Apple Inc.

Apple iPad and iPhone Mobile Devices with iOS 11.2

PP_MD_V3.1, EP_MDM_AGENT_V3.0, & PP_WLAN_CLI_EP_V1.0

Security Target

Version 1.01
2018-03-30
VID: 10851

Prepared for:
Apple Inc.
One Apple Park Way
MS 927-1CPS
Cupertino, CA 95014
www.apple.com

Prepared by:
atsec information security Corp.
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Table of Contents

Table of Figures

Table of Tables

Revision History

Version	Date	Change
1.0	2018-02-28	Initial Version
1.01	2018-03-30	Addressing validation comments

1 Security Target Introduction

This document is the Common Criteria (CC) Security Target (ST) for the Apple iOS 11.2 operating system on various hardware platforms, listed in section 1.4, *TOE Description*, to be evaluated as Mobile Devices in exact compliance with:

- The Mobile Device Fundamentals Protection Profile Version 3.1, dated 16 June, 2017 [PP_MD_V3.1],
- The Extended Package for Mobile Device Management Agents Version 3.0 [EP_MDM_AGENT_V3.0], dated 21 November, 2016, and
- The General Purpose Operating Systems Protection Profile/ Mobile Device Fundamentals Protection Profile Extended Package (EP) Wireless Local Area Network (WLAN) Clients, Version 1.0, dated 8 February, 2016 [PP_WLAN_CLI_EP_V1.0].

1.1 Security Target Reference

ST Title: Apple iPad and iPhone Mobile Devices with iOS 11.2 PP_MD_V3.1, EP_MDM_AGENT_V3.0, & PP_WLAN_CLI_EP_V1.0 Security Target

ST Version: Version 1.01

ST Date: 2018-03-30

1.2 TOE Reference

Target of Evaluation (TOE) Identification:

- Apple iPad and iPhone Mobile Devices with iOS 11.2.
  Please refer to section 1.4, *TOE Description*, for specific device information.
- The TOE guidance documentation as detailed in section 1.5.3.
- TOE Developer: Apple Inc.
- Evaluation Sponsor: Apple Inc.

1.3 TOE Overview

The TOE is a series of Apple iPad and iPhone mobile devices running the iOS 11.2 operating system, an MDM agent which is included on the mobile devices as an application and a WLAN client application component also executing on the mobile devices.

The operating system manages the device hardware, provides mobile device agent functionality, and provides the technologies required to implement native apps. iOS 11.2 provides a built-in Mobile Device Management (MDM) framework application programmer interface (API), giving management features that may be utilized by external MDM solutions, allowing enterprises to use profiles to control some of the device settings.

iOS 11.2 provides a consistent set of capabilities allowing the supervision of enrolled devices. This includes the preparation of devices for deployment, the subsequent management of the devices, and the termination of management.

The TOE provides cryptographic services for the encryption of data-at rest, for secure communication channels, for protection of configuration profiles, and for use by applications.

User data protection is provided by encrypting the user data, restricting access by applications and by restricting access until the user has been successfully authenticated.

User identification and authentication is provided by a user defined passphrase where the minimum length of the passphrase, passphrase rules, and the maximum number of consecutive failed authentication attempts can be configured by an administrator.

Security management capabilities are provided to users via the user interface of the device and to administrators through the installation of configuration profiles on the device. This installation can be done using the Apple Configurator tool or by using an MDM system.

The TOE protects itself by having its own code and data protected from unauthorized access (using hardware provided memory protection features), by encrypting user and TOE Security Functionality (TSF) data using TSF protected keys and encryption/decryption functions, by self-tests, by ensuring the integrity and authenticity of TSF updates and downloaded applications, and by locking the TOE upon user request or after a defined time of user inactivity.

In addition, the TOE implements a number of cryptographic protocols that can be used to establish a trusted channel to other IT entities.

The MDM Agent provides secure alerts to the MDM Server indicating status events.

1.4 TOE Description

The TOE is a series of Apple iPad and iPhone mobile devices running the iOS 11.2 operating system, an MDM agent which is included on the mobile devices as an application and a WLAN client application component also executing on the mobile devices.

The TOE is intended to be used as a communication solution providing mobile staff connectivity to enterprise data.

The TOE hardware is uniquely identified by the model number (See Table 1: Devices Covered by the Evaluation), and the TOE software is identified by its version number, Apple iOS 11.2. The TOE includes documentation that is listed in section 1.5.3, *TOE Documentation*.

The TOE provides wireless connectivity and includes software for VPN connection, for access to the protected enterprise network, enterprise data and applications, and for communicating with other Mobile Devices. The software for the VPN connection is evaluated separately.

The TOE does not include the user applications that run on top of the operating system but does include controls that limit application behavior. The TOE may be used as a mobile device within an enterprise environment where the configuration of the device is managed through an evaluated MDM solution.

The normal distribution channels for a regular end user to obtain these hardware devices include the following.

- The Apple Store (either a physical store or online at https://apple.com)
- Apple retailers
- Service carriers (e.g., AT&T, Verizon)
- Resellers

Business

There is a distinct online store for Business customers with a link from the “Apple Store.” From the link to the “Apple Store” (http://www.apple.com), go to the upper left of the page and click “Business Store Home.” Or optionally, use the following link.

http://www.apple.com/us_smb_78313/shop

Government

Government customers can use the following link.

http://www.apple.com/r/store/government/

Additional

Large customers can also have their own Apple Store Catalog for their employees to purchase devices directly from Apple under their corporate employee purchase program.

Software updates are obtained through the Apple push service.

The TOE provides an interface allowing the enterprise to supervise devices under their control.

At the highest level, the operating system part of the TOE acts as an intermediary between the underlying hardware and the apps operating on the TOE. Apps do not talk to the underlying hardware directly. Instead, they communicate with the hardware through a set of well-defined system interfaces. These interfaces make it easy to write apps that work consistently on devices having different hardware capabilities.

The TOE needs to be configured by an authorized administrator to operate in compliance with the requirements defined in this [ST]. The evaluated configuration for this includes:

- the requirement to define a passcode for user authentication,
- the specification of a passcode policy defining criteria on the minimum length and complexity of a passcode,
- the specification of the maximum number of consecutive failed attempts to enter the passcode,
- the specification of the session locking policy,
- the specification of the audio and video collection devices allowed,
- the specification of the virtual private network (VPN) connection,
- the specification of the wireless networks allowed, and
- the requirement of the certificates in the trust anchor database.

The following tables lists the devices that are covered by this evaluation and give the technical characteristics of each.

Device Name	Model Number	Processor	Wi-Fi	Cellular	Blue-tooth	BAF
iPhone 5s	A1533 (GSM) A1533 (CDMA) A1453 A1457 A1530	A7	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac 802.11/a/b/g/n/ac 802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	See table 2 See table 2 See table 2 See table 2 See table 2	4.0 4.0 4.0 4.0 4.0	Touch ID 1 Touch ID 1 Touch ID 1 Touch ID 1 Touch ID 1
iPhone 6 Plus/ iPhone 6	A1549/A1522 (GSM) A1549/A1522 (CDMA) A1586/A1524	A8	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	See table 2 See table 2 See table 2	4.0 4.0 4.0	Touch ID 1 Touch ID 1 Touch ID 1
iPhone 6s Plus iPhone 6s	A1634/A1633 (US) A1687/A1688 (Global)	A9	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	See table 2 See table 2	4.2 4.2	Touch ID 2 Touch ID 2
iPhone 7 Plus/ iPhone 7	A1784/A1778 (GSM) A1661/A1660 (CDMA)	A10 Fusion	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	See table 2 See table 2	4.2 4.2	Touch ID 3 Touch ID 3
iPhone 8 Plus iPhone 8	A1864/A1898/A1899/A1897 A1863/A1906/A1907/A1905	A11 Bionic	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	See table 2 See table 2	5.0 5.0	Touch ID 3 Touch ID 3
iPhone X	A1865/A1902 A1901	A11 Bionic	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	See table 2 See table 2	5.0 5.0	Face ID Face ID
iPhone SE	A1662 (US) A1723 (Global)	A9	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	See table 2 See table 2	4.2 4.2	Touch ID 1 Touch ID 1
iPad mini 3	A1599 (Wi-Fi only) A1600 (Wi-Fi + cellular) A1601 (Wi-Fi + cellular)	A7	802.11a/b/g/n 802.11a/b/g/n 802.11a/b/g/n	- See table 2 See table 2	4.0 4.0 4.0	Touch ID 1 Touch ID 1 Touch ID 1
iPad mini 4	A1538 (Wi-Fi only) A1550 (Wi-Fi + cellular)	A8	802.11a/b/g/n 802.11a/b/g/n	- See table 2	4.2 4.2	Touch ID 1 Touch ID 1
iPad Air 2	A1566 (Wi-Fi only) A1567 (Wi-Fi + cellular)	A8X	802.11a/b/g/n/ac 802.11a/b/g/n/ac	- See table 2	4.2 4.2	Touch ID 1 Touch ID 1
iPad Pro 12.9”	A1584 (Wi-Fi only) A1652 (Wi-Fi + cellular)	A9X	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	- See table 2	4.2 4.2	Touch ID 1 Touch ID 1
iPad Pro 9.7”	A1673 (Wi-Fi only) A1674 (Wi-Fi + cellular)	A9X	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	- See table 2	4.2 4.2	Touch ID 1 Touch ID 1
iPad	A1822 (Wi-Fi only) A1823 (Wi-Fi + cellular)	A9	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	- See table 2	4.2 4.2	Touch ID 1 Touch ID 1
iPad Pro 12.9”	A1670 (Wi-Fi) A1671 (Wi-Fi + Cellular)	A10X Fusion	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	- See table 2	4.2 4.2	Touch ID 3 Touch ID 3
iPad Pro 10.5”	A1701 (Wi-Fi) A1709 (Wi-Fi + Cellular)	A10X Fusion	802.11/a/b/g/n/ac 802.11/a/b/g/n/ac	- See table 2	4.2 4.2	Touch ID 3 Touch ID 3

Table 1: Devices Covered by the Evaluation

Device Name	Model Number	Cellular

iPhone 5s	A1533 (GSM)	UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz);
		GSM/EDGE (850, 900, 1800, 1900 MHz);
		LTE (Bands 1, 2, 3, 4, 5, 8, 13, 17, 19, 20, 25)

	A1533 (CDMA)	CDMA EV-DO Rev. A and Rev. B (800, 1700/2100, 1900, 2100 MHz);
		UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz);
		GSM/EDGE (850, 900, 1800, 1900 MHz);
		LTE (Bands 1, 2, 3, 4, 5, 8, 13, 17, 19, 20, 25)

	A1453	CDMA EV-DO Rev. A and Rev. B (800, 1700/2100, 1900, 2100 MHz);
		UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz);
		GSM/EDGE (850, 900, 1800, 1900 MHz);
		LTE (Bands 1, 2, 3, 4, 5, 8, 13, 17, 18, 19, 20, 25, 26)

	A1457	UMTS/HSPA+/DC-HSDPA (850, 900, 1900, 2100 MHz);
		GSM/EDGE (850, 900, 1800, 1900 MHz);
		LTE (Bands 1, 2, 3, 5, 7, 8, 20)

	A1530	UMTS/HSPA+/DC-HSDPA (850, 900, 1900, 2100 MHz);
		GSM/EDGE (850, 900, 1800, 1900 MHz);
		FDD-LTE (Bands 1, 2, 3, 5, 7, 8, 20);
		TD-LTE (Bands 38, 39, 40)

iPhone 6 Plus/ iPhone 6	A1549/A1522 (GSM)	UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 13, 17, 18, 19, 20, 25, 26, 28, 29)
	A1549/A1522 (CDMA)	CDMA EV-DO Rev. A and Rev. B (800, 1700/2100, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 13, 17, 18, 19, 20, 25, 26, 28, 29)
	A1586/A1524	CDMA EV-DO Rev. A and Rev. B (800, 1700/2100, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) TD-SCDMA 1900 (F), 2000 (A) GSM/EDGE (850, 900, 1800, 1900 MHz) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 13, 17, 18, 19, 20, 25, 26, 28, 29) TD-LTE (Bands 38, 39, 40, 41)
iPhone 6s Plus/ iPhone 6s	A1634/A1633 (US)	CDMA EV-DO Rev. A (800, 1700/2100, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) TD-SCDMA 1900 (F), 2000 (A) GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 27, 28, 29, 30) TD-LTE (Bands 38, 39, 40, 41)

Device Name	Model Number	Cellular
	A1687/A1688 (Global)	CDMA EV-DO Rev. A (800, 1700/2100, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) TD-SCDMA 1900 (F), 2000 (A) GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 27, 28, 29) TD-LTE (Bands 38, 39, 40, 41)
iPhone 7 Plus/ iPhone 7	A1784/A1778 (GSM)	UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 27, 28, 29, 30) TD-LTE (Bands 38, 39, 40, 41)
	A1661/A1660 (CDMA)	CDMA EV-DO Rev. A (800, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) TD-SCDMA 1900 (F), 2000 (A) GSM/EDGE (850, 900, 1800, 1900 MHz) FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 27, 28, 29, 30) TD-LTE (Bands 38, 39, 40, 41)
iPhone 8 Plus iPhone 8	A1863/A1864	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 29, 30, 66) TD-LTE (Bands 34, 38, 39, 40, 41) TD-SCDMA 1900 (F), 2000 (A) CDMA EV-DO Rev. A (800, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	A1897	Models A1905 and A1897 do not support CDMA networks, such as those used by Verizon and Sprint. FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 29, 30, 66) TD-LTE (Bands 34, 38, 39, 40, 41) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
	A1905	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 29, 30, 66) TD-LTE (Bands 34, 38, 39, 40, 41) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
iPhone X	A1865/A1902	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 29, 30, 66) TD-LTE (Bands 34, 38, 39, 40, 41) TD-SCDMA 1900 (F), 2000 (A) CDMA EV-DO Rev. A (800, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)

