.12 Reliable Time Stamps (FPT_STM.1)

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

5.1.6.13 Extended: TSF Cryptographic Functionality Testing (FPT_TST_EXT.1)

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.

[16] This requirement applies only to devices with a TPM 2.0 and when the devices are enrolled as described in the deployment guidance.

[17] This requirement applies only to devices with a TPM 2.0 and when the devices are enrolled as described in the deployment guidance.

5.1.6.14 Extended: TSF Integrity Testing (FPT_TST_EXT.2)

FPT_TST_EXT.2.1 The TSF shall verify the integrity of the bootchain up through the Application Processor OS kernel, and [***all executable code stored in mutable media, operating system executable code and application executable code***], stored in mutable media prior to its execution through the use of [***a digital signature using a hardware-protected asymmetric key***].

FPT_TST_EXT.2.2 The TSF shall not execute code if the code signing certificate is deemed invalid.

5.1.6.15 Extended: Trusted Update: TSF Version Query (FPT_TUD_EXT.1)

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.

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

5.1.6.16 Extended: Trusted Update Verification (FPT_TUD_EXT.2)

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

FPT_TUD_EXT.2.2 The TSF shall [***update only by verified software***] the TSF boot integrity [***key***].

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

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

FPT_TUD_EXT.2.5 The TSF shall by default only install mobile applications cryptographically verified by [***a built-in X.509v3 certificate***].

FPT_TUD_EXT.2.6 The TSF shall not install code if the code signing certificate is deemed invalid.

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.

5.1.7 TOE Access (FTA)

5.1.7.1 Extended: TSF- and User-initiated Locked State (FTA_SSL_EXT.1)

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) [**Disabling any activity of the user's data access / TSF controlled display devices other than unlocking the session and displaying application status**].

5.1.7.2 Extended: Wireless Network Access (FTA_WSE_EXT.1)

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.

5.1.7.3 Default TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1 Before establishing a user session, the TSF shall display an advisory warning message regarding unauthorized use of the TOE. {optional requirement, Appendix D}

5.1.8 Trusted Path / Channels (FTP)

5.1.8.1 Extended: Trusted Channel Communication (FTP_ITC_EXT.1)

FTP_ITC_EXT.1.1 The TSF shall use 802.11-2012, 802.1X, and EAP-TLS and [***TLS, 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.

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 [***no other connections***].

5.2 TOE Security Assurance Requirements

5.2.1 CC Part 3 Assurance Requirements

The following table is the collection of CC Part 3 assurance requirements from the Mobile Device Fundamentals Protection Profile.

Requirement Class	Requirement Component
ASE: Security Target	ASE_INT.1: ST introduction
	ASE_CCL.1: Conformance claims
	ASE_OBJ.1: Security objectives
	ASE_ECD.1: Extended components definition
	ASE_REQ.1: Stated security requirements
	ASE_SPD.1: Security Problem Definition
	ASE_TSS.1: TOE summary specification
ADV: Design	ADV_FSP.1: Basic functional specification
AGD: Guidance Documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle Support	ALC_CMC.1: Labeling of the TOE
	ALC_CMS.1: TOE CM Coverage
	ALC_TSU_EXT.1: Timely Security Updates
ATE: Testing	ATE_IND.1: Independent testing - sample
AVA: Vulnerability Assessment	AVA_VAN.1: Vulnerability survey

Table 10 TOE Security Assurance Requirements

5.2.1.1 Timely Security Updates (ALC_TSU_EXT.1)

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).

5.2.2 Mobile Device Fundamentals PP Assurance Activities

This section copies the assurance activities from the protection profile in order to ease reading and comparisons between the protection profile and the security target.

5.2.2.1 Security Audit (FAU)

5.2.2.1.1 Audit Data Generation (FAU_GEN.1)

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.

5.2.2.1.2 Audit Review (FAU_SAR.1)

The assurance activity for this requirement is performed in conjunction with test 32 of FMT_SMF_EXT.1.

5.2.2.1.3 Selective Audit (FAU_SEL.1)

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.

5.2.2.1.4 Audit Storage Protection (FAU_STG.1)

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.

5.2.2.1.5 Prevention of Audit Data Loss (FAU_STG.4)

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.

5.2.2.2 Cryptographic Support (FCS)

5.2.2.2.1 Cryptographic Key Generation (FCS_CKM.1(ASYM KA))[18]

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

[18] FCS_CKM.1(ASYM KA) corresponds to FCS_CKM.1(1) in the MDF protection profile.

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.

5.2.2.2.2 WLAN Cryptographic Key Generation (FCS_CKM.1(WLAN384))[19]

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 4way 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.

5.2.2.2.3 WLAN Cryptographic Key Generation (FCS_CKM.1(WLAN704))[20]

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

[19] FCS_CKM.1(WLAN384) corresponds to FCS_CKM.1(2) in the MDF protection profile.

[20] FCS_CKM.1(WLAN) corresponds to FCS_CKM.1(3) in the MDF protection profile.

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.

5.2.2.2.4 Cryptographic Key Establishment (FCS_CKM.2.1(ASYM AU))[21]

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-

[21] FCS_CKM.2(ASYM AU) corresponds to FCS_CKM.2(1) in the MDF protection profile.

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 KTSOAEP 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

5.2.2.2.5 Cryptographic Key Establishment (FCS_CKM.2(GTK))[22]

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 4way 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 preshared 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.

5.2.2.2.6 Extended: Cryptographic Key Support (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1, FCS_CKM_EXT.1.2, FCS_CKM_EXT.1.3

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.

[22] FCS_CKM.2(GTK) corresponds to FCS_CKM.2(2) in the MDF protection profile.

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. This will likely necessitate 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.

FCS_CKM_EXT.1.4

The assurance activity for this element is performed in conjunction with the assurance activity for the other elements in this component.

5.2.2.2.7 Extended: Cryptographic Key Random Generation (FCS_CKM_EXT.2)

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.

5.2.2.2.8 Extended: Cryptographic Key Generation (FCS_CKM_EXT.3)

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 by an RBG, 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 greater than or equal to the key size and mode to be used for the encryption/decryption of the data.

- If the KEK is generated according to an asymmetric key scheme, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_CKM.1(1) is invoked. The evaluator uses the description of the key generation functionality in FCS_CKM.1(1) or documentation available for the operational environment to determine that the key strength being requested is greater than or equal to 112 bits.
- 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.)

5.2.2.2.9 Extended: Key Destruction (FCS_CKM_EXT.4)

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.

5.2.2.2.10 Extended: TSF Wipe (FCS_CKM_EXT.5)

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).

5.2.2.2.11 Extended: Salt Generation (FCS_CKM_EXT.6)

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.

5.2.2.2.12 Extended: Bluetooth Key Generation (FCS_CKM_EXT.7)

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.

5.2.2.2.13 Cryptographic Operation for Data Encryption/Decryption (FCS_COP.1(SYM))[23]

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

[23] FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the MDF protection profile.

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 3tuples. 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 216 bytes, an associated data length of 216 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 AESCCMP 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 nonzero 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 216 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 XTSAES 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

5.2.2.2.14 [24]Cryptographic Operation for Hashing (FCS_COP.1(HASH))

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.

[24] FCS_COP.1(HASH) corresponds to FCS_COP.1(2) in the MDF protection profile.

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function o 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.

5.2.2.2.15 Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))[25]

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

[25] FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the MDF protection profile.

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.

5.2.2.2.16 Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))[26]

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.

5.2.2.2.17 Cryptographic Operation for Password-Based Key Derivation (FCS_COP.1(PBKD))[27]

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.

[26] FCS_COP.1(HMAC) corresponds to FCS_COP.1(4) in the MDF protection profile.
[27] FCS_COP.1(PBKD) corresponds to FCS_COP.1(5) in the MDF protection profile.

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).

5.2.2.2.18 Extended: Initialization Vector Generation (FCS_IV_EXT.1)

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.

5.2.2.2.19 Extended: Random Bit Generation (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1, FCS_RBG_EXT.1.2, FCS_RBG_EXT.1.3

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.

FCS_RBG_EXT.1.4

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 behavior 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.

5.2.2.2.20 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1)

FCS_SRV_EXT.1.1

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.

FCS_SRV_EXT.1.2

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.

5.2.2.2.21 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

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.

5.2.2.2.22 Extended: Encrypted Cryptographic Key Storage (FCS_STG_EXT.2)

FCS_STG_EXT.2.1

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

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 methods.

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.

5.2.2.2.23 Extended: Encrypted Integrity of Cryptographic Key Storage (FCS_STG_EXT.3)

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.

5.2.2.2.24 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1

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 valid Server Finished message in plaintext and verify the client sends a fatal alert upon receipt and does not send any application data. The server's finished message shall contain valid verify_data and shall parse correctly using a network protocol analysis tool.

FCS_TLSC_EXT.1.2

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 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 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.

FCS_TLSC_EXT.1.5

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.

FCS_TLSC_EXT.1.6

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, FCS_TLSC_EXT.1.8

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.

5.2.2.2.25 Extended: TLS Protocol (FCS_TLSC_EXT.2) [28]

FCS_TLSC_EXT.2.1

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:

[28] The assurance activity for this requirement was modified as part of NIAP Technical Decision 34.

- 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 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 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 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.

FCS_TLSC_EXT.2.5

Testing for this element are performed in conjunction with the assurance activities for FPT_TST_EXT.2.1.

FCS_TLSC_EXT.2.6

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.276, FCS_TLSC_EXT.2.8

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.

5.2.2.2.26 Extended: HTTPS Protocol (FCS_HTTPS_EXT.1)

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.

5.2.2.3 User Data Protection (FDP)

5.2.2.3.1 Extended: Security Access Control (FDP_ACF_EXT.1)

FDP_ACF_EXT.1.1

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 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.

FDP_ACF_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 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.

5.2.2.3.2 Extended: Protected Data Encryption (FDP_DAR_EXT.1)

FDP_DAR_EXT.1.1

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.

5.2.2.3.3 Extended: Subset Information Flow Control (FDP_IFC_EXT.1)

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.

5.2.2.3.4 Extended: User Data Storage (FDP_STG_EXT.1)

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.

5.2.2.3.5 Extended: Inter-TSF User Data Transfer Protection (FDP_UPC_EXT.1)

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.

5.2.2.3.6 Extended: Limitation of Bluetooth Device Access (FDP_BLT_EXT.1)

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.

5.2.2.4 Identification and Authentication (FIA)

5.2.2.4.1 Authentication Failure Handling (FIA_AFL_EXT.1)

FIA_AFL_EXT.1.1, FIA_AFL_EXT.1.2

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.

FIA_AFL_EXT.1.3

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.

5.2.2.4.2 Extended: Bluetooth User Authorization (FIA_BLT_EXT.1.)

FIA_BLT_EXT.1.1

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 manin-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).

FIA_BLT_EXT.1.2

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.

5.2.2.4.3 Extended: Bluetooth Authentication (FIA_BLT_EXT.2)

FIA_BLT_EXT.2.1

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 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.

5.2.2.4.4 Extended: PAE Authentication (FIA_PAE_EXT.1)

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.

5.2.2.4.5 Extended: Password Management (FIA_PMG_EXT.1)

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.

5.2.2.4.6 Extended: Authentication Throttling (FIA_TRT_EXT.1)

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.

5.2.2.4.7 Protected Authentication Feedback (FIA_UAU.7)

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.

5.2.2.4.8 Extended: Authentication for Cryptographic Operation (FIA_UAU_EXT.1)

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.

5.2.2.4.9 Extended: Timing of Authentication (FIA_UAU_EXT.2)

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.

5.2.2.4.10 Extended: Re-Authentication (FIA_UAU_EXT.3)

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.

5.2.2.4.11 Extended: Validation of Certificates (FIA_X509_EXT.1)

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.)

5.2.2.4.12 Extended: X509 Certificate Authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1, FIA_X509_EXT.2.2

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.

FIA_X509_EXT.2.3, FIA_X509_EXT.2.4

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 userinput 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.

5.2.2.4.13 Extended: Request Validation of Certificates (FIA_X509_EXT.3)

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.

5.2.2.5 Security Management (FMT)

5.2.2.5.1 Extended: Management of Security Functions Behavior (FMT_MOF_EXT.1)

FMT_MOF_EXT.1.1

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 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.

5.2.2.5.2 Extended: Specification of Management Functions (FMT_SMF_EXT.1)

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., userinstalled 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 33 [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.

5.2.2.5.3 Extended: Specification of Remediation Actions (FMT_SMF_EXT.2)

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.

5.2.2.6 Protection of the TSF (FPT)

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

FPT_AEX_EXT.1.1, FPT_AEX_EXT.1.2

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.

FPT_AEX_EXT.1.3, FPT_AEX_EXT.1.4
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.

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

FPT_AEX_EXT.2.1
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.

FPT_AEX_EXT.2.2
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.

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

FPT_AEX_EXT.3.1
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.

FPT_AEX_EXT.3.2
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.

5.2.2.6.4 Extended: Domain Isolation (FPT_AEX_EXT.4)

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.

5.2.2.6.5 Extended: Application Processor Mediation (FPT_BBD_EXT.1)

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.

5.2.2.6.6 Extended: Limitation of Bluetooth Profile Support (FPT_BLT_EXT.1)

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.

5.2.2.6.7 Extended: Key Storage (FPT_KST_EXT.1)

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.

5.2.2.6.8 Extended: No Key Transmission (FPT_KST_EXT.2)

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.

5.2.2.6.9 Extended: No Plaintext Key Export (FPT_KST_EXT.3)

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.

5.2.2.6.10 Extended: Self-Test Notification (FPT_NOT_EXT.1)

FPT_NOT_EXT.1.1

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.

FPT_NOT_EXT.1.2

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 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

5.2.2.6.11 Reliable Time Stamps (FPT_STM.1)

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.

5.2.2.6.12 Extended: TSF Cryptographic Functionality Testing (FPT_TST_EXT.1)

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.

5.2.2.6.13 Extended: TSF Integrity Testing (FPT_TST_EXT.2)

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.

5.2.2.6.14 Extended: Trusted Update: TSF Version Query (FPT_TUD_EXT.1)

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.

5.2.2.6.15 Extended: Trusted Update Verification (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1, FPT_TUD_EXT.2.2, FPT_TUD_EXT.2.3

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 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.

FPT_TUD_EXT.2.5

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.2.6

Testing for this element are performed in conjunction with the assurance activities for FPT_TUD_EXT.2.3 and FPT_TUD_EXT.2.5.

FPT_TUD_EXT.2.7

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.

5.2.2.7 TOE Access (FTA)

5.2.2.7.1 Extended: TSF- and User-initiated Locked State (FTA_SSL_EXT.1)

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.

5.2.2.7.2 Extended: Wireless Network Access (FTA_WSE_EXT.1)

The assurance activity for this requirement is performed in conjunction with the assurance activity for FMT_SMF_EXT.1.

5.2.2.7.3 Default TOE Access Banners (FTA_TAB.1)

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.

5.2.2.8 Trusted Path / Channels (FTP)

5.2.2.8.1 Extended: Trusted Channel Communication (FTP_ITC_EXT.1)

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 TOE Summary Specification (TSS)

This chapter describes the Windows security functions which satisfy the security functional requirements of the protection profile. The TOE also includes additional relevant security functions which are also described in the following sections, as well as a mapping to the security functional requirements satisfied by the TOE.

6.1 Product Architecture

6.2 TOE Security Functions

This section presents the TOE Security Functions (TSFs) and a mapping of security functions to Security Functional Requirements (SFRs). The TOE performs the following security functions:

- Audit
- Cryptographic Support
- User Data Protection
- Identification and Authentication
- Security Management
- Protection of the TSF
- TOE Access
- Trusted Path / Channels

6.3 Audit

The TOE Audit security function performs:

- Audit Collection
- Selective Audit
- Audit Storage Protection
- Audit Log Overflow Protection
- Audit Log Restricted Access Protection
- Prevention of Audit Loss

6.3.1 Audit Collection

The Windows Event Log service creates the security event log, which contains security relevant audit records collected on a system, along with other event logs which are also registered by other audit entry providers. The Local Security Authority (LSA) server collects audit events from all other parts of the TSF and forwards them to the Windows Event Log service which will place the event into the log for the appropriate provider. While there is no size limit for a single audit record, the authorized administrator can specify a limit for the size of each event log. For each audit event, the Windows Event Log service stores the following data in each audit entry:

Field in Audit Entry	Description
Date	The date the event occurred.
Time	The time the event occurred.
User	The security identifier (SID) of that represents the user on whose behalf the event occurred that represents the user.
Event ID	A unique number within the audit category that identifies the specific audit event.
Source	The Windows component that generated the audit event.
Outcome	Indicates whether the security audit event recorded is the result of a successful or failed attempt to perform the action.
Category	The type of the event defined by the event source.

Table 11 Standard Fields in a Windows Audit Entry

The LSA service defines the following categories for audit events in the security log:

- System,
- Logon / Logoff
- Object Access
- Directory Service Access
- Privilege Use
- Detailed Process Tracking
- Policy Change

- Account Management
- Account Logon

Each audit entry may also contain category-specific data that is contained in the body of the entry as described below:

- For the System Category, the audit entry includes information relating to the system such as the time the audit trail was cleared, start or shutdown of the audit function, and startup and shutdown of Windows. Furthermore, the specific cryptographic operation is identified when such operations are audited.
- For the Logon and Account Logon Category, the audit entry includes the reason the attempted logon failed.
- For the Object Access and the Directory Service Access Category, the audit entry includes the object name and the desired access requested.
- For the Privilege Use Category, the audit entry identifies the privilege.
- For the Detailed Process Tracking Category, the audit event includes the process identifier.
- For the Policy Change and Account Management Category, the audit event includes the new values of the policy or account attributes.
- For the Account Logon Category, the audit event includes the logon type that indicates the source of the logon attempt as one of the following types in the audit record:
    - Interactive (local logon)
    - Network (logon from the network)
    - Service (logon as a service)
    - Batch (logon as a batch job)
    - Unlock (for Unlock screen saver)
    - Network_ClearText (for anonymous authentication to IIS)

There are two places within the TSF where security audit events are collected. Inside the kernel, the Security Reference Monitor (SRM), a part of the NT Executive, is responsible for generation of all audit entries for the object access, privilege use, and detailed process tracking event categories. Windows components can request the SRM to generate an audit record and supply all of the elements in the audit record except for the system time, which the Executive provides. With one exception, audit events for the other event categories are generated by various services that either co-exist in the LSA server or call, with the SeAuditPrivilege privilege, the Authz Report Audit interfaces implemented in the LSA Policy subcomponent. The exception is that the Event Log Service itself records an event record when the security log is cleared and when the security log exceeds the warning level configured by the authorized administrator.

The LSA server maintains an audit policy in its database that determines which categories of events are actually collected. Defining and modifying the audit policy is restricted to the authorized administrator. The authorized administrator can select events to be audited by selecting the category or categories to be audited. An authorized administrator can individually select each category. Those services in the security process determine the current audit policy via direct local function calls. The only other TSF component that uses the audit policy is the SRM in order to record object access, privilege use, and

detailed tracking audit. LSA and the SRM share a private local connection port, which is used to pass the audit policy to the SRM. When an authorized administrator changes the audit policy, the LSA updates its database and notifies the SRM. The SRM receives a control flag indicating if auditing is enabled and a data structure indicating that the events in particular categories to audit.

In addition to the system-wide audit policy configuration, it is possible to define a per-user audit policy using auditpol.exe. This allows individual audit categories (of success or failure) to be enabled or disabled on a per user basis.[29] The per-user audit policy refines the system-wide audit policy with a more precise definition of the audit policy for which events will be audited for a specific user.

Within each category, auditing can be performed based on success, failure, or both. For object access events, auditing can be further controlled based on user/group identify and access rights using System Access Control Lists (SACLs). SACLs are associated with objects and indicate whether or not auditing for a specific object, or object attribute, is enabled.

The TSF is capable of generating the audit events associated with each audit category, as described in the Description column of **Table 12 Audit Event Categories**. The auditable events associated with each category capture the events listed in section **5.1.1.1**. For each category, the associated audit events (listed in **5.1.1.1**) for each of the requirements in the FAU_GEN Required Events column of **Table 12** are captured.

Category	Description	FAU_GEN Required Events
System	Audit attempts that affect security of the entire system such as clearing the audit trail.	FAU_STG.3, FCS_CKM.1*, FCS_CKM.4, FCS_COP.1*, FCS_RBG_EXT.1, FPT_STM.1, FPT_NOT_EXT.1,[30] FPT_TST_EXT.1, FPT_TST_EXT.2
Object Access	Audit attempts to access user objects, such as files.	None for the MDF PP.
Privilege Use	Audits attempts to use security relevant privileges. Security relevant privileges are those privileges that are related to the TSFs and can be assigned in the evaluated configuration.	None for the MDF PP.
Detailed Process Tracking	Audit subject-tracking events, including program activation, handle duplication, indirect access to an object, and process exit.	None for the MDF PP.
Policy Change	Audit attempts to change security policy settings such as the audit	FAU_SEL.1

[29] Windows will prevent a local administrator from disabling auditing for local administrator accounts. If an administrator can bypass auditing, they can avoid accountability for such actions as exfiltrating files without authorization.

[30] For cryptographic self-test and code integrity failures, not MDM health attestation data.

	policy and privilege assignment.	
Account Management	Audit attempts to create, delete, or change user or group accounts and changes to their attributes.	FIA_AFL_EXT.1
Directory Service Access	Audit access to directory service objects and associated properties.	None for the MDF PP.
Logon	Audit attempts to logon or logoff the system, attempts to make a network connection.	FIA_AFL_EXT.1, FIA_UAU_EXT.1, FIA_UID.1, FTA_SSL_EXT.1
Account Logon	Audit when a DC receives a logon request.	FIA_UAU.1(Logon), FIA_UID.1

Table 12 Audit Event Categories

6.3.2 Selective Audit

The authorized administrator has the ability to select events to be audited based upon object identity, user identity, computer (host identity), type (category), and outcome (success or failure) of the event. Selecting the set of events that will be audited can be on a per-machine basis by using tools such as auditpol.exe and wevtutil.exe, or using group policies to audit sets of machines (i.e. auditing based on the host identity).

6.3.3 Audit Log Overflow Protection

The TSF protects against the loss of events through a combination of controls associated with audit queuing and event logging. As configured in the TOE, audit data is appended to the audit log until it is full. The TOE protects against lost audit data by allowing the authorized administrator to configure the system to generate an audit event when the security audit log reaches a specified capacity percentage (e.g., 90%). Additionally, the authorized administrator can configure the system not to overwrite events – overwriting the oldest stored audit records if the audit trail is full – and instead will shut down when the security audit log is full. When so configured, after the system shutdowns due to audit overflow, only the authorized administrator can restart the system to log on and manage the security log. When the security log is full, a message is written to the display indicating the audit log has overflowed.

As described above, the TSF collects security audit data in two ways, via the SRM and via the LSA server. Both components maintain audit in-memory event queues. The SRM puts audit records on an internal queue to be sent to the LSA server. The LSA maintains a second queue where it holds the audit data from SRM and the other services in the security process. Both audit queues detect when an audit event loss has occurred. The SRM service maintains a high water mark and a low water mark on its audit queue to determine when full. The LSA also maintains marks in its queue to indicate when it is full.

Windows also provides an eventing infrastructure that other system components can use to log events which are not managed by the SRM or the LSA. The maximum size for these administrative and operational event logs can either be limited to the maximum size for the log file (and then prevent generation of new audit events for that particular log) or overwrite the oldest audit event in the log file when the log file reaches its maximum size. The Windows security target selects the second option.

The audit logs which contain events relevant to the Mobile Device evaluation are:

- Security

- System
- CAPI2
- CertificateServicesClient-Lifecycle-User
- CertificateServicesClient-Lifecycle-System
- Wcmsvc
- SystemSettings
- Device Configuration
- Microsoft-Windows-AppXDeployment-Server/Operational

6.3.4 Audit Log Restricted Access Protection

The Windows Event Log service controls and protects the security audit log. Note that the underlying files are configured so that only the TSF can open the files and the Event Log service opens those files exclusively when it starts and keeps them open while it is running. To view the contents of the security audit log, the user must be an authorized administrator. The security audit log is a system resource, created during system startup. No interfaces exist to create, destroy, or modify an event within the event log. The LSA subsystem is the only service registered to enter events into the security log. The TOE only offers user interfaces to read and clear the security event log. In order to read the event log, the user must have a read ACE in the access control list for the **Event Log** service.

6.3.5 SFR Mapping

The **Audit** function satisfies the following SFRs:

- **FAU_GEN.1**: The TOE audit collection is capable of generating audit events for items identified in section **5.1.1.1**. For each audit event the TSF records the date, time, user Security Identifier (SID) or name, logon type (for logon audit records), event ID, source, type, and category.
- **FAU_SAR**.1: The event viewer provides authorized administrators with the ability to review audit data in a readable format.
- **FAU_SEL.1**: The TSF provides the ability for the authorized administrator to select the events to be audited based upon object identity, user identity, workstation (host identity), event type, and success or failure of the event.
- **FAU_STG.1**: The interface to the logs are restricted to authorized administrators, including clearing the audit log, and does not allow for the modification of audit data within the audit log. The TOE can be configured such that when any event logs are full the system will overwrite the oldest events in each log type, based on a system-defined value which can be modified by the administrator. The operational logs listed above also restrict authorized administrators to only read-only access.
- **FAU_STG**.4: The TOE can be configured such that when any audit or administrative operational logs are full the system will overwrite the oldest events in each log type.[31]

[31] The TOE can be also configured such that when the security audit log is full the system shuts down so that only the authorized administrator can log on to the system to clear the security log and return the system to an operational state consistent with TOE guidance. Additionally, when the security log reaches a certain percentage, an audit event (alarm) is generated.

6.4 Cryptographic Support

6.4.1 Cryptographic Algorithms and Operations

Cryptography API: Next Generation (CNG) API is designed to be extensible at many levels and agnostic to cryptographic algorithm suites. An important feature of CNG is its native implementation of the Suite B algorithms, including algorithms for AES (128, 192, 256 key sizes)[32], the SHA-1 and SHA-2 family (SHA-256, SHA-384 and SHA-512) of hashing algorithms, elliptic curve Diffie Hellman (ECDH), and elliptical curve DSA (ECDSA) over the NIST-standard prime curves P-256, P-384, and P-521.

Protocols such as the Internet Key Exchange (IKE), and Transport Layer Security (TLS), make use of elliptic curve Diffie-Hellman (ECDH) included in Suite B as well as hashing functions.

Deterministic random bit generation (DRBG) is implemented in accordance with NIST Special Publication 800-90. Windows generates random bits by taking the output of a cascade of two SP800-90 AES-256 counter mode based DRBGs in kernel-mode and four cascaded SP800-90 AES-256 DRBGs in user-mode; programmatic callers can choose to obtain either 128 or 256 bits from the RBG which is seeded from the Windows entropy pool. The entropy pool is populated using the following values:

- An initial entropy value from a seed file provided to the Windows OS Loader at boot time (512 bits of entropy). [33]
- A calculated value based on the high-resolution CPU cycle counter which fires after every 1024 interrupts (a continuous source providing 16384 bits of entropy).
- Random values gathered periodically from the Trusted Platform Module (TPM), (320 bits of entropy on boot, 384 bits thereafter).
- Random values gathered periodically by calling the RDRAND CPU instruction, (256 bits of entropy).

The main source of entropy in the system is the CPU cycle counter which tracks hardware interrupts. This is a sufficient health test; if the computer were not accumulating hardware and software interrupts every processor clock cycle it would not be running and therefore there would be no need for random bit generation. In the same manner, a failure of the TPM chip or processor would be a critical error that halts the computer. In addition, when the user follows the CC administrative guidance, which includes operating Windows in the FIPS validated mode, it will run FIPS 140 AES-256 Counter Mode DBRG Known Answer Tests (instantiate, generate) on start-up. Windows always runs the SP 800-90-mandated self-tests for AES-CTR-DRBG during a reseed.[34]

Each entropy source is independent of the other sources and does not depend on time. The CPU cycle counter inputs vary by environmental conditions such as data received on a network interface card, key presses on a keyboard, mouse movement and clicks, and touch input.

[32] Note that the 192-bit key size is not used by Windows but is available to developers.
[33] The Windows OS Loader implements a SP 800-90 AES-CTR-DRBG and passes along 384 bits of entropy to the kernel for CNG to be use during initialization. This DBRG uses the same algorithms to obtain entropy from the CPU cycle counter, TPM, and RDRAND as described above.
[34] Running Windows in FIPS validated mode is required according to the administrative guidance.

The TSF defends against tampering of the random number generation (RNG) / pseudorandom number generation (PRNG) sources by encapsulating its use in Kernel Security Device Driver. The interface for the Windows random number generator is BCryptGenRandom.

The CNG provider for random number generation is the AES_CTR_DRBG, when Windows requires the use of a salt it uses the Windows RBG.

The encryption and decryption operations are performed by independent modules, known as Cryptographic Service Providers (CSPs). Windows generates symmetric keys (AES keys) using the FIPS Approved random number generator.

In addition to encryption and decryption services, the TSF provides other cryptographic operations such as hashing and digital signatures. Hashing is used by other FIPS Approved algorithms implemented in Windows (the hashed message authentication code, RSA, DSA, and EC DSA signature services, Diffie-Hellman and elliptic curve Diffie-Hellman key agreement, and random bit generation). When Windows needs to establish an RSA-based shared secret key it can act both as a sender or recipient, any decryption errors which occur during key establishment are presented to the user at a highly abstracted level, such as a failure to connect.

The hash-based message authentication code functions (HMAC) are based on SHA-1, SHA-256, SHA-384, and SHA-512, have the following characteristics:

HMAC Algorithm	Hash function Used	Block Size	Output MAC Length	Key Length / Key Size
HMAC-SHA-1	SHA-1	512 bits	20 bytes	The key size is 10-63 bytes when the key size is less than the block size and the key size is 65 to 1024 bytes when the key size is greater than the block size. The key size may also equal the block size. The key size is variable.
HMAC-SHA-256	SHA-256	512 bits	32 bytes	Same as HMAC-SHA-1
HMAC-SHA-384	SHA-384	1024 bits	48 bytes	The key size is 24-127 bytes when the key size is less than the block size and the key size is 129-1024 bytes when the key size is greater than the block size. The key size may also equal the block size. The key size is variable.
HMAC-SHA-512	SHA-512	1024 bits	64 bytes	The key size is 32-127 bytes when the key size is less than the block size and the key size is 129-1024 bytes when the key size is greater than the block size. The key size may also equal the block size. The key size is variable.