Device Name	Model Number	Cellular
	A1901	FDD-LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 29, 30, 66) TD-LTE (Bands 34, 38, 39, 40, 41) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz)
iPhone SE	A1662 (US)	CDMA EV-DO Rev. A (800, 1700/2100, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 8, 12, 13, 17, 18, 19, 20, 25, 26, 29)
	A1723 (Global)	CDMA EV-DO Rev. A (800, 1700/2100, 1900, 2100 MHz) UMTS/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz) TD-SCDMA 1900 (F), 2000 (A) GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 17, 18, 19, 20, 25, 26, 28) TD-LTE (Bands 38, 39, 40, 41)
iPad mini 3	A1600	UMTS/HSPA/HSPA+/DC‑HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) CDMA EV-DO Rev. A and Rev. B (800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 13, 17, 18, 19, 20, 25, 26)
	A1601	UMTS/HSPA/HSPA+/DC‑HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) CDMA EV-DO Rev. A (800, 1900 MHz) TD-SCDMA (1900 (F), 2000 (A)) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 18, 19, 20) TD-LTE (Bands 38, 39, 40)
iPad mini 4	A 1550	UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) CDMA EV-DO Rev. A and Rev. B (800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 13, 17, 18, 19, 20, 25, 26, 28, 29, 38, 39, 40, 41)
iPad Air 2	A1567	GSM/EDGE (850, 900, 1800, 1900 MHz), UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz), CDMA EV-DO Rev. A and Rev. B (800, 1900 MHz), TD-SCDMA LTE (Bands 1, 2, 3, 4, 5, 7, 8, 13, 17,18, 19, 20, 25, 26, 28, 29) TD-LTE (Bands 38, 39, 40,41)
iPad Pro 12.9”	A1652	CDMA EV-DO Rev. A and Rev. B (800, 1900 MHz) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 13, 17, 18, 19, 20, 25, 26, 28, 29, 38, 39, 40, 41)

Device Name	Model Number	Cellular
iPad Pro 9.7”	A1674	CDMA EV-DO Rev. A and Rev. B (800, 1900 MHz) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) LTE Advanced (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 27, 28, 29, 30, 38, 39, 40, 41)
iPad	A1823	CDMA EV-DO Rev. A and Rev. B UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) LTE (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 29, 38, 39, 40, 41)
iPad Pro 12.9”	A1671	CDMA EV-DO Rev. A and Rev. B (800, 1900 MHz) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) LTE Advanced (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 27, 28, 29, 30, 38, 39, 40, 41)
iPad Pro 10.5”	A1709	CDMA EV-DO Rev. A and Rev. B (800, 1900 MHz) UMTS/HSPA/HSPA+/DC-HSDPA (850, 900, 1700/2100, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz) LTE Advanced (Bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 27, 28, 29, 30, 38, 39, 40, 41)

Table 2: Cellular Protocols Supported

1.5 TOE Architecture

The implementation of TOE architecture can be viewed as a set of layers, which are shown in Figure 1. Lower layers contain fundamental services and technologies. Higher-level layers build upon the lower layers and provide more sophisticated services and technologies.

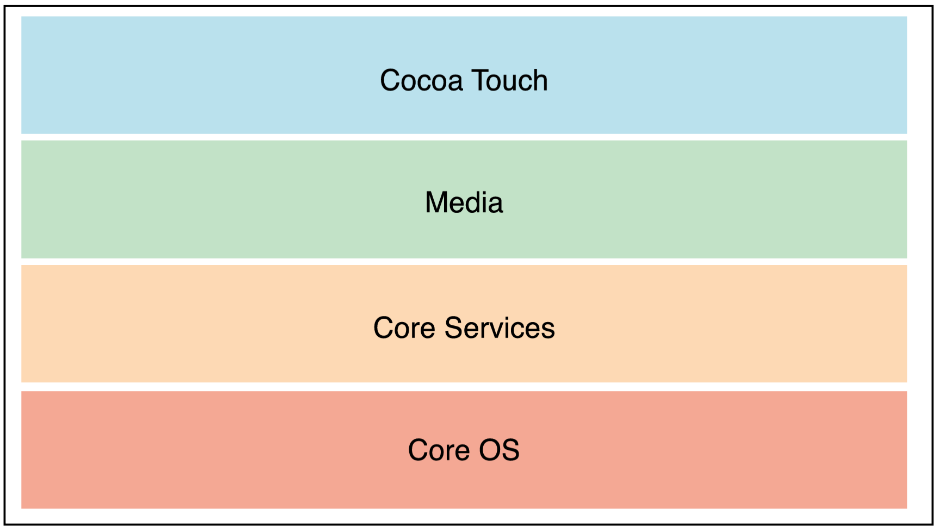

Figure 1: Layers of iOS

The individual layers provide the following services.

The **Cocoa Touch layer** contains key frameworks for building iOS apps. These frameworks define the appearance of applications (apps). They also provide the basic app infrastructure and support for key technologies such as multitasking, touch-based input, push notifications, and many high-level system services. When designing apps, one should investigate the technologies in this layer first to see if they meet the needs of the developer.

The **Media layer** contains the graphics, audio, and video technologies you use to implement multimedia experiences in apps. The technologies in this layer make it easy to build apps that look and sound great.

The **Core Services layer** contains fundamental system services for apps. Key among these services are the Core Foundation and Foundation frameworks, which define the basic types that all apps use. This layer also contains individual technologies to support features such as location, iCloud, social media, and networking.

This layer also implements data protection functions that allow apps that work with sensitive user data to take advantage of the built-in encryption available on some devices. When an app designates a specific file as protected, the system stores that file in an encrypted format. While the device is locked, the contents of the file are inaccessible to both the app and to any potential intruders. However, when the device is unlocked by the user, a decryption key is created to allow the app to access the file. Other levels of data protection are also available.

The **Core OS layer** contains the low-level features that most other technologies are built upon. Even if an app does not use these technologies directly, they are most likely being used by other frameworks. And in situations where an app needs to explicitly deal with security or communicating with an external hardware accessory, it does so by using the frameworks in this layer.

Security related frameworks provided by this layer are:

- the Generic Security Services Framework, providing services as specified in Request for Comment (RFC) 2743 (Generic Security Service Application Program Interface Version 2, Update 1) and RFC 4401 (Pseudo Random Function);
- the Local Authentication Framework;
- the Network Extension Framework, providing support for configuring and controlling VPN tunnels;
- the Security Framework, providing services to manage and store certificates, public and private keys, and trust policies (this framework also provides the Common Crypto library for symmetric encryption and hash-based message authentication codes); and
- the System Framework, providing the kernel environment, drivers, and low-level UNIX interfaces (the kernel manages the virtual memory system, threads, file system, network, and inter-process communication and is therefore responsible for separating apps from each other and controlling the use of low-level resources).

The TOE may be managed by an MDM solution that enables an enterprise to control and administer the TOE instances that are enrolled in the MDM solution.

1.5.1 Physical Boundaries

The TOE's physical boundaries are those of the Mobile Devices.

1.5.2 Security Functions provided by the TOE

The TOE provides the security functionality required by [PP_MD_V3.1], [EP_MDM_AGENT_V3.0] and [PP_WLAN_CLI_EP_V1.0].

1.5.2.1 Cryptographic Support

The TOE provides cryptographic services via the following two cryptographic modules.

- Apple CoreCrypto Cryptographic Module v8 for ARM
- Apple CoreCrypto Kernel Cryptographic Module v8 for ARM

The Apple CoreCrypto Kernel Module v8 for ARM is an iOS kernel extension optimized for library use within the iOS kernel. Once the module is loaded into the iOS kernel its cryptographic functions are made available to iOS Kernel services only.

The following figure shows the boundary of the Apple CoreCrypto Kernel Module v8 for ARM within the TOE.

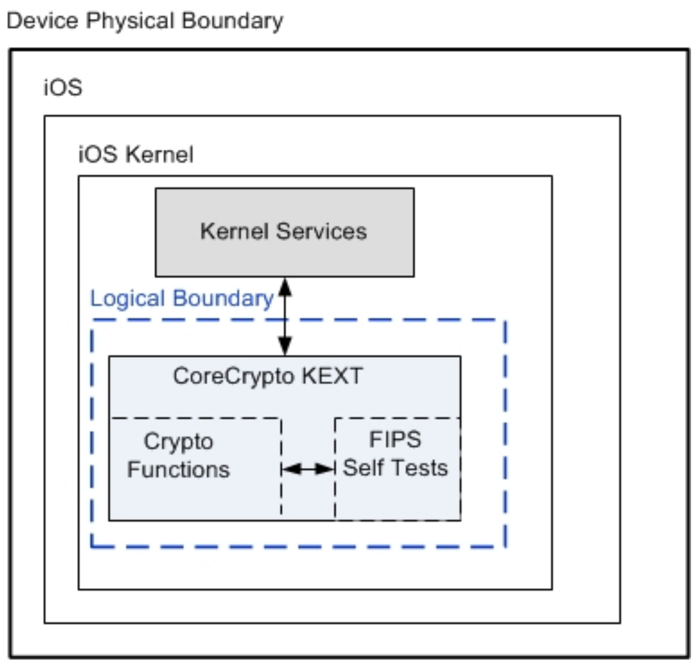

Figure 2: Logical Block Diagram of the Apple CoreCrypto Kernel Module v8 for ARM

The Apple CoreCrypto Module v8 for ARM is designed for library use within the iOS user space. It is implemented as an iOS dynamically loadable library. The dynamically loadable library is loaded into the iOS application and its cryptographic functions are made available to the application. A second instance of this module is used within the secure enclave to provide cryptographic services there.

The following figure shows the boundary of Apple CoreCrypto Module v8 for ARM e within the TOE.

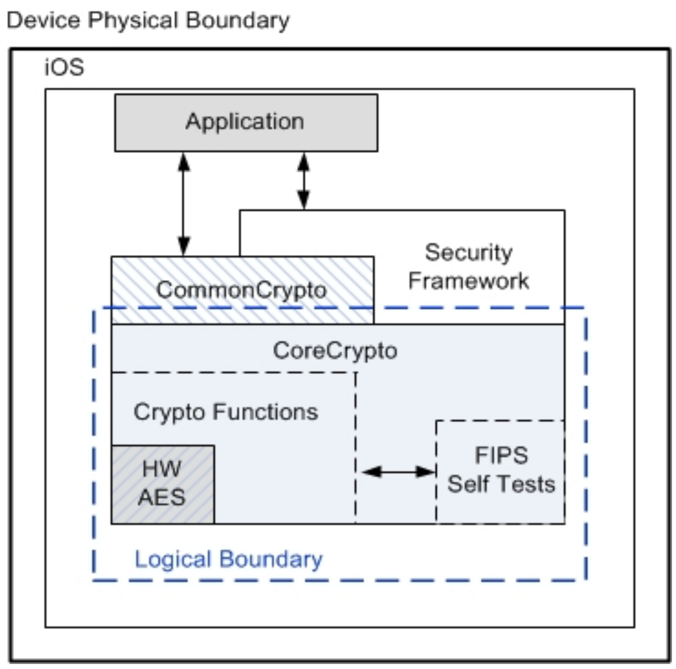

Figure 3: Logical Block Diagram of the Apple CoreCrypto Module v8 for ARM.

The cryptographic functions provided include symmetric key generation, encryption and decryption using the Advanced Encryption Standard (AES) algorithms, asymmetric key generation and key establishment, cryptographic hashing, and keyed-hash message authentication.

Note that the Wi-Fi chip, performs the AES functions.

The functions listed below are used to implement the security protocols supported as well as for the encryption of data-at-rest.

- Random Number Generation; Symmetric Key Generation
    - SP 800-90A Deterministic Random Bit Generator (DRBG)
        - CTR_DRBG with AES 128 core, with derivation function and without prediction resistance
        - CTR_DRBG with AES 256 core without derivation function and without prediction resistance
- Symmetric Encryption and Decryption
    - FIPS 197 AES, SP 800-38 A, SP 800-38 C, SP 800-38 SP 800-38 F
        - Key sizes: 128 /256 bits
          Block chaining modes: CBC, CCM, GCM, KW
- Digital Signature and Asymmetric Key Generation
    - FIPS186-4 RSA, PKCS #1.5
        - KEY(gen)(2048/3072), SigGenPKCS1.5 (2048/3072), SigVerPKCS1.5 (1024/2048/3072)
    - FIPS 186-4 ECDSA, ANSI X9.62
        - PKG: curves P-256, P-384
        - PKV: curves P-256, P-384
        - SIG(gen): curves P-256, P-384
        - SIG(ver): curves P-256, P-384
- Message Digest
    - FIPS 180-4 SHS
        - SHA-1, SHA-2(256, 384, 512)
- Keyed Hash
    - FIPS 198 HMAC
        - SHA-1, SHA-2(256, 384, 512)
- PBKDF
    - SP 800-132
        - Password based key derivation using HMAC with SHA-2 as pseudorandom function
- EC Diffie-Hellman
    - Curve25519

1.5.2.2 User Data Protection

User data in files is protected using cryptographic functions, ensuring this data remains protected even if the device gets lost or is stolen. Critical data, like passcodes used by applications or application defined cryptographic keys, can be stored in the key chain, which provides additional protection. Passcode protection and encryption ensure that data-at-rest remains protected even in the case of the device being lost or stolen.