Table 13 HMAC Characteristics

The HMAC function forms the basis for a FIPS Approved implementation of a password based key derivation function (PBKDF). Windows inputs the password as a text string without any optional padding

or blocking into a HMAC 512 function. The hash functions supported by the Windows implementation of SP 800-132 are SHA-1, SHA-256, SHA-384 or SHA-512. The SHA-512 function is used by DPAPI (see **Protecting Data with DPAPI**).

Cryptographic Operation	Standard	Evaluation Method
Encryption/Decryption	FIPS 197 AES For CBC, CCM, XTS, KW, and GCM modes	NIST CAVP #3653, #3652, #3630, #3629
Digital signature	FIPS 186-4 RSA	NIST CAVP #1889, #1888, #1887, #1871
Digital signature	FIPS 186-4 DSA	NIST CAVP #1024
Digital signature	FIPS 186-4 ECDSA	NIST CAVP #760
Hashing	FIPS 180-3 SHA-2	NIST CAVP #3048, #3047
Keyed-Hash Message Authentication Code	FIPS 198-2 HMAC	NIST CAVP #2381
Random number generation	NIST SP 800-90 CTR_DRBG	NIST CAVP #955
Key agreement	NIST SP 800-56A ECDH	NIST CAVP #72
Key establishment	NIST SP 800-56B	NIST CVL #663
Key-based key derivation	SP800-108	NIST CAVP #72
IKEv1	SP800-135	NIST CVL #664
IKEv2	SP800-135	NIST CVL #664
TLS	SP800-135	NIST CVL #664

Table 14 Cryptographic Algorithm Standards and Evaluation Methods

The TSF includes a key isolation service designed specifically to host secret and private keys in a protected process to mitigate tampering or access to sensitive key materials. The TSF performs a key error detection check on each transfer of key (internal and intermediate transfers). The TSF prevents archiving of expired (private) signature keys. The TSF destroys non-persistent cryptographic keys by a single direct overwrite consisting of zeros. Persistent keys are stored in encrypted form, when it is necessary to delete a persistent key from flash memory this is done by deleting the storage.

Windows uses FIPS Approved algorithms to establish Wi-Fi sessions and can be configured to use ciphersuites that solely use FIPS Approved algorithm primitives.[35] The following table describes the keys and secrets used for networking; when these ephemeral keys or secrets are no longer needed for a network session, they are deleted as described above and in section **5.1.2.9**.

Key	Description
Symmetric encryption/decryption keys	Keys used for AES (FIPS 197) encryption/decryption for IPsec ESP, TLS, Wi-Fi.
HMAC keys	Keys used for HMAC-SHA1, HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512 (FIPS 198-1) as part of IPsec
Asymmetric ECDSA Public Keys	Keys used for the verification of ECDSA digital signatures (FIPS 186-4)

[35] Windows implements IPsec however it was not included in the Mobile Device Fundamentals PP evaluation because there is a separate protection profile for IPsec VPN clients.

	for IPsec traffic and peer authentication.
Asymmetric ECDSA Private Keys	Keys used for the calculation of ECDSA digital signatures (FIPS 186-4) for IPsec traffic and peer authentication.
Asymmetric RSA Public Keys	Keys used for the verification of RSA digital signatures (FIPS 186-4) for IPsec, TLS, Wi-Fi and signed product updates.
Asymmetric RSA Private Keys	Keys used for the calculation of RSA digital signatures (FIPS 186-4) for IPsec, TLS, and Wi-Fi as well as TPM-based health attestations. The key size can be 2048 or 3072 bits.
Asymmetric DSA Private Keys	Keys used for the calculation of DSA digital signatures (FIPS 186-4) for IPsec and TLS. The key size can be 2048 or 3072 bits.
DH Private and Public values	Private and public values used for Diffie-Hellman key establishment for TLS.
ECDH Private and Public values	Private and public values used for EC Diffie-Hellman key establishment for TLS.

Table 15 Types of Keys Used by Windows

6.4.2 Programming Interfaces

Universal Windows Applications can use these interfaces to obtain random bits from the OS:

- CryptographicBuffer.GenerateRandom
- CryptographicBuffer.GenerateRandomNumber

And can use these interfaces to obtain other cryptographic services from the OS:

- CryptographicEngine.Encrypt
- CryptographicEngine.Decrypt
- HashAlgorithmProvider.CreateHash
- HashAlgorithmProvider.HashData
- CryptographicEngine.Sign
- CryptographicEngine.VerifySignature
- KeyDerivationParameters.BuildForPbkdf2
- AsymmetricKeyAlgorithmProvider.CreateKeyPair
- CryptographicEngine.Sign
- CryptographicEngine.SignAsync
- CryptographicEngine.SignHashedData
- CryptographicEngine.SignHashedDataAsync
- CryptographicEngine.VerifySignature
- CryptographicEngine.VerifySignatureWithHashInput
- CryptographicEngine.Encrypt
- CryptographicEngine.Decrypt

6.4.3 Trusted Platform Module

Computers that incorporate a TPM have the ability to both create cryptographic keys within the TPM and protect data stored outside TPM so that the data can be decrypted only by the TPM internal keys. This process, often called "sealing" or "binding", can help protect the data from disclosure, but more importantly associates the key with the TPM. Each TPM contains a master "sealing" key, called the Storage Root Key (SRK), which was generated by the Storage Primary Seed (SPS). Like other cryptographic data within the TPM, the private portion of a key created in a TPM is never exposed to any other component, software, process, or user.

A TPM 2.0 protection profile written by the Trusted Computing Group provides additional detail about the SPS and the SRK: "The TPM holds the Storage Primary Seed (SPS) and generates Storage Root Keys (SRK) from SPS. The SRK are roots of Protected Storage Hierarchies associated with a TPM.[36] The storage keys in these hierarchies are used for symmetric encryption and signing of other keys and data together with their security attributes. The resulting encrypted file, which contains header information in addition to the data or the key, is called a BLOB (Binary Large Object) and is output by the TPM and can be loaded in the TPM when needed. The private keys generated on the TPM can be stored outside the TPM (encrypted) in a way that allows the TPM to use them later without ever exposing such keys in the clear outside the TPM. The TPM uses symmetric cryptographic algorithms to encrypt data and keys …."[37]

The TPM also provides protections that prevent the export of TPM keys and cryptographic data, such as the SPS and SRK, and anti-hammering mechanisms to prevent guessing of a TPM password.

6.4.4 Encrypting the Device with BitLocker

The BitLocker Data Encryption Key (DEK), also known as the Full Volume Encryption Key (FVEK), which encrypts the device's storage volume; the administrator can choose to use either a 128 bit or 256 FVEK, however the instruction in the administrative guidance is use a 256 bit FVEK. The Windows RBG generates the FVEK. The FVEK is ultimately protected by keys within the TPM, namely the Storage Root Key (SRK) and the Storage Primary Seed, the latter is the Root Encryption Key (REK) and is generated by the TPM RBG during initialization. During initialization, the TPM also generates the 2048-bit RSA key pair that is used as the SRK; sealing operations by the SRK in turn protects the BitLocker intermediate keys which are used by Windows when Windows boots (or resumes from hibernation) and so the REK is isolated from operating system and applications, thus preventing reading and exporting the plaintext representation of the REK.

The key hierarchy for BitLocker shows an AES 256 CCM is used to encrypt the Volume Master Key (VMK), which is a KEK and the Full Volume Encryption Key (FVEK), which is a DEK. The FVEK encrypts disk blocks using AES CBC.

The other KEKs are always 256 bits, and so their key size will always the same or larger than the FVEK.

When a user turns on the device, the primary (system) partition is the first data partition that will be unlocked by BitLocker. The Windows OS Loader will prompt the user for the Enhanced PIN which is used to generate a set of intermediate keys, one of which is sealed by the TPM; the ultimate result is a key

[36] Windows creates only one protected storage hierarchy, and that is used by BitLocker.
[37] Draft *Protection Profile PC Client Specific TPM*, FCS_COP.1/AES, page 5.

which decrypts the encrypted VMK, which in turn decrypts the encrypted FVEK, thus enabling the Windows Loader to read the Windows kernel, ntoskrnl.exe, and then transfer execution to the kernel.

The FVEK, VMK, and intermediate keys are all generated by the Windows RBG, or by combining intermediate keys as described in FCS_CKM_EXT.3.

The unencrypted VMKs are zeroized after they are (1) used to encrypt the FVEK and (2) encrypted by an intermediate key. The other keys are also zeroized from volatile memory in the process of generating the VMK. When Windows shuts down normally or goes into hibernation, Windows will zeroize the FVEK as part of shutdown. In the event of a system crash, the BitLocker Crash Dump Filter will zeroize the FVEK in order to prevent the FVEK from being included in the crash dump file.

6.4.5 Key Storage

The Key Isolation Service in Windows hosts secret and private keys within a protected process in order to mitigate tampering or access to sensitive key materials, which can be private keys, secret keys, or other secret material that need to be persisted . The NTFS files that the Key Isolation Service uses to store keys are protected by the Discretionary Access Control security policy described in the *Windows 8, Server 2012 Security Target*. In the NTFS file the key data is further is protected by the Data Protection API (DPAPI), which is described further below. The NTFS files are stored in NTFS volumes which is protected by BitLocker full disk encryption. Please see **Data at Rest Protection** for more information on BitLocker full disk encryption.

The IT administrator can configure Certificate Profiles in a Mobile Device Management (MDM) server for importing keys to the enrolled Windows devices. Applications import keys/secrets into the secure key storage by using the CertificateEnrollmentManager.ImportPfxDataAsync API. In addition, on Windows 10 devices users and local administrators can use the Certificate MMC Snap-in to import keys from Personal Information Exchange (.pfx) files into the secure key storage.

Private keys are protected on disk using DPAPI and BitLocker encryption and access is restricted using the Windows Discretionary Access Control Policy. When a Windows Store Application is deleted the local private keys imported by that app are deleted. All private keys are destroyed when a wipe operation is performed on a device. Local administrators can also perform a wipe on their Windows device to destroy all the keys or secrets. The IT administrator can perform a wipe operation of the enrolled device to destroy the keys.

Windows can restrict access to the application imported key/secret in secure key storage to only the application that imported the key or secret by using the subject identity for the Discretionary Access Control security policy as described in the *Windows 8 Server 2012 Security Target*. Users and local administrators authorize applications at installation to access shared keys or secrets when an application declares the **sharedUserCertificates** capability to share the certificate with other Windows Store Applications for the user. The **sharedUserCertificates** capability is described further in **Restricting Access to System Services**.

Destruction of keys/secrets imported into the secure key storage by applications is conducted automatically by the modern application environment after the keys/secrets are no longer in use.

For the purposes of this Mobile Device evaluation, the cryptographic module is the combination of the operating system and the device running Windows. After the device is configured the only persisted

keys which protect user data via BitLocker are the Storage Root Key held by the TPM, the encrypted VMK (a KEK), and the encrypted FVEK (the DEK). When the device is turned on, the TPM checks the integrity of the SRK as described above, and then the Windows OS Loader unwraps the VMK and FVEK after the user provides the correct authorization factors. When a user provides their password during interactive logon, Windows will use the submask derived from the password to provide access to private keys and secrets protected by DPAPI.

No unencrypted BitLocker key material is transmitted outside the cryptographic module. The encrypted FVEK, VMK, and Intermediate Key are stored on disk as metadata on the storage volume, however the metadata is stored outside of the mounted NTFS volume and so these are never transmitted outside the device, which the boundary of the cryptographic module in this evaluation.

6.4.6 Protecting Data with DPAPI

The Windows RBG generates a DPAPI Master Secret which is used as input into an AES function along with an initialization vector and encryption key, both of which are based on the user's password, to generate the encrypted DPAPI Master Secret. The DPAPI Master Secret is a kind of DEK and the password-based encryption key, which protects the DPAPI Master Secret is a kind of KEK. Also note that the DPAPI Master Secret is ultimately protected by the REK. The password encryption key is generated from a PBKDF2 function takes a result of a one-way function computation of the user's password.[38] Online attacks of the password-based encryption key are prevented by the TOE's implementation of minimum password length, password complexity and maximum incorrect login attempts. Assuming a password of minimum length of eight characters and complexity enforcement turned on, there are at least 1,000,000 possible passwords to guess. A maximum incorrect login attempts setting of three prevents the online attack from succeeding. Offline attacks are prevented because the BitLocker 256-bit FVEK encrypts the DPAPI keys.

Windows will also combine the DPAPI Master Secret along with a salt value which will be used as an encryption key to protect user data, such as a private key. Each user will have a separate encryption key.

The integrity of both the encrypted DPAPI Master Secret and the encryption key is ensured by calculating MAC values.

6.4.7 Networking

Windows has native implementations of IEEE 802.11-2012 and IEEE 802.11ac-2013 to provide secure wireless local area networking (Wi-Fi). Windows can use PRF-384 in WPA2 Wi-Fi sessions and generate AES 128-bit keys or use PRF-704 to generate AES 256-bit keys, both utilize the Windows RBG. Windows complies with the IEEE 802.11-2012 and IEEE 802.11ac-2013 standards and interoperates with other devices that implement the standard. TOE devices have received WPA2 certification, both Enterprise and Personal, and Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance:

- Surface Book

Windows implements key wrapping and unwrapping according to the NIST SP 800-38F specification (the "KW" mode) and so unwraps the Wi-Fi Group Temporal Key (GTK) which was sent by the access point.

[38] Note that data protected by DPAPI is also encrypted by BitLocker when the data is persisted to disk, and so the AES256 encrypted data will be encrypted a second time using the BitLocker 128-bit or 256-bit FVEK.

Because the GTK was protected by AES Key Wrap when it was delivered in an EAPOL-Key frame, the GTK is not exposed to the network.

6.4.7.1 Network Protocols

6.4.7.1.1 TLS and EAP TLS

Windows 10 implements TLS to enable a trusted network path that is used for both EAP, for client and server authentication, as well as HTTPS/ HTTP/TLS.

The following table summarizes the TLS RFCs implemented in Windows:

RFC #	Name	How Used
2246	The TLS Protocol Version 1.0	Specifies requirements for TLS 1.0.
3268	Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)	Specifies additional ciphersuites implemented by Windows.
3546	Transport Layer Security (TLS) Extensions	Updates RFC 2246 with TLS 1.0 extensions implemented by Windows.
4346	The Transport Layer Security (TLS) Protocol Version 1.1	Specifies requirements for TLS 1.1.
4366	Transport Layer Security (TLS) Extensions	Obsoletes RFC 3546 Requirements for TLS 1.1 extensions implemented by Windows.
4492	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)	Specifies additional ciphersuites implemented by Windows.
4681	TLS User Mapping Extension	Extends TLS to include a User Principal Name during the TLS handshake.
5246	The Transport Layer Security (TLS) Protocol Version 1.2	Oboletes RFCs 3268, 4346, and 4366. Specifies requirements for TLS 1.2.
5289	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)	Specifies additional ciphersuites implemented by Windows.
SSL3	The SSL Protocol Version 3	Specifies requirements for SSL3.

Table 16 TLS RFCs Implemented by Windows

These protocols are described at:

- MS-TLSP Transport Layer Security (TLS) Profile
- RFC 2246 The TLS Protocol Version 1.0
- RFC 3268 -AES Ciphersuites for TLS
- RFC 3546 Transport Layer Security (TLS) Extensions
- RFC 4366 Transport Layer Security (TLS) Extensions
- RFC 4492 ECC Cipher Suites for TLS
- RFC 4681 TLS User Mapping Extension
- RFC 5246 - The Transport Layer Security (TLS) Protocol, Version 1.2
- RFC 5289 - TLS ECC Suites with SHA-256384 and AES GCM

The Cipher Suites in Schannel article describes the complete set of TLS cipher suites implemented in Windows (reference: http://msdn.microsoft.com/en-us/library/windows/desktop/aa374757(v=vs.85).aspx), of which the following are used in the evaluated configuration:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- 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_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_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
- 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.

When negotiating a TLS 1.2 elliptic curve cipher suite, Windows will include automatically as part of the Client Hello message both its supported elliptic curves extension, i.e., secp256r1, secp384r1, and secp521r1 as well as signature algorithm, i.e., SHA256, SHA384, and SHA512. Each Windows component that uses TLS checks that the identifying information in the certificate matches what is expected, the component should reject the connection, these checks include checking the expected Distinguished Name (DN), Subject Name (SN), or Subject Alternative Name (SAN) attributes along with the applicable extended key usages. The DN, and any Subject Alternative Name, in the certificate is checked against the identity of the remote computer's DNS entry or IP address to ensure that it matches as described at http://technet.microsoft.com/en-us/library/cc783349(v=WS.10).aspx, and in particular the "Server Certificate Message" section. A certificate the uses a wildcard in the leftmost portion of the resource identifier (i.e., *.contoso.com) can be accepted for authentication, otherwise the certificate will be deemed invalid. Windows does not provide a general-purpose capability to "pin" TLS certificates.

Windows implements HTTPS as described in RFC 2818 so that Windows Store and system applications executing on the TOE can securely connect to external servers using HTTPS.

6.4.7.1.2 IPsec

The Windows IPsec implementation conforms to RFC 4301, Security Architecture for the Internet Protocol.[39] This is documented publicly in the Windows protocol documentation at section 7.5.1 IPsec Overview.[40] Windows implements both RFCS 2409, Internet Key Exchange (IKEv1), and RFC 4306, Internet Key Exchange version 2, (IKEv2).[41] User-mode applications, which include Windows Store Applications, can transparently use IPsec networking services; networking traffic is isolated to the Windows kernel and the IPsec, IPsec Policy Agent, and IKE and AuthIP Keying Module user-mode service processes.

[39] Windows implements IPsec however it was not included in the Mobile Device Fundamentals PP evaluation because there is a separate protection profile for IPsec VPN clients.
[40] Also available as [MS-WSO], *Windows System Overview*, page 43 for offline reading.
[41] [MS-IKEE], *Internet Key Exchange Protocol Extensions*, page 8.

Section **6.5.4** describes which network traffic is not routed through the VPN.

6.4.8 SFR Mapping

The **Cryptographic Support** function satisfies the following SFRs:

- **FCS_CKM.1(ASYM KA), FCS_CKM.2(ASYM AU)**: See **Table 17 Cryptographic Algorithm Standards and Evaluation Methods**.
- **FCS_CKM.1(384), FCS_CKM.1(704), FCS_CKM.2(GTK)**: Windows has a native implementation of IEEE 802.11.
- **FCS_CKM_EXT.1**: The Windows devices in this evaluation use a root key of trust which prevents exporting or tampering the REK. For TPM 2.0 systems, the root encryption key is designated as the Storage Primary Seed.
- **FCS_CKM_EXT.2**: All data encrypting keys are generated by the Windows RBG, which has an input of at least 256 bits of entropy; the Windows 10 data encrypting key is 256 bits in the evaluated configuration.
- **FCS_CKM_EXT.3**: Key encrypting keys have a security strength of 256 bits which is as strong as the 256 bit disk encrypting key.
- **FCS_CKM_EXT.4**: Windows overwrites critical cryptographic parameters immediately after that data is no longer needed.
- **FCS_CKM_EXT.5**: Windows will delete the authorization factor to prevent access to protected data; after a wipe command Windows will format the user data partition to prevent access to protected data.
- **FCS_CKM_EXT.6**: When Windows needs to generate a salt for any kind of signature generation or key agreement, and to derive a key from a passphrase, it uses the Windows random bit generator.
- **FCS_CKM_EXT.7**: Windows will generate an ECDH key on each new pairing.
- **FCS_COP.1(SYM)**: See **Table 18 Cryptographic Algorithm Standards and Evaluation Methods**.
- **FCS_COP.1(HASH)**: See **Table 19 Cryptographic Algorithm Standards and Evaluation Methods**.
- **FCS_COP.1(SIGN)**: See **Table 20 Cryptographic Algorithm Standards and Evaluation Methods**.
- **FCS_COP.1(HMAC)**: See **Table 21 Cryptographic Algorithm Standards and Evaluation Methods**.
- **FCS_COP.1(PBKD)**: Windows implements a FIPS Approved implementation of NIST SP 800-132.
- **FCS_IV_EXT.1**: When it is necessary to generate initialization vectors, Windows follows the guidance in Table 14: References and IV Requirements for NIST-approved Cipher Modes.
- **FCS_RBG_EXT.1**: See **Table 22 Cryptographic Algorithm Standards and Evaluation Methods**.
- **FCS_SRV_EXT.1**: See section **6.4.2 Programming Interfaces**.
- **FCS_STG_EXT.1**: Windows provides secure key storage for private (asymmetric) keys, secret (symmetric) keys, and other data deemed by an authorized subject to require secure storage.
- **FCS_STG_EXT.2**: All keys in Windows are ultimately protected by the TPM-based root of trust for the devices included in this evaluation.
- **FCS_STG_EXT.3**: Key encrypting keys are protected by AES- MAC (CCM) mode.

- **FCS_TLS_EXT.1, FCS_TLS_EXT.2, FCS_HTTPS_EXT.1**: Windows implements TLS 1.0, 1.1, and 1.2 to provide server and mutual authentication, confidentiality and integrity to upper-layer protocols such as Extensible Authentication Protocol and HTTP.

6.5 User Data Protection

6.5.1 Restricting Access to System Services

Windows Store Apps that need programmatic access resources such as device peripherals must declare the capabilities they require as part of the package manifest for the application. There are two types of capabilities, the first is for developers who are registered as having individual accounts in the Windows Store; the second kind is for developers who are registered as having company accounts in the Windows Store. Applications from developers that are registered as companies can have additional capabilities.

The general-use capabilities that apply to most application scenarios are:

General-Use Capability	Description
Music	The **musicLibrary** capability provides programmatic access to the user's Music, allowing the app to enumerate and access all files in the library without user interaction. This capability is typically used in jukebox apps that need to access the entire Music library.
Pictures	The **picturesLibrary** capability provides programmatic access to the user's Pictures, allowing the app to enumerate and access all files in the library without user interaction. This capability is typically used in photo playback apps that need to access the entire Pictures library.
Videos	The **videosLibrary** capability provides programmatic access to the user's Videos, allowing the app to enumerate and access all files in the library without user interaction. This capability is typically used in movie playback apps that need access to the entire Videos library.
Removable Storage	The **removableStorage** capability provides programmatic access to files on removable storage, such as USB keys and external hard drives, filtered to the file type associations declared in the package manifest. For example, if a DOC reader app declared a .doc file type association, it can open .doc files on the removable storage device, but not other types of files.
internetClient	Can receive incoming data from the internet. Cannot act as a server. No local network access.[42]
internetClientClientServer[43]	Can receive incoming data from the internet. Can act as a server. No local network access.
Home and work networks	The **privateNetworkClientServer** capability provides inbound and outbound access to home and work networks through the firewall. This capability is typically used for games that communicate across

[42] This a “least privilege” security measure because many Windows Store Applications need only to receive or send data to remote web services (e.g., social network sites or weather apps) and not communicate with other hosts on the local network.

[43] Most Windows Store Apps that have a web service component will use **internetClient**. Apps that enable peer-to-peer (P2P) scenarios where the app needs to listen for incoming network connections should use **internetClientServer**.

	the local area network (LAN), and for apps that share data across a variety of local devices. On Windows, this capability does not provide access to the internet.
Appointments	The **appointments** capability provides access to the user's appointment store. This capability allows read access to appointments obtained from the synced network accounts and to other apps that write to the appointment store.
Contacts	The **contacts** capability provides access to the aggregated view of the contacts from various contacts stores. This capability gives the app limited access (network permitting rules apply) to contacts that were synced from various networks and the local contact store.
Code generation	The **codeGeneration** capability allows apps to generate code dynamically.
AllJoyn	The **allJoyn** capability allows AllJoyn-enabled apps and devices on a network to discover and interact with each other. All apps that access APIs in the Windows.Devices.AllJoyn namespace must use this capability.
Phone calls	The **phoneCall** capability allows apps to access all of the phone lines on the device and perform the following functions. • Place a call on the phone line and show the system dialer without prompting the user. • Access line-related metadata. • Access line-related triggers. Allows the user-selected spam filter app to set and check block list and call origin information.
Recorded Calls Folder	The **recordedCallsFolder** device capability allows apps to access the recorded calls folder.
User Account Information	The **userAccountInformation** capability gives apps the ability to access the user's name and picture. This capability is required to access some APIs in the Windows.System.User namespace.
VOIP calling	The **voipCall** capability allows apps to access the VOIP calling APIs in the Windows.ApplicationModel.Calls namespace.
3D Objects	The objects3d capability allows apps to have programmatic access to the 3D object files. This capability is typically used in 3D apps and games that need access to the entire 3D objects library. This capability is required to access the folder that contains the 3d objects using APIs in the Windows.Storage namespace
Read Blocked Messages	The blockedChatMessages capability allows apps to read SMS and MMS messages that have been blocked by the Spam Filter app. This capability is required to access the blocked messages using APIs in the Windows.ApplicationModel.Chat namespace.
Chat Message Access	The **chat** capability allows apps to read and delete Text Messages. This capability also allows apps to store chat messages in the system data store.

	This capability is required to use some APIs in the Windows.ApplicationModel.Chat namespace.

Table 23 General Use Capabilities

Device capabilities allow the Windows Store App to access peripheral and internal devices. Device capabilities are specified with the **DeviceCapability** element in the app package manifest.

Device Capability	Description
Location	The **location** capability provides access to location functionality, which you get from dedicated hardware like a GPS sensor in the PC or is derived from available network info. Apps must handle the case where the user has disabled location services from the **Settings** charm.[44]
Microphone	The **microphone** capability provides access to the microphone's audio feed, which allows the app to record audio from connected microphones. Apps must handle the case where the user has disabled the microphone from the **Settings** charm.
Proximity	The **proximity** capability enables multiple devices in close proximity to communicate with one another. This capability is typically used in casual multi-player games and in apps that exchange information. Devices attempt to use the communication technology that provides the best possible connection, including Bluetooth, Wi-Fi, and the internet. This capability is used only to initiate communication between the devices.
Webcam	The **webcam** capability provides access to the video feed of a built-in camera or external webcam, which allows the app to capture photos and videos. On Windows, apps must handle the case where the user has disabled the camera from the **Settings** charm.
USB	The **usb** device capability enables access to APIs in the **Windows.Devices.Usb** namespace.
Human interface device (HID)	The **humaninterfacedevice** device capability enables access to APIs in the **Windows.Devices.HumanInterfaceDevice** namespace. This namespace enables the Windows Store App to access devices that support the Human Interface Device (HID) protocol.
Bluetooth GATT	The **bluetooth.genericAttributeProfile** device capability enables access to APIs in the **Windows.Devices.Bluetooth.GenericAttributeProfile** namespace. This namespace enables the Windows Store App to access Bluetooth LE devices through a collection of primary services, included services, characteristics, and descriptors.
Bluetooth RFCOMM	The **bluetooth.rfcomm** device capability enables access to APIs in the **Windows.Devices.Bluetooth.Rfcomm** namespace. This namespace supports the Basic Rate/Extended Data Rate (BR/EDR) transport and also enables the Windows Store App to access a device that implements Serial Port Profile (SPP).

[44] A charm is an admin tool available by opening the Windows Settings page by swiping from the left side of the screen.

Point of Service (POS)	The pointOfService device capability enables access to APIs in the Windows.Devices.PointOfService namespace. This namespace lets the app access Point of Service (POS) barcode scanners and magnetic stripe readers. The namespace provides a vendor-neutral interface for accessing POS devices from various manufacturers from a Windows Store app.
Bluetooth	The **bluetooth** device capability allows apps to communicate with already paired bluetooth devices. This capability is required to use some APIs in theWindows.Devices.Bluetooth namespace.
Wi-Fi Networking	The **wiFiControl** device capability allows apps to scan and connect to Wi-Fi networks. This capability is required to use some APIs in theWindows.Devices.WiFi namespace.
Radio state	The **radios** device capability allows apps to toggle the Wi-Fi and Bluetooth radios. This capability is required to use the APIs in the Windows.Devices.Radios namespace.
Optical disc	The **optical** device capability allows apps to access functions on optical disk drives such as CD, DVD, and Blu-ray. This capability is required to use some APIs in the Windows.Devices.Custom namespace.
Motion activity	The **activity** device capability allows apps to detect the current motion of the device. This capability is required to use some APIs in the Windows.Devices.Sensors namespace.

Table 24 Device Capabilities

There are additional **special** and **restricted** capabilities associated with Windows Store Applications which are intended for very specific scenarios. Use of these capabilities are highly restricted and subject to additional onboarding policy and review before the App is published to the Windows Store. Apps that apply the **special-use** capabilities require a company account to submit them to the Store. In contrast, **restricted** capabilities do not require a special company account for the Store, they are not available for developers to use. Restricted capabilities are available only to apps that are developed by Microsoft and its partners.

Special-Use Capability	Description
Enterprise authentication	Windows domain credentials, which are domain username and password for a particular user, enable the user to log into remote resources using their credentials, and act as if a user provided their user name and password. The **enterpriseAuthentication** capability is typically used in line-of-business apps that connect to servers within an enterprise and is not needed for basic communications over the Internet. The Enterprise Authentication capability allows a Windows Store App to use the Credential Manager when prompted for domain credentials.
Shared User Certificates	The **sharedUserCertificates** capability enables a Windows Store

	Application to access software and hardware certificates, such as certificates stored on a smart card, the certificate is stored in the user's DPAPI profile location instead of the DPAPI profile associated with the Windows Store Application
Documents	The **documentsLibrary** capability provides programmatic access to the user's Documents, filtered to the file type associations declared in the package manifest, to support offline access to OneDrive. For example, if a DOC reader app declared a .doc file type association, it can open .doc files in Documents, but not other types of files.