The Secure Enclave Processor (SEP), a separate CPU that executes a stand-alone operating system and has separate memory, provides protection for critical security data such as keys.

Data can also be protected such that only the application that owns the data can access it.

1.5.2.3 Identification and Authentication

Except for making emergency calls, using the cameras, and using the flashlight, users need to authenticate using a passcode or a biometric (fingerprint or face). On power up, or after an update of iOS the user is required to use the passcode authentication mechanism.

The passcode can be configured for a minimum length, for dedicated passcode policies, and for a maximum life time. When entered, passcodes are obscured and the frequency of entering passcodes is limited as well as the number of consecutive failed attempts of entering the passcode.

The TOE also enters a locked state after a (configurable) time of user inactivity and the user is required to either enter his passcode or use biometric authentication (fingerprint or face) to unlock the TOE.

External entities connecting to the TOE via a secure protocol (Extensible Authentication Protocol Transport Layer Security (EAP-TLS), Transport Layer Security (TLS), IPsec) can be authenticated using X.509 certificates.

1.5.2.4 Security Management

The security functions listed in Table 5: Management Functions, can be managed either by the user or by an authorized administrator through an MDM system. This table identifies the functions that can be managed and indicates if the management can be performed by the user, by the authorized administrator, or both.

1.5.2.5 Protection of the TSF

Some of the functions the TOE implements to protect the TSF and TSF data are as follows.

- Protection of cryptographic keys—keys used for TOE internal key wrapping and for the protection of data-at-rest are not exportable. There are provisions for fast and secure wiping of key material.
- Use of memory protection and processor states to separate applications and protect the TSF from unauthorized access to TSF resources—in addition, each device includes a separate system called the "secure enclave" which is the only system that can use the Root Encryption Key (REK). The secure enclave is a separate CPU that executes a stand-alone operating system and has separate memory.
- Digital signature protection of the TSF image—all updates to the TSF need to be digitally signed.
- Software/firmware integrity self-test upon start-up—the TOE will not go operational when this test fails.
- Digital signature verification for applications
- Access to defined TSF data and TSF services only when the TOE is unlocked

1.5.2.6 TOE Access

The TSF provides functions to lock the TOE upon request and after an administrator-configurable time of inactivity.

Access to the TOE via a wireless network is controlled by user/administrator defined policy.

1.5.2.7 Trusted Path/Channels

The TOE supports the use of the following cryptographic protocols that define a trusted channel between itself and another trusted IT product.

- IEEE 802.11-2012
- IEEE 802.1X
- EAP-TLS
- TLS
- IPsec (addressed in a separate evaluation)

1.5.2.8 Audit

The TOE provides the ability for responses to be sent from the MDM Device Agent to the MDM Server. These responses are configurable by the organization using a scripting language given in the Over-the-Air Profile Delivery and Configuration document.

1.5.3 TOE Documentation

User Guidance	
[iPhone_UG] iPhone User Guide for iOS 11.2 (2017)	https://help.apple.com/iphone/11/
[iPad_UG] iPad User Guide for iOS 11.2 (2017)	https://help.apple.com/ipad/11/
Administrator Guidance	
[CC_GUIDE] Apple iPad and iPhone Mobile Devices with iOS 11.2 PP_MD_V3.1, EP_MDM_AGENT_V3.0, & PP_WLAN_CLI_EP_V1.0 Common Criteria Guide	(This document.) https://www.niap-ccevs.org/st/st_vid10851-agd.pdf
Supporting Documents	
[iOSDeployRef] iOS Deployment Reference (V3.9)	https://itunes.apple.com/us/book/ios-deployment-reference/id917468024?mt=11
[OTA_Guide] Over-The-Air Profile Delivery and Configuration Guide (Updated 2018-01-24)	https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/iPhoneOTAConfiguration/Introduction/Introduction.html
[IOS_CFG] Configuration Profile Reference (Updated 2018-01-24)	https://developer.apple.com/enterprise/ConfigurationProfileReference.pdf
[AConfig] Apple Configurator 2 Help (online guidance)	http://help.apple.com/configurator/mac/2.6//

[DEP_Guide] Apple Deployment Programs Device Enrollment Program Guide	https://www.apple.com/business/docs/DEP_Guide.pdf
[PM_Help] Profile Manager Help	https://help.apple.com/profilemanager/mac/5.4/
[IOS_LOGS] Profiles and Logs	https://developer.apple.com/bug-reporting/profiles-and-logs/?platforms=ios
App Developer Guidance	
[3CC-MAN] Common Crypto man pages	https://developer.apple.com/legacy/library/documentation/Darwin/Reference/ManPages/#
[CKTSREF] Certificate, Key, and Trust Services	https://developer.apple.com/documentation/security/certificate_key_and_trust_services
[CRYPTOGUIDE] Cryptographic Services Guide	https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/Introduction/Introduction.html
[iOS_MDM] Mobile Device Management Protocol Reference	https://developer.apple.com/library/content/documentation/Miscellaneous/Reference/MobileDeviceManagementProtocolRef/1-Introduction/Introduction.html
[IPLKEYREF] Information Property List Key Reference	https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Introduction/Introduction.html
[KEYCHAINPG] Keychain Services Programming Guide	https://developer.apple.com/library/ios/documentation/Security/Conceptual/keychainServConcepts/01introduction/introduction.html
[SECFWREF] Secure Framework	https://developer.apple.com/library/prerelease/ios/documentation/Security/Reference/SecurityFrameworkReference/index.html
[HTTPSTN2232] Technical Note TN 2232: HTTPS Server Trust Evaluation	https://developer.apple.com/library/ios/technotes/tn2232/_index.html

1.5.4 Other References

[BT] Specification of the Bluetooth System
https://www.bluetooth.com/specifications/adopted-specifications

[PP_MD_V3.1] U.S. Government Approved Protection Profile - Protection Profile for Mobile Device Fundamentals, Version 3.1
https://www.niap-ccevs.org/Profile/Info.cfm?id=417

[EP_MDM_AGENT_V3.0] U.S. Government Approved Protection Profile - Extended Package for Mobile Device Management Agents Version 3.0
https://www.niap-ccevs.org/Profile/Info.cfm?id=403

[PP_WLAN_CLI_EP_V1.0] Extended Package for WLAN Client Version 1.0
https://www.niap-ccevs.org/Profile/Info.cfm?id=386

[FACE_SEC] Face ID Security (November 2017)
https://images.apple.com/business/docs/FaceID_Security_Guide.pdf

[iOS_TEC] iOS Technology Overview
https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

[CORECRYPTO_SECPOL]
– Apple CoreCrypto Cryptographic Module v8.0 for ARM. FIPS 140-2 Non-

Proprietary Security Policy (February 2018)
https://csrc.nist.gov/projects/cryptographic-module-validation-program/Certificate/3148

and

– Apple CoreCrypto Kernel Cryptographic Module v8.0 for ARM. FIPS 140-2 Non-Proprietary Security Policy (February 2018)
https://csrc.nist.gov/projects/cryptographic-module-validation-program/Certificate/3147

2 Conformance Claims

2.1 CC Conformance

This [ST] is conformant to the following CC documents.

- Common Criteria for Information Technology Security Evaluations Part 1, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluations Part 2, Version 3.1, Revision 5, April 2017: Part 2 extended
- Common Criteria for Information Technology Security Evaluations Part 3, Version 3.1, Revision 5, April 2017: Part 3 extended

2.2 Protection Profile (PP) Conformance

This [ST] is conformant to the following PPs.

- Protection Profile for Mobile Device Fundamentals, Version 3.1, dated 16 June, 2017 [PP_MD_V3.1]
- Extended Package for Mobile Device Management Agents Version 3.0, dated 21 November, 2016 [EP_MDM_AGENT_V3.0]
- General Purpose Operating Systems Protection Profile/ Mobile Device Fundamentals Protection Profile Extended Package (EP) Wireless Local Area Network (WLAN) Clients, dated 8 February, 2016 [PP_WLAN_CLI_EP_V1.0]

Note: This ST matches the use case 4 templates described in [PP_MD_V3.1] and [EP_MDM_AGENT_V3.0].

2.2.1 Technical Decisions

The following technical decisions were found to be applicable to the ST:

[TD0244] FCS_TLSC_EXT – TLS Client Curves Allowed
https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=250

[TD0237] FAU_GEN.1.1(2) - FMT_UNR_EXT.1 Audit Record Selection-Based
https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=243

[TD0236] FCS_TLSC_EXT.2.1 – TLS Client Curve Limitation
https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=242

[TD0194] Update to audit of FTP_ITC_EXT.1/WLAN
https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=198

2.3 Conformance Rationale

This [ST] provides exact compliance with version 3.1 of the Mobile Device Fundamentals Protection Profile [PP_MD_V3.1] with the Extended Package for MDM Agents [EP_MDM_AGENT_V3.0], and for the WLAN client [PP_WLAN_CLI_EP_V1.0]. The security problem definition, security objectives and security requirements in this [ST] are all taken from the PP and the Extended Packages (EP) performing only operations defined there.

The requirements in the PPs are assumed to represent a complete set of requirements that serve to address any interdependencies. Given that all of the appropriate functional requirements given in the PPs have been copied into this [ST], the dependency analysis for the requirements is assumed to be already performed by the PP authors and is not reproduced in this document.

3 Security Problem Definition

The security problem definition has been taken from [PP_MD_V3.1], [EP_MDM_AGENT_V3.0] and [PP_WLAN_CLI_EP_V1.0]. It is reproduced here for the convenience of the reader.

3.1 Threats

T.EAVESDROP **Network Eavesdropping** (PP_MD_V3.1)

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

T.NETWORK_EAVESDROP Network Eavesdropping (EP_MDM_AGENT_V3.0)

Unauthorized entities may intercept communications between the MDM and mobile devices to monitor, gain access to, disclose, or alter remote management commands. Unauthorized entities may intercept unprotected wireless communications between the mobile device and the Enterprise to monitor, gain access to, disclose, or alter TOE data.

T.NETWORK Network Attack (PP_MD_V3.1)

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

T.NETWORK_ATTACK Network Attack (EP_MDM_AGENT_V3.0)

An attacker may masquerade as MDM Server and attempt to compromise the integrity of the mobile device by sending malicious management commands.

T.PHYSICAL Physical Access (PP_MD_V3.1)

An attacker, with physical access, may attempt to access user data on the Mobile Device including credentials. These physical access threats may involve attacks, which attempt to access the device through external hardware ports, impersonate the user authentication mechanisms, through its user interface, and also through direct and possibly destructive access to its storage media.

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

T.PHYSICAL_ACCESS Physical Access (EP_MDM_AGENT_V3.0)

The mobile device may be lost or stolen, and an unauthorized individual may attempt to access user data. Although these attacks are primarily directed against the mobile device platform, the MDM Agent configures features, which address this threat.

T.FLAWAPP Malicious or Flawed Application (PP_MD_V3.1)

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

T.MALICIOUS_APPS Malicious and Flawed Application (EP_MDM_AGENT_V3.0)

Malicious or flawed application threats exist because apps loaded onto a mobile device may include malicious or exploitable code. An administrator of the MDM or mobile device user may inadvertently import malicious code, or an attacker may insert malicious code into the TOE, resulting in the compromise of TOE or TOE data.

T.PERSISTENT Persistent Presence (PP_MD_V3.1)

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

T.BACKUP (EP_MDM_AGENT_V3.0)

An attacker may try to target backups of data or credentials and exfiltrate data. Since the backup is stored on either a personal computer or end user’s backup repository, it’s not likely enterprise would detect compromise.

T.TSF_FAILURE (PP_WLAN_CLI_EP_V1.0)

Security mechanisms of the TOE generally build up from a primitive set of mechanisms (e.g., memory management, privileged modes of process execution) to more complex sets of mechanisms. Failure of the primitive mechanisms could lead to a compromise in more complex mechanisms, resulting in a compromise of the TSF.

T.UNAUTHORIZED ACCESS (PP_WLAN_CLI_EP_V1.0)

A user may gain unauthorized access to the TOE data and TOE executable code. A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to data or TOE resources. A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.

T.UNDETECTED ACTIONS (PP_WLAN_CLI_EP_V1.0)

Malicious remote users or external IT entities may take actions that adversely affect the security of the TOE. These actions may remain undetected and thus their effects cannot be effectively mitigated.