Table 25 Special Use Capabilities

As part of installing a Windows Store Application, the user is prompted to authorize the use of the capability by the App, after the App has been installed is it allowed to access the capability when running on behalf of the user. When an App requests to access a resource that is managed by a capability, the Windows App Container, checks if the App has been authorized access, according to the installed package manifest, and then provides mediated access to the resource. In addition to the application-level isolation, Windows also restricts access to hardware resources through the discretionary access control security policy and kernel-mode / user-mode architecture described in the *Windows 8 Server 2012 Security Target*.

6.5.2 Data at Rest Protection

The entire storage volume is protected by BitLocker full disk encryption, this includes user data, Windows configuration (TSF) data, and all programs other than the BitLocker programs needed to unlock the drive. BitLocker uses AES CBC mode with an administrator-specified 128- or 256-bit blocks for Windows 10. The administrative guidance recommends using AES 256.

When the local administrator decides to wipe the device, or the IT administrator decides to wipe the device using a MDM, Windows will delete the BitLocker metadata, which includes the authorization factors that unlock the device. Without the BitLocker metadata, the encrypted data on the storage volume is effectively wiped. The wiping of the BitLocker metadata from flash memory on Windows 10 is performed by first overwriting the metadata with zeros followed by a read-verify. After deleting the metadata, Windows will reboot and install a fresh copy of the operating system from a recovery partition.

6.5.3 Certificate Storage

The MDF PP defines the *Trust Anchor Database* as "[a] list of trusted root Certificate Authority certificates". In a Windows OS, these certificates are known as *trusted root certificates*, which are contained in certificate stores. Each user has their own certificate store and there is a certificate store for the computer account; access to a certificate store is managed by the discretionary access control policy in Windows such that only the authorized administrator, i.e., the user or the local administrator, can add or remove entries.[45] Certificates which are used by applications, for example, TLS, are also placed in certificate stores for the user.

[45] Refer to the Windows 8 Operating System Protection Profile evaluation for more information about the discretionary access control policy.

In addition to the standard certificate revocation processes, application certificates can be loaded by either using administrative tools such as **certutil.exe**, changes to the trusted root certificates can be made using Certificate Trust Lists.

6.5.4 VPN Client

The Windows IPsec VPN client can be configured by the device local administrator or the MDM IT administrator, when the device is enrolled.[46] The administrator can also configure the IPsec VPN client that all IP traffic is routed through the IPsec tunnel except for:

- IKE traffic used to establish the VPN tunnel
- IPv4 ARP traffic for resolution of local network layer addresses and to establish a local address
- IPv6 NDP traffic for resolution of local network layer addresses and to establish a local address

The IPsec VPN is an end-to-end internetworking technology and so VPN sessions can be established over physical network protocols such as wireless LAN (Wi-Fi) or local area network.

The components responsible for routing IP traffic through the VPN client:

- The **IPv4 / IPv6 network stack** in the kernel processes ingoing and outgoing network traffic.
- The **IPsec** and **IKE and AuthIP Keying Modules** service which hosts the IKE and Authenticated Internet Protocol (AuthIP) keying modules. These keying modules are used for authentication and key exchange in Internet Protocol security (IPsec).
- The **Remote Access Service** device driver in the kernel, which is used primarily for VPN connections; known as the “RAS IPsec VPN” or “RAS VPN”.
- The **IPsec Policy Agent** service which enforces IPsec policies.

In addition to the native IPsec VPN Client described above, developers can implement their own VPN client if authorized by Microsoft to use the **networkingVpnProvider** capability.[47]

6.5.5 SFR Mapping

The **User Data Protection** function satisfies the following SFRs:

- **FDP_ACF_EXT.1**: Through the use of capabilities that Windows Store Applications request during installation, Windows restricts system services to Apps.
- **FDP_DAR_EXT.1**: All user data and all Windows data is encrypted on the device.
- **FDP_STG_EXT.1**: Windows provides a trusted and secure store for certificates.
- **FDP_IFC_EXT.1**: Windows provides a VPN Client.
- **FDP_UPC_EXT.1**: Windows provides network transport for TLS, HTTPS, Bluetooth BR/EDR LE that applications can use to ensure communications confidentiality and integrity.
- **FDP_BLT_EXT.1**: Windows uses the device capabilities described in section **6.5.1** to restrict access to Bluetooth devices.

[46] Windows implements IPsec however it was not included in the Mobile Device Fundamentals PP evaluation because there is a separate protection profile for IPsec VPN clients.
[47] See https://msdn.microsoft.com/en-us/library/windows/apps/windows.networking.vpn.aspx.

6.6 Identification and Authentication

All logons are treated essentially in the same manner regardless of their source (e.g., interactive logon, network interface, internally initiated service logon) and start with an account name, domain name (which may be NULL; indicating the local system), and credentials that must be provided to the TSF.

The Local Security Authority component within Windows maintains a count of the consecutive failed logon attempts by security principals from their last successful authentication. When the number of consecutive failed logon attempts is larger than the policy for failed logon attempts, which ranges from 0 (never lockout the account) to 999, Windows 10 will lockout the user account. Windows persists the number of consecutive failed logons on for the user and so rebooting the computer does not reset the failed logon counter. Interactive logons are done on the secure desktop, which does not allow other programs to run, and therefore prevents automated password guessing. In addition, the Windows logon component enforces a one second delay between every failed logon with an increased delay after several consecutive logon failures.

The Windows implementation of Bluetooth follows the Bluetooth SIG Specification, including OBEX data transfer, RFCOMM, L2CAP, and OPP (object push profile). The OBEX specification, which Windows implements, prevents any transfer of user data until both Bluetooth devices have paired, which requires authorization by the Windows user. When a Windows OS encounters an unpaired device, it does not transfer any data to the unpaired device. When paired to a Bluetooth device will reject connection attempts from other devices that purport to use the same Bluetooth address as the connected device. Windows will attempt to authenticate the device connection using the pre-established link key and if there is a failure of the authentication procedure, Windows will terminate the device connection and log an entry into the Windows event log.

6.6.1 Protecting User Data

Windows protects user data with BitLocker, which encrypts the entire device; the user's persistent keys and secrets additionally protected by DPAPI. At the most basic level, all data on stored on the device is encrypted by BitLocker using FIPS Approved symmetric encryption algorithms. During boot, Windows will derive disk encryption keys (DEK) and key encryption keys (KEK) based on the BitLocker authorization factors that unlock the device; the administrative guidance for Windows 10 includes the configuration for an additional BitLocker authorization factors which is a device password, technically known as the "Enhanced PIN", that includes uppercase and lowercase English letters, symbols on an EN-US keyboard, numbers, special characters and spaces. The system and user (protected) data remains encrypted in non-volatile storage, the file system device driver uses the BitLocker FVEK (a DEK) to decrypt the data as it is loaded into volatile storage. The only time user (protected) data is decrypted is after the user authenticates by providing their Enhanced PIN password. The logon password is used to derive the DPAPI secret (a KEK) which provides an additional layer of protection for certain user data, including keys.

6.6.2 X.509 Certificate Validation and Generation

Every Windows component that uses X.509 certificates is responsible for performing certificate validation, however all components use a common subcomponent, which validates certificates as described in RFC 5280 including all applicable usage constraints such as Server Authentication for networking sessions and Code Signing when installing product updates. Every component that uses

X.509 certificates will have a repository for public certificates and will select a certificate based on criteria such as entity name for the communication partner, any extended key usage constraints, and cryptographic algorithms associated with the certificate.

If certificate validation fails, or if Windows is not able to check the validation status for a certificate, Windows will not establish a trusted network channel, however it will inform the user and seek their consent before establishing a HTTPS web browsing session. Certification validation for updates to Windows, mobile applications, and integrity verification is mandatory, neither the administrator nor the user have the option to bypass the results of a failed certificate validation; software installation and updates is further described in **Windows and Application Updates**.

When Windows needs to generate a certificate enrollment request it will include a distinguished name, information about the cryptographic algorithms used for the request, any certification extensions, and information about the client requesting the certificate.

Universal Windows Applications can use these interfaces to check the validity of certificates:

- Certificate.BuildChainAsync
- CertificateChain. Validate

6.6.3 SFR Mapping

- **FIA_AFL_EXT.1**: After the number of consecutive failed authentication attempts for a user account has been surpassed, Windows 10 can be configured to wipe the device.
- **FIA_BLT_EXT.1, FIA_BLT_EXT.2, FIA_BLT_EXT.3, FPT_BLT_EXT.1**: Windows requires Bluetooth mutual authentication between the Windows device and the remote device prior to any data transfer over the Bluetooth connection because all Bluetooth profiles are disabled without an explicit authorization by the user. Windows will also reject any attempts from another Bluetooth device if the address is the same as a device which is already paired. The collection of Windows 10 supported Bluetooth profiles is documented at http://windows.microsoft.com/en-us/windows-10/supported-bluetooth-profiles. Windows 10 operates at security mode 2, service level enforced security, and Bluetooth services proffered by Windows are at the “authorization and authentication” level.
- **FIA_PAE_EXT.1**: Windows conforms to IEEE 802.1X as a Port Access Entity acting in the Supplicant role.
- **FIA_PMG_EXT.1**: Windows devices support logon passwords at least 14 characters in length to as long as 127 characters. Windows 10 logon passwords can be composed from uppercase characters, lowercase characters, digits, and special characters to be used in passwords.
- **FIA_TRT_EXT.1**: Windows logon component enforces a one second delay between every failed logon.
- **FIA_UAU.7**: During an interactive logon, Windows echoes the users password with “*” characters to prevent disclosure of the user’s password.
- **FIA_UAU_EXT.1**: The user must authenticate successfully during interactive logon and prior to decryption of any user data stored on the device.

- **FIA_UAU_EXT.2**: The only actions that an unauthorized user can take when a Windows device is locked is to bring up the authentication dialog or turn the device off.
- **FIA_UAU_EXT.3**: Windows requires that a user provide the correct password prior to changing their password and when unlocking their device.
- **FIA_X509_EXT.1, FIA_X509_EXT.3**: Windows validates X.509 certificates according to RFC 5280 and provides OCSP and CRL services for applications to check certificate revocation status.
- **FIA_X509_EXT.2**: Windows uses X.509 certificates for EAP-TLS exchanges, TLS, HTTPS, code signing for system software updates, code signing for mobile applications, and code signing for integrity verification.

6.7 Security Management

The complete set of management functions are described in **Security Management (FMT),** the following table maps which activities can be done by the device user (who is considered to be a standard user in a Windows client OS), the device administrator (who is considered to be a local administrator), and invoked by a mobile device manager. A checkmark indicates which entity can invoke the management function. A person who uses a Windows 10 device may either be a standard user or a local administrator depending on the kind of user account created for the person. In the terminology of the MDF PP, the device user corresponds to the FMT_MOF_EXT.1.1 requirement and the (device) local administrator or the MDM Agent corresponds to the FMT_MOF_EXT.1.2 requirement because the latter refers to management capabilities after a device has been enrolled into a MDM.

Standard users, or programs running on their behalf, are not able to modify policy or configuration that is set by the administrator, the result is that the user cannot override the configuration specified by the administrator.

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
1. configure password policy: a. minimum password length[48] b. minimum password complexity[49] c. maximum password lifetime[50]	M		√	√
2. configure session locking policy: a. screen-lock enabled/disabled b. screen lock timeout[51] c. number of authentication failures	M		√	√
3. enable/disable the VPN protection: a. across device [~~b. on a per-app basis~~ ***c. no other method***]	M		√	√

[48] The minimum password length can range from 1 to 14 characters.
[49] The complexity requirements include English upper and lowercase characters from A- Z, base 10 digits, non-alphabetic characters, from three of these four categories.
[50] The password lifetime can range from 1 to 999 days.
[51] The timeout can range from 1 minute to 9999 minutes with a default value of 15 minutes.

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
4. enable/disable [**Wi-Fi, Bluetooth,**]	M		√	√
5. enable/disable [**camera, microphone**]: a. across device [~~b. on a per-app basis~~ ***c. no other method***]	M		√	√
6. specify wireless networks (SSIDs) to which the TSF may connect	M		√	√
7. configure security policy for each wireless network: [52] a. [***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		√	√
8. transition to the locked state	M	√	√	
9. TSF wipe of protected data	M		√	
10. configure application installation policy by [***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		√	√
11. import keys/secrets into the secure key storage	M	√	√	-
12. destroy imported keys/secrets and [***[any other keys/secrets]***] in the secure key storage	M	√	√	-
13. import X.509v3 certificates into the Trust Anchor Database	M		√	√
14. remove imported X.509v3 certificates and [***[all X.509v3 certificates]***] in the Trust Anchor Database	M	√		

[52] The configuration data for the Wi-Fi settings can be set by the MDM, the policy is enforced when the computer connects to the Wi-Fi network.

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
15. enroll the TOE in management	M	√	-	-
16. remove applications	M		√	√
17. update system software	M		√	√
18. install applications	M		√	√
19. remove Enterprise applications	M		√	√
20. configure the Bluetooth trusted channel:[53] a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name	M	√	√	
[~~c. allow/disallow additional wireless technologies to be used with Bluetooth~~**,** ***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: [***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	√		
22. enable/disable all data signaling over [**USB hardware ports**][54]	O		√	√
23. enable/disable [**none**]	O			
24. enable/disable developer modes	O		√	√

[53] Windows does not place any restrictions for the kinds of supported Bluetooth profiles and provides an implementation of Bluetooth Discoverable mode and Low Energy (LE) mode.
[54] All of the devices in this evaluation have an external USB interface which can be used for data transfer.

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
25. enable data-at rest protection	O	√	√	
26. enable removable media's data-at-rest protection	O	√	√	
~~27. enable/disable bypass of local user authentication~~	~~O~~	~~O~~	~~O~~	~~O~~
28. wipe Enterprise data	O		√	√
~~29. approve [***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	M	√	√	
~~31. enable/disable the cellular protocols used to connect to cellular network base stations~~	~~O~~			
32. read audit logs kept by the TSF	O	√	√	
33. configure [***certificate***] used to validate digital signature on applications	O		√	√
34. approve exceptions for shared use of keys/secrets by multiple applications	O	O	√	√
35. approve exceptions for destruction of keys/secrets by applications that did not import the key/secret	M	√	√	
36. configure the unlock banner[55]	O		√	√
37. configure the auditable items	O	-	√	
38. retrieve TSF-software integrity verification values	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 [***remote system***][56]	O	√	√	
~~41. enable/disable [selection:~~	~~O~~	~~O~~	~~O~~	~~O~~

[55] The banner can use any text string.
[56] The user can enable/disable backup to a remote system using the "Sync My Settings" settings page. .

Management Function	FMT_SMF EXT.1	FMT_MOF EXT.1.1	Admin	FMT_MOF EXT.1.2
~~a. Hotspot functionality authenticated by [pre-shared key, passcode, no authentication],~~ ~~b. USB tethering authenticated by [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 [~~b. on a per-app basis~~ ***c. no other method***]	M	√	√	
~~45. [none]~~	~~O~~	~~O~~	~~O~~	~~O~~

Table 26 Mobile Device Management Capabilities

The system administrator, through local or group policy or through a MDM, can restrict which applications are installed on the computer by limiting the sources the user is allowed to install applications from and preventing specific applications from being installed. A user is able to uninstall any applications they installed themselves, applications which were installed by the administrator (locally or initiated by a MDM) cannot be removed by the user.[57]

6.7.1 SFR Mapping

The **Security Management** function satisfies the following SFRs:

- **FMT_MOF_EXT.1**: Windows provides the user with the capability to administer the security functions described in the security target. The mappings to specific functions are described in each applicable section of the TOE Summary Specification.
- **FMT_SMF_EXT.1**: Windows provides the management functions that are described by FMT_SMF_EXT.1.1.
- **FMT_SMF_EXT.2**: After unenrollment, Windows will remove enterprise applications and inform the administrator that the device is no longer enrolled.

[57] The MDF PP designates Enterprise Applications as "Applications that are provided and managed by the enterprise", which correspond to applications which are installed by an administrator of the Windows computer.

6.8 Protection of the TSF

6.8.1 Separation and Domain Isolation

The TSF provides a security domain for its own protection and provides process isolation. The security domains used within and by the TSF consists of the following:

- Hardware
- Virtualization Partitions
- Kernel-mode software
- Trusted user-mode processes
- User-mode Administrative tools process
- Application Containers

The TSF hardware is managed by the TSF kernel-mode software and is not modifiable by untrusted subjects. The TSF kernel-mode software is protected from modification by hardware execution state and protection for both physical memory and memory allocated to a partition; an operating system image runs within a partition. The TSF hardware provides a software interrupt instruction that causes a state change from user mode to kernel mode within a partition. The TSF kernel-mode software is responsible for processing all interrupts, and determines whether or not a valid kernel-mode call is being made. In addition, the TSF memory protection features ensure that attempts to access kernel-mode memory from user mode results in a hardware exception, ensuring that kernel-mode memory cannot be directly accessed by software not executing in the kernel mode.

The TSF provides process isolation for all user-mode processes through private virtual address spaces (private per process page tables), execution context (registers, program counters), and security context (handle table and token). The data structures defining process address space, execution context and security context are all stored in protected kernel-mode memory. All security relevant privileges are considered to enforce TSF Protection.

User-mode administrator tools execute with the security context of the process running on behalf of the authorized administrator. Administrator processes are protected like other user-mode processes, by process isolation.

Application Containers ("App Containers") provide an execution environment for Universal Windows Applications which prevents Universal Windows Applications from accessing data created by other Universal Windows Applications except through brokered operating system services such as the File Picker dialog.[58]

By definition, Universal Windows Applications do not have the capability to launch ("execute" in the language of the MDF PP) other programs, the application can read or write to a file.

Like TSF processes, user processes also are provided a private address space and process context, and therefore are protected from each other. Additionally, the TSF has the added ability to protect memory pages using Data Execution Prevention (DEP) which marks memory pages in a process as non-executable

[58] This would be considered "private data" in the terminology of the MDF PP.

unless the location explicitly contains executable code. When the processor is asked to execute instructions from a page marked as data, the processor will raise an exception for the OS to handle.

The TSF implements cryptographic mechanisms within a distinct user-mode process, where its services can be accessed by both kernel- and user-mode components, in order to isolate those functions from the rest of the TSF to limit exposure to possible errors while protecting those functions from potential tampering attempts.

Furthermore, the TSF includes a Code Integrity Verification feature, also known as Kernel-mode code signing (KMCS), whereby device drivers will be loaded only if they are digitally signed by either Microsoft or from a trusted root certificate authority recognized by Microsoft. KMCS uses public-key cryptography technology to verify the digital signature of each driver as it is loaded. When a driver tries to load, the TSF decrypts the hash included with the driver using the public key stored in the certificate. It then verifies that the hash matches the one that it computes based on the driver code using the FIPS - certified cryptographic libraries in the TSF. The authenticity of the certificate is also checked in the same way, but using the certificate authority's public key, which must be configured in and trusted by the TOE.

6.8.1.1 Supporting Hardware

The devices used in the evaluation have the following characteristics:

6.8.1.1.1 Processor and Memory

Device	Processor	Hardware Specifications
Microsoft Surface Book	Intel Core i7-4650U	http://www.intel.com/content/www/us/en/processors/core/4th-gen-core-family-mobile-u-y-processor-lines-vol-1-datasheet.html (See section 2.1 System Memory Interface for MMU description and section 3.9 for description of enforcement of read, write and execute permissions with support from Execute Disable Bit technology. The Execute Disable Bit is used to segregate areas of memory for use by either storage of processor instructions for execution of code or by storage of data for read/write access)

Table 27 Supporting Hardware Specifications

6.8.1.1.2 Frequency Ranges

Device	Wi-Fi	Bluetooth	Broadband
Microsoft Surface Book	2412 MHz – 2462 MHz 5180 – 5320 MHz 5500 MHz – 5700 MHz 5745 MHz – 5855 MHz	2402 MHz – 2480 MHz	N.A.

6.8.1.1.3 Mobile Broadband Isolation

None of the devices used in this evaluation include the ability to initiate or receive telephony calls and so the devices do not contain a dialer or any USSD or MMI codes.

Windows does not have an "auxiliary boot mode" that is distinctly used by a wired interface.[59]

6.8.2 Protection from Implementation Weaknesses

Windows runs on processors that provide support for virtual memory and enforce restrictions to read, write, and execute pages of virtual and physical memory. Collectively, this is known as Data Execution Prevention (DEP). On Intel platforms, DEP is called NX (no execute).

Windows provides a default heap allocator for use by user-mode processes; Windows applications can use the default heap or implement their own allocator. The heap is managed with a collection of metadata (which isn't pre-allocated to a specific address), with integrity protection provided by internal checksums and encoding the metadata. If the heap detects corruption due to a heap overrun (e.g. integrity checks fail), and heap termination on corruption is enabled for the process, then the process is immediately terminated.

The Windows kernel, user-mode applications, and all Windows Store Applications implement Address Space Layout Randomization (ASLR) in order to load executable code at unpredictable base addresses. The base address is generated using a pseudo-random number generator that is seeded by high quality entropy sources when available which provides at least 8 random bits for memory mapping. [60]

The Windows runtime also provides stack buffer overrun protection capability that will terminate a process after Windows detects a potential buffer overrun on the thread's stack by checking canary values in the function prolog and epilog as well as reordering the stack. All Windows binaries and Windows Store Applications implement stack buffer overrun protection by being complied with the /GS option, which is used for all Windows binaries; and checking that all Windows Store Applications are compiled with buffer overrun protection before ingesting the Windows Store Application into the Windows Store.

To enable these protections using the Microsoft Visual Studio development environment, programs are complied with /DYNAMICBASE option for ASLR, and optionally with /HIGHENTROPYVA for 64-bit ASLR, or /NXCOMPAT:NO to opt out of software-based DEP, and /GS (switched on by default) for stack buffer overrun protection.

Windows Store Applications are compiled with the /APPCONTAINER option which builds the executable to run in a Windows appcontainer, to run with the user-mode protections described in this section.

6.8.3 Time Service

Each hardware platform supported by the TOE includes a real-time clock as the primary time source. The real-time clock is a device that can only be accessed using functions provided by the TSF and serves as the reference clock that maintains the system time. Specifically, the TSF provides functions that allow users, including the TSF itself, to query and set the clock, as well as functions to synchronize clocks within a domain. The ability to query the clock is unrestricted, while the ability to set the clock requires

[59] See page 106 of the MDF PP for details about "auxiliary boot mode".
[60] The PRNG is seeded by the TPM RBG, the RDRAND instruction and other sources.

the SeSystemtimePrivilege. This privilege is only granted to authorized administrators to protect the integrity of the time service.

Synchronizing the clocks within a managed Windows deployment is critical for cross-machine communications and correlating activities which occur on multiple computers. Accuracy (which the NIAP OS PP describes as "reliable and monotonically increasing" is described in How the Windows Time Service Works. In addition this communications path can be protected using IPsec between the computers in the Active Directory domain.

How To Configure an Authoritative Time Server in Windows Server describes additional steps a domain administrator can take to explicitly specify the reference clock for the domain or an arbitrary NTP server. If Windows connects to a broadband network, it will use the network's time server as a secondary time server in the same manner as a domain or a NTP time source.

Windows capabilities that are included in the ongoing OS protection profile evaluation which use the centralized (i.e., reliable) time service are:

- Audit record generation
- Network expirations for authentication and data access
- Session timeout and screen locking
- X.509 certificate generation, revocation, and expiration

These capabilities use the interfaces described at http://msdn.microsoft.com/en-us/library/ms725473(v=vs.85).aspx. Public documentation about time functions in Windows is located at http://msdn.microsoft.com/en-us/library/ms724962(v=vs.85).aspx. This describes the different types of time services offered to developers.

6.8.4 Self-Tests

The Windows self-tests are a collection of tests which verify that the Windows is operating correctly. The self-tests are enabled when the administrator sets the "System Cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing" policy; Windows will always run the self-tests described in this section.

The kernel-mode startup self-tests are:[61]

- AES-128 encrypt/decrypt EBC Known Answer Test
- AES-128 encrypt/decrypt CBC Known Answer Test
- AES-128 CMAC Known Answer Test
- AES-128 encrypt/decrypt CCM Known Answer Test
- AES-128 encrypt/decrypt GCM Known Answer Test
- RSA Known Answer Test
- ECDSA sign/verify test on P256 curve
- ECDH secret agreement Known Answer Test on P256 curve
- HMAC-SHA-1 Known Answer Test
- HMAC-SHA-256 and HMAC-SHA-512 Known Answer Tests

[61] When the System Cryptography policy is set, Windows will always perform these self-tests however the evaluated configuration does not use the ECDH, HMAC, and SP800-56A algorithms.

- SP800-56A concatenation KDF Known Answer Tests (same as Diffie-Hellman KAT)
- SP800-90 AES-256 counter mode DRBG Known Answer Tests (instantiate, generate and reseed)

The Windows kernel-mode cryptographic module, the Kernel Mode Cryptographic Primitives Library, also performs pair-wise consistency checks upon each invocation of RSA, ECDH, and ECDSA key-pair generation and import as defined in FIPS 140-2. SP 800-56A conditional self-tests are also performed. A continuous RNG test (CRNGT) is used for the random number generators of this cryptographic module. All approved and non-approved RNGs have a CRNGT. The SP 800-90 DRBGs have health tests. A pair-wise consistency test is done for Diffie-Hellman.

The Kernel Mode Cryptographic Primitives Library is loaded into the kernel's memory early during the boot process. If there is a failure in any startup self-test, the Kernel Mode Cryptographic Primitives Library DriverEntry function will fail to return the STATUS_SUCCESS status to its caller. The only way to recover from the failure of a startup self-test is to attempt to invoke DriverEntry again, which will rerun the self-tests, and will only succeed if the self-tests passes.

By thoroughly exercising the cryptographic functions, Windows will prevent situations where user data is not stored in an encrypted state.

All operations on the TSF ultimately involve the use of cryptography, and so these tests are sufficient to determine that Windows is operating correctly.

6.8.5 Windows Code Integrity

A Windows operating system verifies the integrity of Windows program code using the Secure Boot and Code Integrity capability in Windows.[62] On computers with a TPM, such as those used in the Mobile Device evaluation, before Windows will unlock the operating system drive, it will verify the integrity of the early boot components, which include the Boot Loader, OS Loader, and OS Resume binaries, in order to prevent tampering and to ensure that the drive is in the same computer as when the OS was initialized.

The Secure Boot capability Windows checks that the file integrity of early boot components has not been compromised and ensures that the files have not been modified, which mitigates the risk of rootkits and viruses, and that the data elements that contribute to creating the composite keys, which will ultimately unlock the operating system drive, have not been compromised. Secure Boot collects these file measurements and seals them to the TPM. When Secure Boot starts in the preboot environment, it will compare the sealed values from the TPM and if those values do not match the calculated values, Secure Boot will lock the system (which prevents booting) and display a warning on the computer display.

After Secure Boot verifies the integrity of early-running kernel components, including Code Integrity, the Code Integrity capability provides measures code integrity for kernel-mode and user-mode programs. Kernel-mode code signing (KMCS) prevents kernel-mode device drivers, such as the BitLocker Drive Encryption Drivers (fvevol.sys), from loading unless they are published and digitally signed by developers who have been vetted by one of a handful of trusted certificate authorities (CAs). KMCS, using public-

[62] In MDF PP terminology, Windows runs on the application processor.

key cryptography technologies, requires that kernel-mode code include a digital signature generated by one of the trusted certificate authorities. When a kernel device driver tries to load, Windows decrypts the hash included with the driver using the public key stored in the certificate, then verifies that the hash matches the one computed with the code. The authenticity of the certificate is checked in the same way, but using the certificate authority's public key, which is trusted by Windows. The root public key of the certificate chain that verifies the signature must match one of the Microsoft's root public keys indicating that Microsoft is the publisher of the Windows image files. These Microsoft's root public keys are hardcoded in the Windows boot loader.

6.8.6 Windows and Application Updates

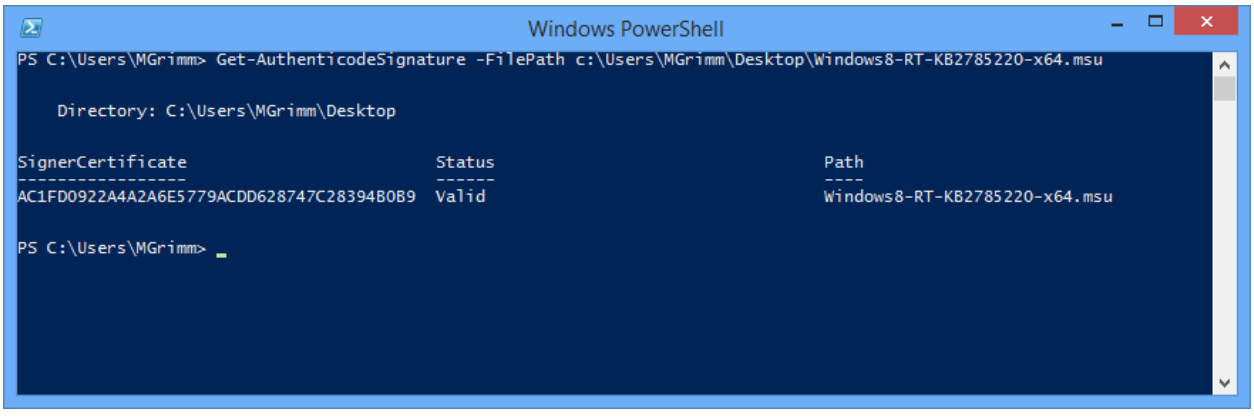
Updates to Windows are delivered as Microsoft Update Standalone Package files (.msu files) and are signed by Microsoft with two digital signatures, a SHA1 signature for legacy applications and a SHA256 signature for modern applications. The RSA SHA256 digital signature is signed by *Microsoft Corporation*, with a certification path through a Microsoft Code Signing certificate and ultimately the Microsoft Root Certification Authority. These certificates are checked by the Windows Trusted Installer prior to installing the update.

The Windows operating system will check that the certificate is valid and has not been revoked using a standard PKI CRL. Once the Trusted Installer determines that the package is valid, it will update Windows; otherwise the installation will abort and there will be an error message in the event log.. Note that the Windows installer will not install an update if the files in the package have lower version numbers than the installed files.