3.2 Assumptions

A.CONFIG (PP_MD_V3.1)

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 (PP_MD_V3.1)

It is assumed that the mobile user will immediately notify the administrator if the Mobile Device is lost or stolen.

A.PRECAUTION (PP_MD_V3.1)

It is assumed that the mobile user exercises precautions to reduce the risk of loss or theft of the Mobile Device.

A.CONNNECTIVITY (EP_MDM_AGENT_V3.0)

The TOE relies on network connectivity to carry out its management activities. The TOE will robustly handle instances when connectivity is unavailable or unreliable.

A.MOBILE_DEVICE_PLATFORM (EP_MDM_AGENT_V3.0)

The MDM Agent relies upon Mobile platforms and hardware evaluated against the MDFPP and assured to provide policy enforcement as well as cryptographic services and data protection. The Mobile platform provides trusted updates and software integrity verification of the MDM Agent.

A.PROPER_ADMIN (EP_MDM_AGENT_V3.0)

One or more competent, trusted personnel who are not careless, willfully negligent, or hostile, are assigned and authorized as the TOE Administrators, and do so using and abiding by guidance documentation.

A.PROPER_USER (EP_MDM_AGENT_V3.0)

Mobile device users are not willfully negligent or hostile, and use the device within compliance of a reasonable Enterprise security policy.

A.NO_TOE_BYPASS (PP_WLAN_CLI_EP_V1.0)

Information cannot flow between the wireless client and the internal wired network without passing through the TOE.

A.TRUSTED_ADMIN (PP_WLAN_CLI_EP_V1.0)

TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

3.3 Organizational Security Policies

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

P.ADMIN (EP_MDM_AGENT_V3.0)

The configuration of the mobile device security functions must adhere to the Enterprise security policy.

P.DEVICE_ENROLL (EP_MDM_AGENT_V3.0)

A mobile device must be enrolled for a specific user by the administrator of the MDM prior to being used in the Enterprise network by the user.

P.NOTIFY (EP_MDM_AGENT_V3.0)

The mobile user must immediately notify the administrator if a mobile device is lost or stolen so that the administrator may apply remediation actions via the MDM system.

P.ACCOUNTABILITY (EP_MDM_AGENT_V3.0)

Personnel operating the TOE shall be accountable for their actions within the TOE.

4 Security Objectives

The security objectives have been taken from [PP_MD_V3.1], [EP_MDM_AGENT_V3.0], and [PP_WLAN_CLI_EP_V1.0]. They are reproduced here for the convenience of the reader.

4.1 Security Objectives for the TOE

O.COMMS Protected Communications (PP_MD_V3.1)

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

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

O.STORAGE Protected Storage (PP_MD_V3.1)

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

O.CONFIG Mobile Device Configuration (PP_MD_V3.1)

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

O.AUTH Authorization and Authentication (PP_MD_V3.1)

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

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

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

O.INTEGRITY Mobile Device Integrity (PP_MD_V3.1)

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

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

O.PRIVACY End User Privacy and Device Functionality (PP_MD_V3.1)

In a BYOD environment (use cases 3 and 4), a personally-owned mobile device is used for both personal activities and enterprise data. Enterprise management solutions may have the technical capability to monitor and enforce security policies on the device. However, the privacy of the personal activities and data must be ensured. In addition, since there are limited controls that the enterprise can enforce on the personal side, separation of personal and enterprise data is needed. This will protect against the T.FLAWAPP and T.PERSISTENT threats.

O. APPLY_POLICY (EP_MDM_AGENT_V3.0)

The TOE must facilitate configuration and enforcement of enterprise security policies on mobile devices via interaction with the mobile OS and the MDM Server. This will include the initial enrollment of the device into management, through its lifecycle including policy updates and through its possible unenrollment from management services.

O.ACCOUNTABILITY (EP_MDM_AGENT_V3.0)

The TOE must provide logging facilities which record management actions undertaken by its administrators.

O. DATA_PROTECTION_TRANSIT (EP_MDM_AGENT_V3.0)

Data exchanged between the MDM Server and the MDM Agent must be protected from being monitored, accessed, or altered.

O.AUTH_COMM (PP_WLAN_CLI_EP_V1.0)

The TOE will provide a means to ensure that it is communicating with an authorized Access Point and not some other entity pretending to be an authorized Access Point and will provide assurance to the Access Point of its identity.

O.CRYPTOGRAPHIC_FUNCTIONS (PP_WLAN_CLI_EP_V1.0)

The TOE shall provide or use cryptographic functions (i.e., encryption/decryption and digital signature operations) to maintain the confidentiality and allow for detection of modification of data that are transmitted outside the TOE and its host environment.

O.SYSTEM_MONITORING (PP_WLAN_CLI_EP_V1.0)

The TOE will provide the capability to generate audit data.

O.TOE_ADMINISTRATION (PP_WLAN_CLI_EP_V1.0)

The TOE will provide mechanisms to allow administrators to be able to configure the TOE.

O.TSF_SELF_TEST (PP_WLAN_CLI_EP_V1.0)

The TOE will provide the capability to test some subset of its security functionality to ensure it is operating properly.

O.WIRELESS_ACCESS_POINT_CONNECTION (PP_WLAN_CLI_EP_V1.0)

The TOE will provide the capability to restrict the wireless access points to which it will connect.

4.2 Security Objectives for the TOE Environment

OE.CONFIG (PP_MD_V3.1)

TOE administrators will configure the Mobile Device security functions correctly to create the intended security policy.

OE.NOTIFY (PP_MD_V3.1)

The Mobile User will immediately notify the administrator if the Mobile Device is lost or stolen.

OE.PRECAUTION (PP_MD_V3.1)

The Mobile User exercises precautions to reduce the risk of loss or theft of the Mobile Device.

OE.IT_ENTERPRISE (EP_MDM_AGENT_V3.0)

The Enterprise IT infrastructure provides security for a network that is available to the TOE and mobile devices that prevents unauthorized access.

OE.MOBILE_DEVICE_PLATFORM (EP_MDM_AGENT_V3.0)

The MDM Agent relies upon the trustworthy Mobile platform and hardware to provide policy enforcement as well as cryptographic services and data protection. The Mobile platform provides trusted updates and software integrity verification of the MDM Agent.

OE.DATA_PROPER_ADMIN (EP_MDM_AGENT_V3.0)

TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

OE.DATA_PROPER_USER (EP_MDM_AGENT_V3.0)

Users of the mobile device are trained to securely use the mobile device and apply all guidance in a trusted manner.

OE.WIRELESS_NETWORK (EP_MDM_AGENT_V3.0)

A wireless network will be available to the mobile devices.

OE.NO_TOE_BYPASS (PP_WLAN_CLI_EP_V1.0)

Information cannot flow between external and internal networks located in different enclaves without passing through the TOE.

OE.TRUSTED_ADMIN (PP_WLAN_CLI_EP_V1.0)

TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

5 Extended Components Definition

The Security Target draws upon the extended components implicitly defined in the [PP_MD_V3.1], [PP_MD_AGENT_V3.0] and [PP_WLAN_CLI_EP_V1.0].

6 Security Functional Requirements

This chapter describes the Security Functional Requirements (SFRs) for the TOE. The SFRs have been taken from [PP_MD_V3.1], [PP_MD_AGENT_V3.0] and [PP_WLAN_CLI_EP_V1.0] with selections and assignments being applied.

For each SFR, the source of the SFR is indicated in parentheses, thus:

(MDF) – The SFR can be found in [PP_MD_V3.1],

(AGENT) – The SFR can be found in [PP_MD_AGENT_V3.0]

(WLAN) – The SFR can be found in [PP_WLAN_CLI_EP_V1.0]

Selections and assignment operations performed as required by the PP and EPs are marked in bold.

This Security Target (ST) does not identify selections or assignments already applied in the PP and EPs.

6.1 Security Audit (FAU)

Agent Alerts (FAU_ALT)

FAU_ALT_EXT.2 Extended: Agent Alerts

FAU_ALT_EXT.2.1(AGENT)

The MDM Agent shall provide an alert via the trusted channel to the MDM Server in the event of any of the following:

- successful application of policies to a mobile device;
- **receiving** periodic reachability events;
- **no other events**

FAU_ALT_EXT.2.2(AGENT)

The MDM Agent shall queue alerts if the trusted channel is not available.

Audit Data Generation (FAU_GEN)

FAU_GEN.1(1) Audit Data Generation

FAU_GEN.1.1(1)(MDF)

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 auditable events for the [not selected] level of audit;
3) All administrative actions;
4) Start-up and shutdown of the Rich OS;
5) Insertion or removal of removable media;
6) Specifically defined auditable events in Table 1**;**
7) **No other auditable events derived from this profile.**
8) **No additional auditable events.**

Note: For this element, Table 1 refers to Table 1 in [PP_MD_V3.1].

Table 3: Combined mandatory auditable events from [PP_MD_V3.1] and [PP_WLAN_CLI_EP_V1.0], below, presents the information given in Table 1 of [PP_MD_V3.1] combined with Table 2 of [PP_WLAN_CLI_EP_V1.0] as instructed in [PP_WLAN_CLI_EP_V1.0].

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1 (MDF)	None	
FAU_GEN.1/WLAN (WLAN)	None	
FAU_STG.1(MDF)	None	
FAU_STG.4(MDF)	None	
FCS_CKM_EXT.1(MDF)	None	No additional information
FCS_CKM_EXT.2(MDF)	None	
FCS_CKM_EXT.3(MDF)	None	
FCS_CKM_EXT.4(MDF)	None	
FCS_CKM_EXT.4(WLAN)	None	
FCS_CKM_EXT.5(MDF)	None	No additional information
FCS_CKM_EXT.6(MDF)	None	
FCS_CKM.1(MDF)	None	No additional information
FCS_CKM.1/WLAN(WLAN)	None	
FCS_CKM.2(*)(MDF)	None	
FCS_CKM.2/WLAN(WLAN)	None	
FCS_COP.1(*)(MDF)	None	
FCS_IV_EXT.1 (MDF)	None	
FCS_SRV_EXT.1 (MDF)	None	
FCS_STG_EXT.1(MDF)	Import or destruction of key	Identity of key. Role and identity of requestor
	No other events	
FCS_STG_EXT.2(MDF)	None	
FCS_STG_EXT.3(MDF)	Failure to verify integrity of stored key	Identity of key being verified
FCS_TLSC_EXT.1(WLAN)	Failure to establish an EAP-TLS session	Reason for failure. Non-TOE endpoint connection
	Establishment/termination of an EAP-TLS session	
FDP_DAR_EXT.1(MDF)	Failure to encrypt/decrypt data	No additional information
FDP_DAR_EXT.2(MDF)	Failure to encrypt/decrypt data	No additional information
FDP_IFC_EXT.1(MDF)	None	No additional information
FDP_STG_EXT.1(MDF)	Addition or removal of certificate from Trust Anchor Database	Subject name of certificate
FIA_PAE_EXT.1 (WLAN)	None	
FIA_PMG_EXT.1(MDF)	None	
FIA_TRT_EXT.1(MDF)	None	
FIA_UAU_EXT.1(MDF)	None	
FIA_UAU.5 (MDF)	None	
FIA_UAU.7 (MDF)	None	

Requirement	Auditable Events	Additional Audit Record Contents
FIA_X509_EXT.1(MDF)	Failure to validate x.509v3 certificate	Reason for failure of validation
FMT_MOF_EXT.1(MDF)	None	
FMT_SMF_EXT.1/WLAN(WLAN)	None	
FMT_UNR_EXT.1(AGENT) (Note: TD0237 is applicable here)	None	No additional information
FPT_AEX_EXT.1(MDF)	None	
FPT_AEX_EXT.2(MDF)	None	
FPT_AEX_EXT.3(MDF)	None	
FPT_JTA_EXT.1(MDF)	None	
FPT_JTA_EXT.1(MDF)	None	
FPT_KST_EXT.2(MDF)	None	
FPT_KST_EXT.3(MDF)	None	
FPT_NOT_EXT.1(MDF)	None	No additional information.
FPT_STM.1(MDF)	None	
FPT_TST_EXT.1(MDF)	Initiation of self-test	None
	Failure of self-test	
FPT_TST_EXT.2(1)(MDF)	Start-up of TOE	No additional information
	None	No additional information
FPT_TST_EXT.1/WLAN (WLAN)	Execution of this set of TSF self-tests. None	No additional information
FPT_TUD_EXT.1(MDF)	None	
FTA_SSL_EXT.1(MDF)	None	
FTA_TAB.1	None	
FTA_WSE_EXT.1/WLAN (WLAN)	All attempts to connect to access points	Identity of access point being connected to as well as success and failures (including reason for failure)
FTP_ITC_EXT.1(3)(WLAN)[1]	All attempts to establish a trusted channel	Identification of the non-TOE endpoint of the channel

Table 3: Combined mandatory auditable events from [PP_MD_V3.1] and [PP_WLAN_CLI_EP_V1.0]

FAU_GEN.1.2(1)(MDF)

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;
5) additional information in Table 1**;**
6) **no additional information.**

Note: For this element, Table 1 refers to Table 1 in [PP_MD_V3.1].

Table 3: Combined mandatory auditable events from [PP_MD_V3.1] and [PP_WLAN_CLI_EP_V1.0], above, presents the information given in Table 1 of [PP_MD_V3.1] combined with Table 2 of [PP_WLAN_CLI_EP_V1.0] as instructed in [PP_WLAN_CLI_EP_V1.0].

FAU_GEN.1(2) Audit Data Generation

FAU_GEN.1.1(2)(AGENT)

The MDM Agent shall be able to generate an MDM Agent audit record of the following auditable events:

- startup and shutdown of the MDM Agent,
- change in MDM policy,
- any modification commanded by the MDM Server,
- specifically defined auditable events listed in Table 1,
- **no other events.**

Note: For this element, Table 1 refers to Table 1 in [EP_MDM_AGENT_V3.0].

Table 4: Auditable events from [EP_MDM_AGENT_V3.0] in this ST presents the same information given in Table 1 of [EP_MDM_AGENT_V3.0].] for the convenience of the reader.

[1] "Detection of modification of channel data" was removed by TD0194

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ALT_EXT.2(AGENT)	Type of alert	No additional information
FAU_GEN.1 (AGENT)	None	N/A
FAU_SEL.1(AGENT)	All modifications to the audit configuration that occur while the audit collection functions are operating.	No additional information
FAU_STG_EXT.4 (AGENT) FCS_STG_EXT.1(1) (AGENT)	None	N/A
FCS_TLSC_EXT.1(AGENT)	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
FIA_ENR_EXT.2(AGENT)	Enrollment in management	Reference identifier of MDM Server
FMT_UNR_EXT.1(AGENT)[2]	None	No additional information
FMT_POL_EXT.2(AGENT)	Failure of policy validation	Reason for failure of validation
FMT_SMF_EXT.3(AGENT)	Success or failure of function	No additional information
FTP_ITC_EXT.1(2)(AGENT) FTP_ITT_EXT.1(AGENT)	Initiation and termination of trusted channel	Trusted channel protocol. Non-TOE endpoint of connection

Table 4: Auditable events from [EP_MDM_AGENT_V3.0]

FAU_GEN.1.2(2)(AGENT)

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

- date and time of the event,
- type of event,
- subject identity,
- (if relevant) the outcome (success or failure) of the event,
- additional information in Table 1**;**
- **no other relevant audit information**.

Note: For this element, Table 1 refers to Table 1 in [EP_MDM_AGENT_V3.0].

[2] Please see TD0237 which is applicable here.

Table 4: Auditable events from [EP_MDM_AGENT_V3.0], above, presents the same information given in Table 1 of [EP_MDM_AGENT_V3.0].] for the convenience of the reader.

Security Audit Event Selection (FAU_SEL)

FAU_SEL.1(2) Security Audit Event Selection

FAU_SEL.1.1(2)(AGENT)

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:

- event type;
- success of auditable security events;
- failure of auditable security events; and
- none.

Security Audit Event Storage (FAU_STG)

FAU_STG.1 Audit Storage Protection

FAU_STG.1.1(MDF)

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

FAU_STG.1.2(MDF)

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

FAU_STG.4 Prevention of Audit Data Loss

FAU_STG.4.1(MDF)

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

6.2 Cryptographic Support (FCS)

Cryptographic Key Management (FCS_CKM)

FCS_CKM.1(1) Cryptographic Key Generation

FCS_CKM.1.1(1)(MDF)

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 FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.3;
- **ECC schemes using**
  - **“NIST curves” P-384 and P-256 that meet the following: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4,**
  - **Curve25519 schemes that meet the following: [RFC7748]**

FCS_CKM.1(2) WLAN Cryptographic Key Generation (Symmetric Keys for WPA2 Connections)

FCS_CKM.1.1(2)(WLAN)

The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm PRF-384 and **no other** and specified cryptographic key sizes 128 bits and **no other key sizes** using a Random Bit Generator as specified in FCS_RBG_EXT.1 that meet the following: IEE 802.11-2012 and **no other standards**.

FCS_CKM.2(1) Cryptographic Key Establishment

FCS_CKM.2.1(1)(MDF)

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**.”

FCS_CKM.2(2) Cryptographic Key Establishment (While device is locked)

FCS_CKM.2.1(2)(MDF)

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.”

FCS_CKM.2(3) WLAN Cryptographic Key Distribution (GTK)

FCS_CKM.2.1(3)(WLAN)

The TSF shall decrypt Group Temporal Key in accordance with a specified cryptographic key distribution method **AES Key Wrap in an EAPOL-Key frame** that meets the following: RFC 3394 for AES Key Wrap, 802.11-2012 for the packet format and timing considerations and does not expose the cryptographic keys.

FCS_CKM_EXT.1 Extended Cryptographic Key Support (REK)

FCS_CKM_EXT.1.1(MDF)

The TSF shall support **mutable hardware** REK(s) with a **symmetric** key of strength **256 bits**.

FCS_CKM_EXT.1.2(MDF)

Each REK shall be hardware-isolated from Rich OS on the TSF in runtime.

FCS_CKM_EXT.1.3(MDF)

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

FCS_CKM_EXT.1.4(MDF)

A REK shall not be able to be read from or exported from the hardware.

Note: FCS_CKM_EXT.1.4 is included as required by Annex C.1 of the Protection Profile.

FCS_CKM_EXT.2 Cryptographic Key Random Generation

FCS_CKM_EXT.2.1(MDF)

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

FCS_CKM_EXT.3 Extended: Cryptographic Key Generation

FCS_CKM_EXT.3.1(MDF)

The TSF shall use **symmetric KEKs of 128 bit, 256-bit** security strength corresponding to at least the security strength of the keys encrypted by the KEK.

FCS_CKM_EXT.3.2(MDF)

The TSF shall generate all KEKs using one or more 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) **combine the KEK from other KEKs in a way that preserves the effective entropy of each factor by concatenating the keys and use a KDF (as described in SP 800-56c), encrypting one key with another**.

Note: The random number generator on the main device is used.

FCS_CKM_EXT.4 Extended: Key Destruction

FCS_CKM_EXT.4.1(MDF)

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 by 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.**
    - 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(MDF)

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

FCS_CKM_EXT.5 Extended: TSF Wipe

FCS_CKM_EXT.5.1(MDF)

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.**

FCS_CKM_EXT.5.2(MDF)

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

FCS_CKM_EXT.6 Extended: Salt Generation

FCS_CKM_EXT.6.1(MDF)

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

Note: the salt is generated using the random number generator implemented in the secure enclave, which like the one implemented in the main device, satisfies the requirements of FCS_RBG_EXT.1. A proprietary Entropy Assessment Report (EAR) has been provided to NIAP that gives details of both random number generators.

FCS_CKM_EXT.7 Extended: Cryptographic Key Support (REK)

FCS_CKM_EXT.7.1(MDF)

A REK shall not be able to be read from or exported from the hardware.

Cryptographic Operations (FCS_COP)

FCS_COP.1(1): Confidentiality Algorithms

FCS_COP.1.1(1)(MDF)

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),**

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

FCS_COP.1(2): Hashing Algorithms

FCS_COP.1.1(2)(MDF)

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.

FCS_COP.1(3): Signature Algorithms

FCS_COP.1.1(3)(MDF)

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-384 and P-256 that meet the following: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 5.**

FCS_COP.1(4) Keyed Hash Algorithms

FCS_COP.1.1(4)(MDF)

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 to 256 bit** 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".

FCS_COP.1(5) Password-Based Key Derivation Functions

FCS_COP.1.1(5)(MDF)

The TSF shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC-**SHA-256**, with **a minimum of 50,000** iterations, and output cryptographic key sizes **128, 256** that meet the following: NIST SP 800-132.

Note: The number of iterations is calibrated to take at least 100 to 150 milliseconds and is a minimum of 50,000. The number of iterations may be greater in some devices.

HTTPS Protocol (FCS_HTTPS)

FCS_HTTPS_EXT.1 Extended: HTTPS Protocol

FCS_HTTPS_EXT.1.1(MDF)

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

FCS_HTTPS_EXT.1.2(MDF)

The TSF shall implement HTTPS using TLS (FCS_TLSC_EXT.1).

FCS_HTTPS_EXT.1.3(MDF)

The TSF shall notify the application and **not establish the connection** if the peer certificate is deemed invalid.

Initialization Vector Generation (FCS_IV)

FCS_IV_EXT.1 Extended: Initialization Vector Generation

FCS_IV_EXT.1.1(MDF)

The TSF shall generate IVs in accordance with Table 11: References and IV Requirements for NIST-approved Cipher Modes.

Note: The referenced table 11 is found in [PP_MD_V3.1],

Random Bit Generation (FCS_RBG)

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

FCS_RBG_EXT.1.1(MDF)

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(MDF)

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from **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(MDF)

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

Cryptographic Algorithm Services (FCS_SRV)

FCS_SRV_EXT.1 Extended: Cryptographic Algorithm Services

FCS_SRV_EXT.1.1(MDF)

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

- All mandatory and **selected algorithms with the exception of ECC over curve 25519-based algorithms** in FCS_CKM.2(2)
- The following algorithms in FCS_COP.1(1): AES-CBC
- All mandatory and selected algorithms in FCS_COP.1(3)
- All mandatory and selected algorithms in FCS_COP.1(2)
- All mandatory and selected algorithms in FCS_COP.1(4)
- **No other cryptographic operations**

Cryptographic Key Storage (FCS_STG)

FCS_STG_EXT.1 Extended: Secure Key Storage

FCS_STG_EXT.1.1(MDF)

The TSF shall provide **software-based** secure key storage for asymmetric private keys and **symmetric keys, persistent secrets**.

FCS_STG_EXT.1.2(MDF)

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

FCS_STG_EXT.1.3(MDF)

The TSF shall be capable of destroying keys/secrets in the secure key storage upon request of **the administrator**.

FCS_STG_EXT.1.4(MDF)

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 **a common application developer**.

FCS_STG_EXT.1.5(MDF)

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 **a common application developer**.

FCS_STG_EXT.2 Extended: Encrypted Cryptographic Key Storage

FCS_STG_EXT.2.1(MDF)

The TSF shall encrypt all DEKs and KEKs, no long-term trusted channel key material, and all software-based key storage by KEKs that are

1) **Protected by the REK with**
    a. **encryption by a KEK chaining from a REK,**
2) **Protected by the REK and the password with**
    b. **encryption by a KEK chaining to a REK and the password-derived or biometric unlocked KEK.**

FCS_STG_EXT.2.2(MDF)

DEKs, KEKs, **no long-term trusted channel key material**, and **all software-based key storage** shall be encrypted using one of the following methods:

- **using AES in the Key Wrap (KW) mode**.

FCS_STG_EXT.3 Extended: Integrity of Encrypted Key Storage

FCS_STG_EXT.3.1(MDF)

The TSF shall protect the integrity of any encrypted DEKs and KEKs and no other keys by an immediate application of the key for decrypting the protected data followed by a successful verification of the decrypted data with a previously known information.

FCS_STG_EXT.3.2(MDF)

The TSF shall verify the integrity of the **MAC** of the stored key prior to use of the key.

FCS_STG_EXT.4 Cryptographic Key Storage:

FCS_STG_EXT.4.1(AGENT)

The MDM Agent shall use the platform provided key storage for all persistent secret and private keys.

TLS Client Protocol (FCS_TLSC)

FCS_TLSC_EXT.1 Extended: TLS Protocol

FCS_TLSC_EXT.1.1(MDF)

The TSF shall implement TLS 1.2 (RFC 5246) supporting the following ciphersuites:

- Mandatory Ciphersuites:
  - TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- Optional Ciphersuites:
  - **TLS_ECDHE_RSA_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**
  - **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(MDF)

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

FCS_TLSC_EXT.1.3(MDF)

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

FCS_TLSC_EXT.1.4(MDF)

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

FCS_TLSC_EXT.1/WLAN Extensible Authentication Protocol-Transport Layer Security (EAP-TLS)

FCS_TLSC_EXT.1.1(WLAN)

The TSF shall implement TLS 1.0 and **TLS 1.1 (RFC4346), TLS 1.2 (RFC 5246)** in support of the EAP-TLS protocol as specified in RFC 5216 supporting the following ciphersuites:

- Mandatory Ciphersuites in accordance with RFC 5246:
  - 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_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246**
  - **TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246**

FCS_TLSC_EXT.1.2(WLAN)

The TSF shall generate random values used in the EAP-TLS exchange using the RBG specified in FCS_RBG_EXT.1.

FCS_TLSC_EXT.1.3(WLAN)

The TSF shall use X509 v3 certificates as specified in FIA_X509_EXT.1

FCS_TLSC_EXT.1.4(WLAN)

The TSF shall verify that the server certificate presented includes the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.

FCS_TLSC_EXT.1.5(WLAN)

The TSF shall allow an authorized administrator to configure the list of CAs that are allowed to sign authentication server certificates that are accepted by the TOE.

FCS_TLSC_EXT.1.6/(WLAN)

The TSF shall allow an authorized administrator to configure the list of algorithm suites that may be proposed and accepted during the EAP-TLS exchanges.

FCS_TLSC_EXT.2 Extended: TLS Protocol

FCS_TLSC_EXT.2.1 (MDF)[3]

The TSF shall present the Supported Elliptic Curves Extension in the Client Hello handshake message with the following NIST curves: **secp256r1, secp384r1**.