The integrity of the Microsoft Code Signing certificate on the computer is protected by the storage root key within the TPM, and the validated integrity of the Windows binaries as a result of Secure Boot and Code Integrity.

Updates to Windows are delivered through the Windows Update capability, which is enabled by default, or the user can go to http://www.microsoft.com/security/default.aspx to search and obtain security updates on their own volition.

A user can then check that the signature is valid either by viewing the digital signature details of the file from Windows Explorer or by using the Get-`AuthenticodeSignature` PowerShell Cmdlet. The following is an example of using PowerShell:



If the Get-`AuthenticodeSignature` PowerShell Cmdlet or Windows Explorer could not verify the signature, the status will be marked as invalid. This verification check uses the same functionality described above.

6.8.6.1.1 Windows Store Applications

In the same manner as checking the integrity of the Microsoft Update Packages and Windows executable code, Windows Store Applications and their installation packages are verified using a digital signature from *Microsoft Corporation* with the Code Signing usage.

6.8.7 SFR Mapping

The **TSF Protection** function satisfies the following SFRs:

- **FPT_AEX_EXT.1**: All Windows Store applications use address space layout randomization.
- **FPT_AEX_EXT.2**: The Intel processors included in this evaluation enforce read, write, and execute permissions for physical memory.
- **FPT_AEX_EXT.3**: Windows binaries are compiled with stack overflow protection (compiled using the **/Gs** option for native applications). **Appendix B: Interfaces and Binaries** contains a list of Windows binaries along with any exceptions which do not use stack overflow protection.
- **FPT_AEX_EXT.4**: The Windows kernel and user-mode system services protect themselves from modification by untrusted subject programs; moreover user-mode programs execute in separate virtual address spaces.
- **FPT_BBD_EXT**.1: The application processor and the baseband processor for the Microsoft Surface Book is separated such that the baseband processor can only access resources of the application processor under the control of the application processor.
- **FPT_KST_EXT.1**: During normal operation, Windows does not store plaintext key material in non-volatile storage.
- **FPT_KST_EXT.2**: Plaintext keys are not exported from the operating system as described in section Error! Reference source not found..
- **FPT_KST_EXT.3**: Users cannot export plain text keys from Windows Store applications.
- **FPT_NOT_EXT.1(AUDIT)**: Windows will fall into a non-operational state after a failure of the Windows cryptographic self-tests and integrity failure for Windows system binaries.
- **FPT_NOT_EXT.1(ATTEST):** When configured to generate health attestations, Windows will use the Attestation Key (AK) in the TPM along with the associated certificate issued by Microsoft to notify the remote administrator via a MDM.
- **FPT_STM.1**: The real-time clock in each Windows platform, in conjunction with periodic domain synchronization, for domain-joined devices, provide a reliable source of time stamps for the TSF; changing the clock can be restricted to authorized administrators.
- **FPT_TST_EXT.1**: Windows runs a series of self-tests that confirm that essential cryptographic operations are performed correctly and halts if the self-tests fail. Those cryptographic functions are then used to check integrity of TOE executables.
- **FPT_TST_EXT.2**: Windows checks the integrity of the Windows boot loader, OS loader, kernel, and system binaries and all application executable code, i.e., Windows Store Applications and updates to Windows and Windows Store Applications.

- **FPT_TUD_EXT.1**: Windows provides a means to identify the current version of the Windows software, the hardware model, and installed applications.
- **FPT_TUD_EXT.2**: Windows has an update mechanism to deliver updated binaries and a means for a user to confirm that the digital signatures, which ensure the integrity of the update, are valid for both the operating system and Windows Store Applications.

6.9 TOE Access

Windows provides the ability for a user to lock their interactive logon session at their own volition or after a user-defined inactivity timeout. Windows also provides the ability for the administrator to specify the interval of inactivity after which the session will be locked. This policy will be applied to either the local machine or the computers within a domain using either local policy or group policy respectively. If both the administrator and a standard user specify an inactivity timeout period, Windows will lock the session when the shortest time period expires.

Once a user has a desktop session, they can invoke the session locking function by using the same key sequence used to invoke the trusted path (**Ctrl+Alt+Del**). This key sequence is captured by the TSF and cannot be intercepted or altered by any user process. The result of that key sequence is a menu of functions, one of which is to lock the workstation. The user can also lock their desktop session by going to the Start screen, selecting their logon name, and then choosing the "Lock" option.

Windows constantly monitors the mouse, keyboard, touch display, and the orientation sensor for inactivity in order to determine if they are inactive for the specified time period. After which, Windows will lock the workstation and execute the screen saver unless the user is streaming video such as a movie. Note that if the workstation was not locked manually, the TSF will lock the display and start the screen saver program if and when the inactivity period is exceeded, as well any notifications from applications which have registered to publish the application's badge or the badge with associated notification text to the locked screen.[63] The user has the option to not display any notifications, or choose one Windows Store Application to display notification text, and select other applications display their badge.

For Windows 10 the inbox Calendar, Weather, and Alarm applications can generate notifications, and when selected to display notification text they will show the location and time of the upcoming and in-progress meeting, the current weather conditions, and an expired alarm times. In addition, Mail application can be configured to display a badge but not notification text.

After the computer was locked, in order to unlock their session, the user either presses a key or swipes the display. The user must provide the **Ctrl+Alt+Del** key combination if the **Interactive Logon: Do not required CTRL+ALT+DEL** policy is set to disabled.[64] Either action will result in an authentication dialog. The user must then re-enter their authentication data, which has been cached by the local system from the initial logon, after which the user's display will be restored and the session will resume. Alternately, an authorized administrator can enter their administrator identity and password in the authentication

[63] The badge is a logo which represents the Windows Store Application and the notification text can be items such as a count of unread messages or an appointment.
[64] This policy is defined under Local Policies / Security Options.

dialog. If the TSF can successfully authenticate the administrator, the user will be logged off, rather than returning to the user's session, leaving the workstation ready to authenticate a new user.

As part of establishing the interactive logon session, Windows can be configured to display a logon banner, which is specified by the administrator, that the user must accept prior to establishing the session.

6.9.1 SFR Mapping

The **TOE Access** function satisfies the following SFRs:

- **FTA_SSL_EXT.1**: Windows 10 will transition to a locked state when there is an administrator-specified period of inactivity or when the user explicitly locks the device.
- **FTA_WSE_EXT.1**: An authorized administrator can specify which Wi-Fi networks to connect to, as specified in FMT_SMF.1.
- **FTA_TAB.1**: An authorized administrator can define and modify a banner that will be displayed prior to allowing a user to logon.

6.10 Trusted Path / Channels

Windows Store applications used the HttpClient interface to establish a secure HTTPS/TLS channel. Windows Store applications do not have access to low level interfaces to perform TLS, the HttpClient interface supports performing TLS in the context of an HTTPS connection by passing a HTTPS Uniform Resource Identifier (URI) to the HttpClient constructor. When a HTTPS URI is used then TLS will be used when establishing the HTTP connection. Mobile Device Managers use HTTPS/TLS: the mobile device authenticates against the MDM to check the identity of the MDM service, and the MDM authenticates the client to ensure the identity of the client device.

Third party VPN Windows Store applications use the Windows.Networking.Vpn interface to establish an IPsec VPN channel.

Windows implements IEEE 802.11-2012, IEEE 802.1X and EAP-TLS to provide authenticated wireless networking sessions when requested by the user.

The specific details for each protocol are described in section Windows has native implementations of IEEE 802.11-2012 and IEEE 802.11ac-2013 to provide secure wireless local area networking (Wi-Fi). Windows can use PRF-384 in WPA2 Wi-Fi sessions and generate AES 128-bit keys or use PRF-704 to generate AES 256-bit keys, both utilize the Windows RBG. Windows complies with the IEEE 802.11-2012 and IEEE 802.11ac-2013 standards and interoperates with other devices that implement the standard. TOE devices have received WPA2 certification, both Enterprise and Personal, and Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance:

- Surface Book

Windows implements key wrapping and unwrapping according to the NIST SP 800-38F specification (the "KW" mode) and so unwraps the Wi-Fi Group Temporal Key (GTK) which was sent by the access point. Because the GTK was protected by AES Key Wrap when it was delivered in an EAPOL-Key frame, the GTK is not exposed to the network.

Network Protocols.

To summarize the **Trusted Path / Channel** function satisfies this SFR:

- **FPT_ITC_EXT.1**: Windows provides several trusted network channels that protect data in transit from disclosure, provide data integrity, and endpoint identification that is used by 802.11-2012, 802.1X, EAP-TLS, TLS, and HTTPS.

6.11 Security Response Process

Microsoft utilizes industry standard practices to address reported product vulnerabilities. This includes a central email address (secure@microsoft.com) to report issues (as described at https://technet.microsoft.com/en-us/security/ff852094), timely triage and root cause analysis, and responsible resolution of the report which may result in the release of a binary update. If a binary update is required, it is made available through automated channels to all customers following the process described at https://technet.microsoft.com/en-us/security/dn436305. If the sender wishes to send secure email, there is a public PGP key for S/MIME at https://technet.microsoft.com/en-us/security/dn606155.aspx. Security updates for Microsoft products – operating system, firmware, and applications – are delivered as described in section 6.8.6.

7 Protection Profile Conformance Claim

This section provides the protection profile conformance claim and supporting justifications and rationale.

7.1 Rationale for Conformance to Protection Profile

This Security Target is in strict compliance and exact conformance with the Mobile Device Fundamentals Protection Profile, version 2.0, September 17, 2014.

For all of the content incorporated from the protection profile, the corresponding rationale in that protection profile remains applicable to demonstrate the correspondence between the TOE security functional requirements and TOE security objectives.

The requirements in the protection profile are assumed to represent a complete set of requirements that serve to address any interdependencies. Given that all of the functional requirements in the protection profile have been copied into this security target, the dependency analysis for those requirements is not reproduced here.

8 Rationale for Modifications to the Security Requirements

This section provides a rationale that describes how the Security Target reproduced the security functional requirements and security assurance requirements from the protection profile.

8.1 Functional Requirements

This Security Target includes security functional requirements (SFRs) that can be mapped to SFRs found in the protection profile along with SFRs that describe additional features and capabilities. The mapping from protection profile SFRs to security target SFRs along with rationale for operations is presented in Table 28. SFR operations left incomplete in the protection profile have been completed in this security and are identified within each SFR in section 5.1 TOE Security Functional Requirements.

Table 28 Rationale for Operations

MDF PP Requirement	ST Requirement	Operation & Rationale
FAU_GEN.1	FAU_GEN.1	Multiple selections which are allowed by the PP.
FAU_SAR.1	FAU_SAR.1	Copied from the PP without changes.
FAU_SEL.1	FAU_SEL.1	An assignment which is allowed by the PP.
FAU_STG.1	FAU_STG.1	Copied from the PP without changes.
FAU_STG.4	FAU_STG.4	Copied from the PP without changes.
FCS_CKM.1(1)	FCS_CKM.1(ASYM KA)	A selection which is allowed by the PP.
FCS_CKM.1(2)	FCS_CKM.1(WLAN384)	Copied from the PP without changes.
FCS_CKM.1(3)	FCS_CKM.1(WLAN704)	Copied from the PP without changes.
FCS_CKM.1(2)	FCS_CKM.1(ASYM AU)	A selection which is allowed by the PP.
FCS_CKM.2(2)	FCS_CKM.2(GTK)	Copied from the PP without changes.
FCS_CKM_EXT.1	FCS_CKM_EXT.1	Multiple selections which are allowed by the PP.
FCS_CKM_EXT.2	FCS_CKM_EXT.2	Copied from the PP without changes.
FCS_CKM_EXT.3	FCS_CKM_EXT.3	Two selections which are allowed by the PP.
FCS_CKM_EXT.4	FCS_CKM_EXT.4	Copied from the PP without changes.
FCS_CKM_EXT.5	FCS_CKM_EXT.5	A selection which is allowed by the PP.
FCS_CKM_EXT.6	FCS_CKM_EXT.6	Copied from the PP without changes.
FCS_CKM_EXT.7	FCS_CKM_EXT.7	An assignment which is allowed by the PP.
FCS_COP.1(1)	FCS_COP.1(SYM)	A selection which is allowed by the PP.
FCS_COP.1(2)	FCS_COP.1(HASH)	Two selections which are allowed by the PP.
FCS_COP.1(3)	FCS_COP.1(SIGN)	A selection which is allowed by the PP.
FCS_COP.1(4)	FCS_COP.1(HMAC)	Two selections which are allowed by

MDF PP Requirement	ST Requirement	Operation & Rationale
		the PP.
FCS_COP.1(5)	FCS_COP.1(PBKD)	Two selections which are allowed by the PP.
FCS_IV_EXT.1	FCS_IV_EXT.1	Copied from the PP without changes.
FCS_RBG_EXT.1	FCS_RBG_EXT.1	Multiple selections which are allowed by the PP.
FCS_SRV_EXT.1	FCS_SRV_EXT.1	A selection which is allowed by the PP and refinements to switch to the SFR labels used in the security target.
FCS_STG_EXT.1	FCS_STG_EXT.1	Multiple selections which are allowed by the PP.
FCS_STG_EXT.2	FCS_STG_EXT.2	Multiple selections which are allowed by the PP.
FCS_STG_EXT.3	FCS_STG_EXT.3	Multiple selections which are allowed by the PP.
FCS_TLSC_EXT.1	FCS_TLSC_EXT.1	Three selections which are allowed by the PP.
FCS_TLSC_EXT.2	FCS_TLSC_EXT.2	Two selections which are allowed by the PP.
FCS_HTTPS_EXT.1	FCS_HTTPS_EXT.1	A selection which is allowed by the PP.
FDP_ACF_EXT.1	FDP_ACF_EXT.1	Multiple selections which are allowed by the PP.
FDP_DAR_EXT.1	FDP_DAR_EXT.1	Two selections which are allowed by the PP.
FDP_IFC_EXT.1	FDP_IFC_EXT.1	A selection which is allowed by the PP.
FDP_STG_EXT.1	FDP_STG_EXT.1	Copied from the PP without changes.
FDP_UPC_EXT.1	FDP_UPC_EXT.1	A selection which is allowed by the PP.
FDP_BLT_EXT.1	FDP_BLT_EXT.1	Copied from the PP without changes.
FIA_AFL_EXT.1	FIA_AFL_EXT.1	One assignment which is allowed by the PP.
FIA_BLT_EXT.1	FIA_BLT_EXT.1	Multiple assignment which are allowed by the PP.
FIA_BLT_EXT.2	FIA_BLT_EXT.2	Copied from the PP without changes.
FIA_BLT_EXT.3	FIA_BLT_EXT.3	Copied from the PP without changes.
FIA_PAE_EXT.1	FIA_PAE_EXT.1	Copied from the PP without changes.
FIA_PMG_EXT.1	FIA_PMG_EXT.1	An assignment and a selection which is allowed by the PP.
FIA_TRT_EXT.1	FIA_TRT_EXT.1	A selection which is allowed by the PP.
FIA_UAU.7	FIA_UAU.7	Copied from the PP without changes.
FIA_UAU_EXT.1	FIA_UAU_EXT.1	A selection which is allowed by the PP.
FIA_UAU_EXT.2	FIA_UAU_EXT.2	A selection which is allowed by the