[3] Both TD0236 TD0244 are applicable to this element.

6.3 User Data Protection (FDP)

Access Control (FDP_ACF)

FDP_ACF_EXT.1 Extended: Security Access Control

FDP_ACF_EXT.1.1(MDF)

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

FDP_ACF_EXT.1.2(MDF)

The TSF shall provide an access control policy that prevents **application** from accessing **all** data stored by other **application**. Exceptions may only be explicitly authorized for such sharing by **a common application developer**.

Data-At-Rest Protection (FDP_DAR)

FDP_DAR_EXT.1 Extended: Protected Data Encryption

FDP_DAR_EXT.1.1(MDF)

Encryption shall cover all protected data.

FDP_DAR_EXT.1.2(MDF)

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

FDP_DAR_EXT.2 Extended: Sensitive Data Encryption

FDP_DAR_EXT.2.1(MDF)

The TSF shall provide a mechanism for applications to mark data and keys as sensitive.

FDP_DAR_EXT.2.2(MDF)

The TSF shall use an asymmetric key scheme to encrypt and store sensitive data received while the product is locked.

FDP_DAR_EXT.2.3(MDF)

The TSF shall encrypt any stored symmetric key and any stored private key of the asymmetric key(s) used for the protection of sensitive data according to FCS_STG_EXT.2.1 selection 2.

FDP_DAR_EXT.2.4(MDF)

The TSF shall decrypt the sensitive data that was received while in the locked state upon transitioning to the unlocked state using the asymmetric key scheme and shall re-encrypt that sensitive data using the symmetric key scheme.

Subset Information Flow Control - VPN (FDP_IFC)

FDP_IFC_EXT.1 Extended: Subset Information Flow Control

FDP_IFC_EXT.1.1(MDF)

The TSF shall **provide a VPN client which can protect all IP traffic using IPSec** with the exception of IP traffic required to establish the VPN connection.

Note: The VPN client shall be validated in a separate evaluation.

Storage of Critical Biometric Parameters (FDP_PBA)

FDP_PBA_EXT.1 Extended: Storage of Critical Biometric Parameters

FDP_PBA_EXT.1.1(MDF)

The TSF shall protect the authentication template **using passcode as an additional factor**.

Certificate Data Storage (FDP_STG)

FDP_STG_EXT.1 Extended: User Data Storage

FDP_STG_EXT.1.1(MDF)

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

Inter-TSF User Data Protected Channel (FDP_UPC)

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

FDP_UPC_EXT.1.1(MDF)

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(MDF)

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

6.4 Identification and Authentication (FIA)

Authentication Failures (FIA_AFL)

FIA_AFL_EXT.1 Authentication failure handling

FIA_AFL_EXT.1.1(MDF)

The TSF shall consider password and **no other** as critical authentication mechanisms.

FIA_AFL_EXT.1.2(MDF)

The TSF shall detect when a configurable positive integer within **2 to 10** of **non-unique** unsuccessful authentication attempts occur related to last successful authentication for each authentication mechanism.

FIA_AFL_EXT.1.3(MDF)

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

FIA_AFL_EXT.1.4(MDF)

When the defined number of unsuccessful authentication attempts has exceeded the maximum allowed for a given authentication mechanism, all future authentication attempts will be limited to other available authentication mechanisms, unless the given mechanism is designated as a critical authentication mechanism.

FIA_AFL_EXT.1.5(MDF)

When the defined number of unsuccessful authentication attempts for the last available authentication mechanism or single critical authentication mechanism has been surpassed, the TSF shall perform a wipe of all protected data.

FIA_AFL_EXT.1.6(MDF)

The TSF shall increment the number of unsuccessful authentication attempts prior to notifying the user that the authentication was unsuccessful.

Bluetooth Authorization and Authentication (FIA_BLT)

FIA_BLT_EXT.1 Extended: Bluetooth User Authorization

FIA_BLT_EXT.1.1(MDF)

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

FIA_BLT_EXT.2 Extended: Bluetooth Mutual Authentication

FIA_BLT_EXT.2.1(MDF)

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

FIA_BLT_EXT.3 Extended: Rejection of Duplicate Bluetooth Connections

FIA_BLT_EXT.3.1(MDF)

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

FIA_BLT_EXT.4 Extended: Secure Simple Pairing

FIA_BLT_EXT.4.1(MDF)

The TOE shall support Bluetooth Secure Simple Pairing, both in the host and the controller. Furthermore, Secure Simple Pairing shall be used during the pairing process if the remote device also supports it.

Biometric Authentication (FIA_BMG)

FIA_BMG_EXT.1 Extended: Accuracy of Biometric Authentication

FIA_BMG_EXT.1.1(1)(MDF)(Touch ID Gen.1)

The one-attempt BAF False Accept Rate (FAR) for **fingerprint authentication** shall not exceed **1:10000** with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in **20**.

FIA_BMG_EXT.1.1(2)(MDF)(Touch ID Gen.2 and Gen.3)

The one-attempt BAF False Accept Rate (FAR) for **fingerprint authentication** shall not exceed **1:50000** with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in **20**.

FIA_BMG_EXT.1.1(3)(MDF)(Face ID)

The one-attempt BAF False Accept Rate (FAR) for **face authentication** shall not exceed **1:10000** with a one-attempt BAF False Reject Rate (FRR) not to exceed 1 in **10.**

FIA_BMG_EXT.1.2(1)(MDF)(Touch ID)

The overall System Authentication False Accept Rate (SAFAR) shall be no greater than 1 in **10000** within a 1% margin.

FIA_BMG_EXT.1.2(2)(MDF)(Face ID)

The overall System Authentication False Accept Rate (SAFAR) shall be no greater than 1 in **200000** within a 1% margin.

FIA_BMG_EXT.2 Extended: Biometric Enrollment

FIA_BMG_EXT.2.1(1)(MDF)

The TSF shall only use biometric samples of sufficient quality for enrollment. Sample data shall have **sufficient fingerprint content and no severe structural sensing artifacts**.

FIA_BMG_EXT.2.1(2)(MDF)

The TSF shall only use biometric samples of sufficient quality for enrollment. Sample data shall have **sufficient content of face and no severe structural sensing artifacts**.

FIA_BMG_EXT.3 Extended: Biometric Verification

FIA_BMG_EXT.3.1(1)(MDF)

The TSF shall only use biometric samples of sufficient quality for verification. As such, sample data shall have **sufficient fingerprint content and no severe structural sensing artifacts**.

FIA_BMG_EXT.3.1(2)(MDF)

The TSF shall only use biometric samples of sufficient quality for verification. As such, sample data shall have **sufficient fingerprint content and no severe structural sensing artifacts**.

FIA_BMG_EXT.5 Extended: Handling Unusual Biometric Templates

FIA_BMG_EXT.5.1(MDF)

The matching algorithm shall handle properly formatted enrollment templates and/or authentication templates, especially those with unusual data properties, appropriately. If such templates contain incorrect syntax, are of low quality, or contain enrollment data considered unrealistic for a given modality, then they shall be rejected by the matching algorithm and an error code shall be reported.

Enrollment of Mobile Device into Management (FIA_ENR)

FIA_ENR_EXT.2 Extended: Enrollment of Mobile Device into Management

FIA_ENR_EXT.2.1(AGENT)

The MDM Agent shall record the reference identifier of the MDM Server during the enrollment process.

Port Access Entity Authentication (FIA_PAE)

FIA_PAE_EXT.1 Extended: PAE Authentication

FIA_PAE_EXT.1.1(WLAN)

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

Password Management (FIA_PMG)

FIA_PMG_EXT.1 Extended: Password Management

FIA_PMG_EXT.1.1(MDF)

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: "!", "@", "#", "$", "%", "^", "&", "*", "(", ")";**
2) Password length up to **16** characters shall be supported.

Authentication Throttling (FIA_TRT)

FIA_TRT_EXT.1 Extended: Authentication Throttling

FIA_TRT_EXT.1.1(MDF)

The TSF shall limit automated user authentication attempts by **enforcing a delay between incorrect authentication attempts** for all authentication mechanisms selected in FIA_UAU.5.1. The minimum delay shall be such that no more than 10 attempts can be attempted per 500 milliseconds.

User Authentication (FIA_UAU)

FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UAU.5.1(MDF)

The TSF shall provide password and **fingerprint, face** to support user authentication.

Note: iOS does not support hybrid authentication factor

FIA_UAU.5.2(MDF)

The TSF shall authenticate any user's claimed identity according to the **validation of the user's password, fingerprint, or face.**

Note: The TSS describes authentication rules in more detail.

FIA_UAU.6 Re-Authentication

FIA_UAU.6.1(1)(MDF)

The TSF shall re-authenticate the user via the Password Authentication Factor under the conditions attempted change to any supported authentication mechanisms.

FIA_UAU.6.1(2)(MDF)

The TSF shall re-authenticate the user via an authentication factor defined in FIA_UAU.5.1 under the conditions TSF-initiated lock, user-initiated lock, and **no other conditions.**

FIA_UAU.7 Protected authentication feedback

FIA_UAU.7.1(MDF)

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

FIA_UAU_EXT.1 Extended: Authentication for Cryptographic Operation

FIA_UAU_EXT.1.1(MDF)

The TSF shall require the user to present the Password Authentication Factor prior to decryption of protected data and encrypted DEKs, KEKs and **all software-based key storage** at startup.

FIA_UAU_EXT.2 Extended: Timing of Authentication

FIA_UAU_EXT.2.1(MDF)

The TSF shall **allow answering calls, make emergency calls, use the cameras (unless their use is generally disallowed), and the flashlight** on behalf of the user to be performed before the user is authenticated.

FIA_UAU_EXT.2.2(MDF)

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

X509 Certificates (FIA_X509)

FIA_X509_EXT.1 Extended: Validation of certificates

FIA_X509_EXT.1.1(MDF)

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**.
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - (Conditional) Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the extendedKeyUsage field.

FIA_X509_EXT.1.2(MDF)

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.

X509 Certificate Authentication

FIA_X509_EXT.2 Extended: X509 certificate authentication

FIA_X509_EXT.2.1(MDF)

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for IPsec, TLS, HTTPS, and code signing for system software updates, code signing for mobile applications, code signing for integrity verification, and no additional uses.

FIA_X509_EXT.2.2(MDF)

When the TSF cannot establish a connection to determine the revocation status 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.

FIA_X509_EXT.2/WLAN Extended: X509 certificate authentication (EAP-TLS)

FIA_X509_EXT.2.1(WLAN)

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for EAP-TLS exchanges.

FIA_X509_EXT.2.2(WLAN)

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.

Request Validation of Certificates

FIA_X509_EXT.3 Extended: Request Validation of certificates

FIA_X509_EXT.3.1(MDF)

The TSF shall provide a certificate validation service to applications.

FIA_X509_EXT.3.2(MDF)

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

6.5 Security Management (FMT)

Management of Functions in TSF (FMT_MOF)

FMT_MOF_EXT.1 Extended: Management of security functions behavior

FMT_MOF_EXT.1.1(MDF)

The TSF shall restrict the ability to perform the functions in column **3** of Table **5** to the user.

FMT_MOF_EXT.1.2(MDF)

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

Note: The referenced Table 5 is found in [PP_MD_V3.1].

Trusted Policy Update

FMT_POL_EXT.2 Trusted Policy Update

FMT_POL_EXT.2.1(AGENT)

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

FMT_POL_EXT.2.2(AGENT)

The MDM Agent shall not install policies if the policy signing certificate is deemed invalid.

Specification of Management Functions (FMT_SMF)

FMT_SMF_EXT.1 Extended: Specification of Management Functions

FMT_SMF_EXT.1.1(MDF)

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

Management Function	**User**	**Administrator**	**Administrator, when enrolled in MDM**
Function 1: Configure password policy: a. minimum password length b. minimum password complexity c. maximum password lifetime	-	X	X
Function 2: Configure session locking policy: a. screen-lock enabled/disabled b. screen lock timeout c. number of authentication failures	-	X	X
Function 3: Enable/disable the VPN protection: a. across device b. on a per-app basis	-	X	X
Function 4: Enable/disable Bluetooth, Wi-Fi, cellular radio	X	-	-
Function 5: Enable/disable **cameras**: a. across device b. on a per-app basis	 - X	 X -	 - -
Function 5: Enable/disable microphones: b. on a per-app basis	X	-	-
Function 6: Transition to the locked state	-	X	-
Function 7: TSF wipe of protected data	-	X	-
Function 8: Configure application installation policy by c. denying installation of applications	-	X	-
Function 9: Import keys/secrets into the secure key storage	-	X	-
Function 10: Destroy imported keys/secrets and **no other keys/secrets** in the secure key storage	-	X	-
Function 11: Import X.509v3 certificates in the Trust Anchor Database	-	X	X

Management Function	**User**	**Administrator**	**Administrator, when enrolled in MDM**
Function 12: Remove imported X509v3 certificates and **no other X509v3 certificates** in the Trust Anchor Database	X	-	-
Function 13: Enroll the TOE in management	X	-	-
Function 14: Remove applications	-	X	X
Function 15: Update system software	-	X	-
Function 16: Install applications	-	X	X
Function 17: Remove Enterprise applications	-	X	-
Function 18: Configure the Bluetooth trusted channel: a. disable/enable the Discoverable mode (for BR/EDR) b. change the Bluetooth device name i. specify minimum level of security for each pairing. (for BR/EDR and LE)	X	-	-
Function 19: Enable/disable display notifications in the locked state of: f. all notifications	X	X	-
Function 22: enable/disable location services: a. across device b. on a per-app basis	X	X	-
Function 23: Enable/disable the use of Biometric Authentication Factor	-	X	X
Function 28: Wipe Enterprise data	-	X	-
Function 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	-	X	-
Function 33: Configure **certificate** used to validate digital signature on application	-	X	X
Function 36: Configure the unlock banner	-	X	X
Function 37: Configure the auditable items	-	X	X

Table 5: Management Functions

Note: Most of the administrator management functions are implemented by the specification and installation of Configuration Profiles. Also, for the enforcement of other functions, such as the password policy, the installation of Configuration Profiles with dedicated values for some of the payload keys is required.