MDF PP Requirement	ST Requirement	Operation & Rationale
		PP.
FIA_UAU_EXT.3	FIA_UAU_EXT.3	A selection which is allowed by the PP.
FIA_X509_EXT.1	FIA_X509_EXT.1	A selection which is allowed by the PP.
FIA_X509_EXT.2	FIA_X509_EXT.2	Four selections which are allowed by the PP.
FIA_X509_EXT.3	FIA_X509_EXT.3	Copied from the PP without changes.
FMT_MOF_EXT.1	FMT_MOF_EXT.1	Copied from the PP without changes.
FMT_SMF_EXT.1	FMT_SMF_EXT.1	Multiple selections, assignments, and refinements which are allowed by the PP.
FMT_SMF_EXT.2	FMT_SMF_EXT.2	A selection which is allowed by the PP.
FPT_AEX_EXT.1	FPT_AEX_EXT.1	Copied from the PP without changes.
FPT_AEX_EXT.2	FPT_AEX_EXT.2	A selection which is allowed by the PP.
FPT_AEX_EXT.3	FPT_AEX_EXT.3	Copied from the PP without changes.
FPT_AEX_EXT.4	FPT_AEX_EXT.4	Copied from the PP without changes.
FPT_BBD_EXT.1	FPT_BBD_EXT.1	Copied from the PP without changes.
FPT_BLT_EXT.1	FPT_BLT_EXT.1	Assignment allowed by the PP.
FPT_KST_EXT.1	FPT_KST_EXT.1	Copied from the PP without changes.
FPT_KST_EXT.2	FPT_KST_EXT.2	Copied from the PP without changes.
FPT_KST_EXT.3	FPT_KST_EXT.3	Copied from the PP without changes.
FPT_NOT_EXT.1	FPT_NOT_EXT.1(AUDIT)	Two selections which are allowed by the PP.
FPT_NOT_EXT.1	FPT_NOT_EXT.1(ATTEST)	Three selections which are allowed by the PP.
FPT_STM.1	FPT_STM.1	Copied from the PP without changes.
FPT_TST_EXT.1	FPT_TST_EXT.1	Copied from the PP without changes.
FPT_TST_EXT.2	FPT_TST_EXT.2	Two selections which are allowed by the PP.
FPT_TUD_EXT.1	FPT_TUD_EXT.1	Copied from the PP without changes.
FPT_TUD_EXT.2	FPT_TUD_EXT.2	Five selections which are allowed by the PP.
FTA_SSL_EXT.1	FTA_SSL_EXT.1	Assignment allowed by the PP.
FTA_WSE_EXT.1	FTA_WSE_EXT.1	Copied from the PP without changes.
FTA_TAB.1	FTA_TAB.1	Copied from the PP without changes.
FTP_ITC_EXT.1	FTP_ITC_EXT.1	Two selections which are allowed by the PP.

8.2 Security Assurance Requirements

The statement of security assurance requirements (SARs) found in section **5.2 TOE Security Assurance Requirements**, is in strict conformance with the Mobile Device Fundamentals Protection Profile.

8.3 Rationale for the TOE Summary Specification

This section, in conjunction with section 6, the **TOE Summary Specification (TSS)**, provides evidence that the security functions are suitable to meet the TOE security requirements.

Each subsection in section 6, TOE Security Functions (TSFs), describes a Security Function (SF) of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding SF. The set of security functions work together to satisfy all of the functional requirements. Furthermore, all the security functions are necessary in order for the TSF to provide the required security functionality.

The set of security functions work together to provide all of the security requirements as indicated in Table 29. The security functions described in the TOE Summary Specification and listed in the tables below are all necessary for the required security functionality in the TSF.

Table 29 Requirement to Security Function Correspondence

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
FAU_GEN.1	X								
FAU_SAR.1	X								
FAU_SEL.1	X								
FAU_STG.1	X								
FAU_STG.4	X								
FCS_CKM.1(ASYM KA)		X							
FCS_CKM.1(WLAN384)		X							
FCS_CKM.1(WLAN704)		X							
FCS_CKM.1(ASYM AU)		X							
FCS_CKM.2(GTK)		X							
FCS_CKM_EXT.1		X							
FCS_CKM_EXT.2		X							
FCS_CKM_EXT.3		X							
FCS_CKM_EXT.4		X							
FCS_CKM_EXT.5		X							
FCS_CKM_EXT.6		X							
FCS_CKM_EXT.7		X							
FCS_COP.1(SYM)		X							
FCS_COP.1(HASH)		X							
FCS_COP.1(SIGN)		X							

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
FCS_COP.1(HMAC)		X							
FCS_COP.1(PBKD)		X							
FCS_IV_EXT.1		X							
FCS_RBG_EXT.1		X							
FCS_SRV_EXT.1		X							
FCS_STG_EXT.1		X							
FCS_STG_EXT.2		X							
FCS_STG_EXT.3		X							
FCS_TLSC_EXT.1		X							
FCS_TLSC_EXT.2		X							
FCS_HTTPS_EXT.1		X							
FDP_ACF_EXT.1			X						
FDP_DAR_EXT.1			X						
FDP_IFC_EXT.1			X						
FDP_STG_EXT.1			X						
FDP_UPC_EXT.1			X						
FDP_BLT_EXT.1			X						
FIA_AFL_EXT.1				X					
FIA_BLT_EXT.1				X					
FIA_BLT_EXT.2				X					
FIA_PAE_EXT.1				X					
FIA_PMG_EXT.1				X					
FIA_TRT_EXT.1				X					
FIA_UAU.7				X					
FIA_UAU_EXT.1				X					
FIA_UAU_EXT.2				X					
FIA_UAU_EXT.3				X					
FIA_X509_EXT.1				X					
FIA_X509_EXT.2				X					
FIA_X509_EXT.3				X					
FMT_MOF_EXT.1					X				
FMT_SMF_EXT.1					X				
FMT_SMF_EXT.2					X				
FPT_AEX_EXT.1						X			
FPT_AEX_EXT.2						X			
FPT_AEX_EXT.3						X			

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
FPT_AEX_EXT.4						X			
FPT_BBD_EXT.1						X			
FPT_BLT_EXT.1						X			
FPT_KST_EXT.1						X			
FPT_KST_EXT.2						X			
FPT_KST_EXT.3						X			
FPT_NOT_EXT.1(AUDIT)						X			
FPT_NOT_EXT.1(ATTEST)						X			
FPT_STM.1						X			
FPT_TST_EXT.1						X			
FPT_TST_EXT.2						X			
FPT_TUD_EXT.1						X			
FPT_TUD_EXT.2						X			
FTA_SSL_EXT.1								X	
FTA_WSE_EXT.1								X	
FTA_TAB.1								X	
FTP_ITC_EXT.1									X

9 Appendix A: List of Abbreviations

Abbreviation	Meaning
3DES	Triple DES
ACE	Access Control Entry
ACL	Access Control List
ACP	Access Control Policy
AD	Active Directory
ADAM	Active Directory Application Mode
AES	Advanced Encryption Standard
AGD	Administrator Guidance Document
AH	Authentication Header
ALPC	Advanced Local Process Communication
ANSI	American National Standards Institute
API	Application Programming Interface
APIC	Advanced Programmable Interrupt Controller
BTG	BitLocker To Go
CA	Certificate Authority
CBAC	Claims Basic Access Control, see DYN
CBC	Cipher Block Chaining
CC	Common Criteria
CD-ROM	Compact Disk Read Only Memory
CIFS	Common Internet File System
CIMCPP	Certificate Issuing and Management Components For Basic Robustness Environments Protection Profile, Version 1.0, April 27, 2009
CM	Configuration Management; Control Management
COM	Component Object Model
CP	Content Provider
CPU	Central Processing Unit
CRL	Certificate Revocation List
CryptoAPI	Cryptographic API
CSP	Cryptographic Service Provider
DAC	Discretionary Access Control
DACL	Discretionary Access Control List
DC	Domain Controller
DEP	Data Execution Prevention
DES	Data Encryption Standard
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DFS	Distributed File System
DMA	Direct Memory Access
DNS	Domain Name System
DS	Directory Service
DSA	Digital Signature Algorithm

DYN	Dynamic Access Control
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
EFS	Encrypting File System
ESP	Encapsulating Security Protocol
FEK	File Encryption Key
FIPS	Federal Information Processing Standard
FRS	File Replication Service
FSMO	Flexible Single Master Operation
FTP	File Transfer Protocol
FVE	Full Volume Encryption
GB	Gigabyte
GC	Global Catalog
GHz	Gigahertz
GPC	Group Policy Container
GPO	Group Policy Object
GPOSPP	US Government Protection Profile for General-Purpose Operating System in a Networked Environment
GPT	Group Policy Template
GPT	GUID Partition Table
GUI	Graphical User Interface
GUID	Globally Unique Identifiers
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP
I/O	Input / Output
I&A	Identification and Authentication
IA	Information Assurance
ICF	Internet Connection Firewall
ICMP	Internet Control Message Protocol
ICS	Internet Connection Sharing
ID	Identification
IDE	Integrated Drive Electronics
IETF	Internet Engineering Task Force
IFS	Installable File System
IIS	Internet Information Services
IKE	Internet Key Exchange
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
IPC	Inter-process Communication
IPI	Inter-process Interrupt
IPsec	IP Security
ISAPI	Internet Server API
IT	Information Technology
KDC	Key Distribution Center
LAN	Local Area Network

LDAP	Lightweight Directory Access Protocol
LPC	Local Procedure Call
LSA	Local Security Authority
LSASS	LSA Subsystem Service
LUA	Least-privilege User Account
MAC	Message Authentication Code
MB	Megabyte
MMC	Microsoft Management Console
MSR	Model Specific Register
NAC	(Cisco) Network Admission Control
NAP	Network Access Protection
NAT	Network Address Translation
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NLB	Network Load Balancing
NMI	Non-maskable Interrupt
NTFS	New Technology File System
NTLM	New Technology LAN Manager
OS	Operating System
PAE	Physical Address Extension
PC/SC	Personal Computer/Smart Card
PIN	Personal Identification Number
PKCS	Public Key Certificate Standard
PKI	Public Key Infrastructure
PP	Protection Profile
RADIUS	Remote Authentication Dial In Service
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAS	Remote Access Service
RC4	Rivest's Cipher 4
RID	Relative Identifier
RNG	Random Number Generator
RPC	Remote Procedure Call
RSA	Rivest, Shamir and Adleman
RSASSA	RSA Signature Scheme with Appendix
SA	Security Association
SACL	System Access Control List
SAM	Security Assurance Measure
SAML	Security Assertion Markup Language
SAR	Security Assurance Requirement
SAS	Secure Attention Sequence
SD	Security Descriptor
SHA	Secure Hash Algorithm
SID	Security Identifier
SIP	Session Initiation Protocol
SIPI	Startup IPI

SF	Security Functions
SFP	Security Functional Policy
SFR	Security Functional Requirement
SMB	Server Message Block
SMI	System Management Interrupt
SMTP	Simple Mail Transport Protocol
SP	Service Pack
SPI	Security Parameters Index
SPI	Stateful Packet Inspection
SRM	Security Reference Monitor
SSL	Secure Sockets Layer
SSP	Security Support Providers
SSPI	Security Support Provider Interface
ST	Security Target
SYSVOL	System Volume
TCP	Transmission Control Protocol
TDI	Transport Driver Interface
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSC	TOE Scope of Control
TSF	TOE Security Functions
TSS	TOE Summary Specification
UART	Universal Asynchronous Receiver / Transmitter
UI	User Interface
UID	User Identifier
UNC	Universal Naming Convention
US	United States
UPN	User Principal Name
URL	Uniform Resource Locator
USB	Universal Serial Bus
USN	Update Sequence Number
v5	Version 5
VDS	Virtual Disk Service
VPN	Virtual Private Network
VSS	Volume Shadow Copy Service
WAN	Wide Area Network
WCF	Windows Communications Framework
WebDAV	Web Document Authoring and Versioning
WebSSO	Web Single Sign On
WDM	Windows Driver Model
WIF	Windows Identity Framework
WMI	Windows Management Instrumentation
WSC	Windows Security Center
WU	Windows Update
WSDL	Web Service Description Language

WWW	World-Wide Web
X64	A 64-bit instruction set architecture
X86	A 32-bit instruction set architecture

10 Appendix B: Interfaces and Binaries

This section is a list of Universal Windows Platform (UWP) APIs used during testing of Windows 10.

API	Description
CryptographicBuffer.GenerateRandom	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.cryptographicbuffer.generaterandom.aspx
CryptographicBuffer.GenerateRandomNumber	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.cryptographicbuffer.generaterandomnumber.aspx
CryptographicEngine.Encrypt	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.encrypt.aspx
CryptographicEngine.Decrypt	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.decrypt.aspx
HashAlgorithmProvider.CreateHash	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.hashalgorithmprovider.createhash.aspx
HashAlgorithmProvider.HashData	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.hashalgorithmprovider.hashdata.aspx
CryptographicEngine.Sign	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.sign.aspx
CryptographicEngine.VerifySignature	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.verifysignature.aspx
KeyDerivationParameters.BuildForPbkdf2	http://msdn.microsoft.com/en-us/library/windows/apps/windows.security.cryptography.core.keyderivationparameters.buildforpbkdf2.aspx
AsymmetricKeyAlgorithmProvider.CreateKeyPair	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.asymmetrickeyalgorithmprovider.createkeypair.aspx
CryptographicEngine.SignAsync	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.signasync.aspx
CryptographicEngine.SignHashedData	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.signhasheddata.aspx
CryptographicEngine.SignHashedDataAsync	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.signhasheddataasync.aspx
CryptographicEngine.VerifySignatureWithHashInput	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.core.cryptographicengine.verifysignaturewithhashinput.aspx
AsymmetricKeyAlgorithmPro	http://msdn.microsoft.com/en-

vider.ImportKeyPair	us/library/windows/apps/windows.security.cryptography.core.asymmetrickeyalgorithmprovider.importkeypair.aspx
CertificateEnrollmentManager.ImportPfxDataAsync	http://msdn.microsoft.com/en-us/library/windows/apps/windows.security.cryptography.certificates.certificateenrollmentmanager.importpfxdataasync.aspx
CmsDetachedSignature.GenerateSignatureAsync	http://msdn.microsoft.com/en-us/library/windows/apps/dn298272.aspx
CmsAttachedSignature.GenerateSignatureAsync	http://msdn.microsoft.com/en-us/library/windows/apps/dn298266.aspx
HttpClient	http://msdn.microsoft.com/en-us/library/windows/apps/windows.web.http.httpclient.aspx
Windows.Networking.Vpn	https://msdn.microsoft.com/en-us/library/windows/apps/windows.networking.vpn.aspx
Certificate.BuildChainAsync	http://msdn.microsoft.com/en-us/library/windows/apps/windows.security.cryptography.certificates.certificate.buildchainasync.aspx
CertificateChain.Validate	http://msdn.microsoft.com/en-us/library/windows/apps/dn279161.aspx
Windows.Security.Cryptography.DataProtection	http://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.security.cryptography.dataprotection.aspx

Please send mail to wincc@microsoft.com if you would like a list of the Windows binaries included in this evaluation.