Note: Function 20 has not been included in the table since the TOE has data-at-rest protection natively enabled which cannot be turned off.

Note: Function 21 has not been included in the table since the TOE does not support removable media. Backups can be made using iTunes, and backup encryption can be made mandatory.

Note: Function 24 has not been included in the table since the function is optional.

Note: Function 25 has not been included in the table since the function is optional.

Note: Function 26 has not been included in the table since the TOE does not support a developer mode.

Note: Function 27 has not been included in the table since the TOE (in its evaluated configuration) does not support bypass of local user authentication.

Note: Function 29 has not been included in the table since the function is optional.

Note: Function 31 has not been included in the table since the function is optional.

Note: Function 32 has not been included in the table since audit review is not implemented on TOE devices.

Note: Function 34 has not been included in the table since the function is optional.

Note: Function 35 has not been included in the table since the function is optional.

Note: Function 38 has not been included in the table since the function is optional.

Note: Function 39 has not been included in the table since the function is optional.

Note: Function 40 has not been included in the table since the function is optional.

Note: Function 41 has not been included in the table since the function is optional.

Note: Function 42 has not been included in the table since the function is optional.

Note: Function 43 has not been included in the table since the function is optional.

Note: Function 44 has not been included in the table since the function is optional.

Note: Function 45 has not been included in the table since the function is optional.

Note: Function 46 has not been included in the table since the function is optional.

Note: Function 47 has not been included in the table since IPSEC is not selected.

FMT_SMF_EXT.1/WLAN Specification of Management Functions

FMT_SMF_EXT.1.1(WLAN)

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

- configure security policy for each wireless network:

- specify the CA(s) from which the TSF will accept WLAN authentication server certificates(s)
- security type
- authentication protocol
- client credentials to be used for authentication

FMT_SMF_EXT.2 Extended: Specification of Remediation Actions

FMT_SMF_EXT.2.1(MDF)

The TSF shall offer wipe of all data associated with profiles under management upon unenrollment and when issuing a remote wipe command.

FMT_SMF_EXT.3 Extended: Specification of management Functions

FMT_SMF_EXT.3.1(AGENT)

The MDM Agent shall be capable of interacting with the platform to perform the following functions:

- administrator-provided management functions in MDF PP;
- Import the certificates to be used for authentication of MDM Agent communications
- **no additional functions**

FMT_SMF_EXT.3.2(AGENT)

The MDM Agent shall be capable of performing the following functions:

- Enroll in management;
- Configure whether users can unenroll the agent from management
- **no other functions**

User Unenrollment Prevention

FMT_UNR_EXT.1 Extended: User Unenrollment Prevention

FMT_UNR_EXT.1.1(AGENT)

The MDM Agent shall provide a mechanism to enforce the following behavior upon an attempt to unenroll the mobile device from management: **prevent the unenrollment from occurring.**

6.6 Protection of the TSF (FPT)

Anti-Exploitation Services (FPT_AEX)

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

FPT_AEX_EXT.1.1(MDF)

The TSF shall provide address space layout randomization ASLR to applications.

FPT_AEX_EXT.1.2(MDF)

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

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

FPT_AEX_EXT.2.1(MDF)

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

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

FPT_AEX_EXT.3.1(MDF)

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

FPT_AEX_EXT.4 Extended: Domain Isolation

FPT_AEX_EXT.4.1(MDF)

The TSF shall protect itself from modification by untrusted subjects.

FPT_AEX_EXT.4.2(MDF)

The TSF shall enforce isolation of address space between applications.

JTAG Disablement (FPT_JTA)

FPT_JTA_EXT.1 Extended: JTAG Disablement

FPT_JTA_EXT.1.1(MDF)

The TSF shall disable access through hardware to JTAG.

Key Storage (FPT_KST)

FPT_KST_EXT.1 Extended: Key Storage

FPT_KST_EXT.1.1(MDF)

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

FPT_KST_EXT.2 Extended: No Key Transmission

FPT_KST_EXT.2.1(MDF)

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

FPT_KST_EXT.3 Extended: No Plaintext Key Export

FPT_KST_EXT.3.1(MDF)

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

Self-Test Notification (FPT_NOT)

FPT_NOT_EXT.1 Extended: Self-Test Notification

FPT_NOT_EXT.1.1(MDF)

The TSF shall transition to non-operational mode and **no other actions** when the following types of failures occur:

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

Reliable Time Stamps (FPT_STM)

FPT_STM.1 Reliable time stamps

FPT_STM.1.1(MDF)

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

TSF Functionality Testing (FPT_TST)

FPT_TST_EXT.1 Extended: TSF Cryptographic Functionality Testing

FPT_TST_EXT.1.1(MDF)

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.

FPT_TST_EXT.1/WLAN TSF Cryptographic Functionality Testing (Wireless LAN)

FPT_TST_EXT.1.1(WLAN)

The **TOE** shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

FPT_TST_EXT.1.2(WLAN)

The **TOE** shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the TSF-provided cryptographic services.

TSF Integrity Testing

FPT_TST_EXT.2 Extended: TSF Integrity Testing

FPT_TST_EXT.2.1(1)(MDF)

The TSF shall verify the integrity of the bootchain up through the Application Processor OS kernel stored in mutable media prior to its execution through the use of **a digital signature using a hardware-protected asymmetric key**.

FPT_TST_EXT.3 Extended: TSF Integrity Testing

FPT_TST_EXT.3.1(MDF)

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

Trusted Update (FPT_TUD)

FPT_TUD_EXT.1 Extended: Trusted Update: TSF version query

FPT_TUD_EXT.1.1(MDF)

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

FPT_TUD_EXT.1.2(MDF)

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(MDF)

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

Trusted Update Verification (FPT_TUD_EXT)

FPT_TUD_EXT.2 Extended: Trusted Update Verification

FPT_TUD_EXT.2.1(MDF)

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(MDF)

The TSF shall **never update** the TSF boot integrity **key**.

FPT_TUD_EXT.2.3(MDF)

The TSF shall verify that the digital signature verification key used for TSF updates **matches an immutable hardware-protected public key**.

FPT_TUD_EXT.2.4(MDF)

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

FPT_TUD_EXT.3 Extended: Trusted Update Verification

FPT_TUD_EXT.3.1(MDF)

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

FPT_TUD_EXT.4 Extended: Trusted Update Verification

FPT_TUD_EXT.4.1(MDF)

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

FPT_TUD_EXT.4.2(MDF)

The TSF shall verify that software updates to the TSF are a current or later version than the current version of the TSF.

6.7 TOE Access (FTA)

Default TOE Access Banners (FTA_TAB)

FTA_TAB.1 Default TOE Access Banners

FTA_TAB.1.1(MDF)

Before establishing a user session, the TSF shall display an advisory warning message regarding unauthorized use of the TOE.

Session Locking (FTA_SSL)

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

FTA_SSL_EXT.1.1(MDF)

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

FTA_SSL_EXT.1.2(MDF)

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

FTA_SSL_EXT.1.3(MDF)

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

a) clearing or overwriting display devices, obscuring the previous contents;
b) **zeroize the decrypted class key for the NSFileProtectionComplete class.**

Wireless Network Access (FTA_WSE)

FTA_WSE_EXT.1 Wireless Network Access

FTA_WSE_EXT.1.1(WLAN)

The TSF shall be able to attempt connections only to wireless networks specified as acceptable networks as configured by the administrator in FMT_SMF_EXT.1.1/WLAN.

6.8 Trusted Path/Channels (FTP)

Trusted Channel Communication (FTP_ITC)

FTP_ITC_EXT.1(2) Extended: Trusted channel Communication

FTP_ITC_EXT.1.1(2)(AGENT)

The TSF shall use **HTTPS** 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.

Note: The Agent EP modifies FTP_ITC_EXT.1(1) and is labeled as FTP_ITC_EXT.1(2) for clarity.

FTP_ITC_EXT.1.2(2)(AGENT)

The TSF shall permit the TSF and the MDM Server and **no other IT entities** to initiate communication via the trusted channel.

FTP_ITC_EXT.1.3(2)(AGENT)

The TSF shall initiate communication via the trusted channel for wireless access point connections, administrative communication, configured enterprise connections, and **no other connections**.

FTP_ITC_EXT.1(3) Extended: Trusted channel Communication

FTP_ITC_EXT.1.1(3)(WLAN)

The TSF shall use 802.11-2012, 802.1X, and EAP-TLS to provide a trusted communication channel between itself and a wireless access point 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(3)(WLAN)

The TSF shall initiate communication via the trusted channel for wireless access point connections.

6.10 Security Functional Requirements Rationale

The requirements in the PP are assumed to represent a complete set of requirements that serve to address any interdependencies. Given that all of the appropriate functional requirements given in the PPs have been copied into this [ST], the dependency analysis for the requirements is assumed to be already performed by the PP authors and is not reproduced in this document.

7 Security Assurance Requirements

The Security Assurance Requirements (SARs) for the TOE are defined in [PP_MD_V3.1]. They consist of the assurance components of Evaluation Assurance Level (EAL1) as defined in part 3 of the CC augmented by ASE_SPD.1 and ALC_TSU_EXT.1, which is defined in [PP_MD_V3.1]. These security assurance requirements are also applicable to the EP [EP_MDM_AGENT_V3.0] and [PP_WLAN_CLI_EP_V1.0

The assurance components included in [PP_MD_V3.1] are:

- ASE_CCL.1
- ASE_ECD.1
- ASE_INT.1
- ASE_OBJ.1
- ASE_REQ.1
- ASE_SPD.1
- ASE_TSS.1
- ADV_FSP.1
- AGD_OPE.1
- AGD_PRE.1
- ALC_CMC.1
- ALC_CMS.1
- ALC_TSU_EXT.1
- ATE_IND.1
- AVA_VAN.1

7.1 Security Target Evaluation (ASE)

7.1.1 Conformance Claims (ASE_CCL.1)

ASE_CCL.1.1D

The developer shall provide a conformance claim.

ASE_CCL.1.2D

The developer shall provide a conformance claim rationale.

ASE_CCL.1.1C

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

ASE_CCL.1.2C

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

ASE_CCL.1.3C

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

ASE_CCL.1.4C

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

ASE_CCL.1.5C

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

ASE_CCL.1.6C

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

ASE_CCL.1.7C

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

ASE_CCL.1.8C

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

ASE_CCL.1.9C

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

ASE_CCL.1.10C

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

ASE_CCL.1.1E

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

7.1.2 Extended Components Definition (ASE_ECD.1)

ASE_ECD.1.1D

The developer shall provide a statement of security requirements.

ASE_ECD.1.2D

The developer shall provide an extended components definition.

ASE_ECD.1.1C

The statement of security requirements shall identify all extended security requirements.

ASE_ECD.1.2C

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

ASE_ECD.1.3C

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

ASE_ECD.1.4C

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

ASE_ECD.1.5C

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

ASE_ECD.1.1E

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

ASE_ECD.1.2E

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

7.1.3 ST Introduction (ASE_INT.1)

ASE_INT.1.1D

The developer shall provide an ST introduction.

ASE_INT.1.1C

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

ASE_INT.1.2C

The ST reference shall uniquely identify the ST.

ASE_INT.1.3C

The TOE reference shall identify the TOE.

ASE_INT.1.4C

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

ASE_INT.1.5C

The TOE overview shall identify the TOE type.

ASE_INT.1.6C

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

ASE_INT.1.7C

The TOE description shall describe the physical scope of the TOE.

ASE_INT.1.8C

The TOE description shall describe the logical scope of the TOE.

ASE_INT.1.1E

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

ASE_INT.1.2E

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

7.1.4 Security Objectives for the Operational Environment (ASE_OBJ.1)

ASE_OBJ.1.1D

The developer shall provide a statement of security objectives.

ASE_OBJ.1.1C

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

ASE_OBJ.1.1E

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

7.1.5 Stated Security Requirements (ASE_REQ.1)

ASE_REQ.1.1D

The developer shall provide a statement of security requirements.

ASE_REQ.1.2D

The developer shall provide a security requirements rationale.

ASE_REQ.1.1C

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

ASE_REQ.1.2C

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

ASE_REQ.1.3C

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

ASE_REQ.1.4C

All operations shall be performed correctly.

ASE_REQ.1.5C

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

ASE_REQ.1.6C

The statement of security requirements shall be internally consistent.

ASE_REQ.1.1E

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

7.1.6 Security Problem Definition (ASE_SPD.1)

ASE_SPD.1.1D

The developer shall provide a security problem definition.

ASE_SPD.1.1C

The security problem definition shall describe the threats.

ASE_SPD.1.2C

All threats shall be described in terms of a threat agent, an asset, and an adverse action.

ASE_SPD.1.3C

The security problem definition shall describe the OSPs.

ASE_SPD.1.4C

The security problem definition shall describe the assumptions about the operational environment of the TOE.

ASE_SPD.1.1E

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

7.1.7 TOE Summary Specification (ASE_TSS.1)

ASE_TSS.1.1D

The developer shall provide a TOE summary specification.

ASE_TSS.1.1C

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

ASE_TSS.1.1E

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

ASE_TSS.1.2E

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

7.2 Development (ADV)

7.2.1 Basic Functional Specification (ADV_FSP.1)

ADV_FSP.1.1D

The developer shall provide a functional specification.

ADV_FSP.1.2D

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

ADV_FSP.1.1C

The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.2C

The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.3C

The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.4C

The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ADV_FSP.1.1E

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

ADV_FSP.1.2E

The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

7.3 Guidance documents (AGD)

7.3.1 Operational User Guidance (AGD_OPE.1)

AGD_OPE.1.1D

The developer shall provide operational user guidance.

AGD_OPE.1.1C

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

AGD_OPE.1.2C

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

AGD_OPE.1.3C

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

AGD_OPE.1.4C

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

AGD_OPE.1.5C

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

AGD_OPE.1.6C

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

AGD_OPE.1.7C

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1E

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

7.3.2 Preparative Procedures (AGD_PRE.1)

AGD_PRE.1.1D

The developer shall provide the TOE including its preparative procedures.

AGD_PRE.1.1D

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

AGD_PRE.1.2D

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

AGD_PRE.1.1E

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

AGD_PRE.1.2E

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

7.4 Life-cycle support (ALC)

7.4.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1D

The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.1.1C

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1C

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

7.4.2 TOE CM Coverage (ALC_CMS.1)

ALC_CMS.2.1D

The developer shall provide a configuration list for the TOE.

ALC_CMS.2.1C

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

ALC_CMS.2.2C

The configuration list shall uniquely identify the configuration items.

ALC_CMS.2.1E

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

7.4.3 Timely Security Updates (ALC_TSU_EXT.1)

ALC_TSU_EXT.1.1D

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

ALC_TSU_EXT.1.1C

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

ALC_TSU_EXT.1.2C

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

ALC_TSU_EXT.1.3C

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

ALC_TSU_EXT.1.4C

The description shall include where users can seek information about the availability of new updates including details (e.g. CVE identifiers) of the specific public vulnerabilities correct by each update.

ALC_TSU_EXT.1.1E

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

7.5 Tests (ATE)

7.5.1 Independent Testing - Conformance (ATE_IND.1)

ATE_IND.1.1D

The developer shall provide the TOE for testing.

ATE_IND.1.1C

The TOE shall be suitable for testing.

ATE_IND.1.1E

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

ATE_IND.1.2E

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

7.6 Vulnerability assessment (AVA)

7.6.1 Vulnerability Survey (AVA_VAN.1)

AVA_VAN.1.1D

The developer shall provide the TOE for testing.

AVA_VAN.1.1C

The TOE shall be suitable for testing.

AVA_VAN.1.1E

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

AVA_VAN.1.2E

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

AVA_VAN.1.3E

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

8 TOE Summary Specification (TSS)

This chapter describes the relevant aspects of how the security functional requirements are implemented in the security functionality provided by the TOE. This chapter is structured in accordance with the structuring of the security functional requirements in section 6, *Security Functional Requirements*, of this document, which in turn has been taken from the structure of the description of the security functional requirements in [PP_MD_V3.1], [PP_MD_AGENT_3.0] and [PP_WLAN__CLI_EP_V1.0].

The TOE security boundary is described in section 1.5 TOE Architecture, above.

8.1 Hardware Protection Functions

8.1.1 The Secure Enclave

The Secure Enclave is a coprocessor fabricated in the Apple A7, A8, A8X, A9, A9X, A10 Fusion, A10X Fusion, and A11 Bionic processors. It utilizes its own secure boot and personalized software update separate from the application processor. It provides all cryptographic operations for Data Protection key management and maintains the integrity of Data Protection even if the kernel has been compromised.

The Secure Enclave uses encrypted memory and includes a hardware random number generator. Its microkernel is based on the L4 family, a second-generation microkernel generally used to implement UNIX-like operating systems, with modifications by Apple. Communication between the Secure Enclave and the application processor is isolated to an interrupt-driven mailbox and shared memory data buffers. Note that only a small dedicated amount of memory used for communication between the secure enclave and the main system is shared. The main system has no access to other memory areas of the secure enclave and no keys or key material may be exported.

Each Secure Enclave is provisioned during fabrication with its own Unique ID (UID) that is not accessible to other parts of the system and is not known to Apple. When the device starts up, an ephemeral key is created, entangled with its UID, and used to encrypt the Secure Enclave’s portion of the device’s memory space.

Additionally, data that is saved to the file system by the Secure Enclave is encrypted with a key entangled with the UID and an anti-replay counter.

The UID also serves as the REK for the whole device.

In addition to the UID also the Group Key (GID) and Apple's root certificate are provisioned during manufacturing. The GID is only unique per device type and is used in the secure software update process. Apple's root certificate is used to verify the integrity and authenticity of software during the secure boot process and for updates of the system software.

The Secure Enclave has its own physical noise source and random number generator which is used for generating the salt value for the password-based key generation function (PBKDF) 2, which uses AES with the device UID as the key for the pseudorandom function (PRF). The counter is calculated such that it takes at least 80 milliseconds to execute the function. While the counter value therefore may vary per device type, it is larger than 10,000 for each device type. The salt (128 bit) is regenerated every time the passcode changes. The salt value is stored AES encrypted with the UID in the system keybag.

Other salt values used for functions in iOS are generated using the True Random Number Generator (TRNG) of the application processor. This includes nonces used in the generation of DSA signatures as well as nonces required for the Wi-Fi and TLS protocol.

8.1.2 Memory Protection

iOS uses the read and write protection for memory pages provided by the advanced Reduced Instruction Set (RISC) machine (ARM) processor for separating applications from the kernel and to provide a sandbox for each application.

Further protection is provided by iOS using ARM's Execute Never (XN) feature, which marks memory pages as non-executable. Memory pages marked as both writable and executable can be used only by apps under tightly controlled conditions: the kernel checks for the presence of the Apple-only dynamic code-signing entitlement. Even then, only a single mmap call can be made to request an executable and writable page, which is given a randomized address.

8.2 Cryptographic Support

8.2.1 Overview of Key Management

Each TOE comes with a unique 256-bit AES key called the UID. This key is stored in the Secure Enclave and is not accessible by the "regular" processor.

Even the software in the Secure Enclave cannot read the UID. It can only request encryption and decryption operations performed by a dedicated AES engine accessible only from the Secure Enclave.

The UID itself is generated outside of the device in the production environment using a protected system with a random number generator that complies with the requirements of NIST Special Publication 800-90A and is seeded appropriately. (FCS_RBG_EXT.1.1 and FCS_RBG_EXT.1.2 for the random number generator used to generate the REK).

The UID is used to derive two other keys, called "key 0x89B" and "key 0x835" in accordance with the requirements of NIST SP 800-38F. Both keys are derived during the first boot by encrypting defined constants with the UID. Those keys then are used to wrap two other keys: the "EMF key" (the file system master key, wrapped by "key 0x89B") and the "DKey" (the device key, wrapped by "key 0x835"). Both are stored in block 0 of the flash memory, which is also called the "effaceable storage". This area of flash memory can be wiped very quickly. Both keys are generated using the random number generator of the secure enclave (post-processed by the CTR_DRBG) when iOS is first installed or after the device has been wiped.

All keys are generated using an internal entropy source, post-processed by a deterministic random number generator (DRNG) (CTR_DRBG). System entropy is generated from timing variations during boot, and additionally from interrupt timing once the device has booted. Keys generated inside the Secure Enclave use its true hardware random number generator based on multiple ring oscillators post processed with CTR_DRBG.

The EMF key is used as a master key used for the encryption of file system metadata. The EMF key is generated using the random number generator of the secure enclave (post-processed by the CTR_DRBG) when iOS is first installed or after the device has been wiped. Also, all class keys are generated in the secure enclave and passed to the iOS kernel in wrapped form only.

The DKey is used within the key hierarchy to directly wrap the class keys that can be used when the device is locked. For class keys that can only be used when the device is unlocked the class keys are wrapped with the XOR of the DKey and the passcode key.

Every time a file on the data partition is created, a new 256-bit AES key (the "per-file" key) is created using the hardware random number generator of the secure enclave post-processed by CTR_DBRG. Files are encrypted using this key with AES in cipher block chaining (CBC) mode where the initialization vector (IV) is calculated with the block offset into the file, encrypted with the secure hash algorithm (SHA) -1 hash of the per-file key. (FCS_RBG_EXT.1 for the data encryption keys).

Each per-file key is wrapped (in the secure enclave) with the class key of the file's class and then stored in the metadata of the file. Key wrapping uses AES key wrapping per RFC 3394.

Class keys themselves are wrapped either with device key only (for the class NSFileProtectionNone) or are wrapped with a key derived from the device key and the passcode key using XOR. This key wrapping is also performed within the secure enclave.

Each file belongs to one of the following classes with its associated class key.

NSFileProtectionComplete

The class key is protected with a key derived from the user passcode and the device UID. Shortly after the user locks a device (10 seconds, if the "Require Password" setting is 'Immediately'), the decrypted class key is erased, rendering all data in this class inaccessible until the user enters the passcode again.

NSFileProtectionCompleteUnlessOpen

Some files may need to be written while the device is locked. A good example of this is a mail attachment downloading in the background. This behavior is achieved by using asymmetric elliptic curve cryptography (ECDH over Curve25519). iOS implements this by generating a device-wide asymmetric key pair and then protects the private key of this pair by encrypting it with the class key for the NSFileProtectionCompleteUnlessOpen class. Note that this class key can only be unwrapped when the device is unlocked since it requires the passcode to be entered which then is used in the key derivation function (KDF) for that generates the key encryption key (KEK) for this class key as described above. The device-wide asymmetric key pair is generated within the secure enclave.

When receiving data to be protected when the device is in the locked state, the application can create a file with the file attribute NSFileProtectionCompleteUnlessOpen. In this case iOS generates another asymmetric key pair within the secure enclave (per file object used to store the data). The device-wide public key and the file object private key are then used to generate a shared secret (using one-pass DH (Diffie-Hellman) as described in NIST SP 800-56A). The KDF is Concatenation Key Derivation Function (Approved Alternative 1) as described in 5.8.1 of NIST SP 800-56A. AlgorithmID is omitted. PartyUInfo and PartyVInfo are the ephemeral and static public keys, respectively. SHA-256 is used as the hashing function. The key generated in that fashion is used as the symmetric key to encrypt the data. The object private key and the shared secret are cleared when the file is closed and only the object public key is stored with the file object.

To read the file, the per file object shared secret is regenerated using the device-wide private key and the per file object public key.

Unwrapping of the device-wide private key can only be performed when the correct passcode has been entered, since the device-wide private key is wrapped with a key that can only be unwrapped with a class key that itself can only be unwrapped when the passcode is available. The guidance given in section D.3.3 of MDFPP Version 3 allows a key agreement scheme to be used. The key agreement scheme implemented uses using elliptic curve Diffie Hellman (ECDH) over Curve25519. When the correct passcode has been entered, the files with sensitive data received while the device was in the locked state get the per-file key re-wrapped with the NSFileProtectionCompleetUnlessOpenclass key. It is up to the application to check when the device is unlocked and then cause iOS to re-wrap the file encryption key with the class key for the NSFileProtectionComplete class by changing the file's NSFileProtectionKey attribute to NSFileProtectionComplete.

Protected Until First User Authentication

This class behaves in the same way as Complete Protection, except that the decrypted class key is not removed from memory when the device is locked. The protection in this class has similar properties to desktop full-volume encryption and protects data from attacks that involve a reboot. This is the default class for all third-party app data not otherwise assigned to a Data Protection class.

NSFileProtectionNone

This class key is wrapped only with the device key and is kept in Effaceable Storage. Since all the keys needed to decrypt files in this class are stored on the device, the encryption only affords the benefit of fast remote wipe. If a file is not assigned a Data Protection class, it is still stored in encrypted form (as is all data on an iOS device).

Keychain data is protected using a class structure similar to the one used for files. Those classes have behaviors equivalent to the file Data Protection classes but use distinct keys.

In addition, there are keychain classes with the additional extension "ThisDeviceOnly". Class keys for those classes are wrapped with a key that is also derived from the Device Key which, when copied from a device during backup and restored on a different device will make them useless.

The keys for both file and keychain Data Protection classes are collected and managed in keybags. iOS uses the following four keybags: system, backup, escrow, and iCloudBackup. The keys stored in the System keybag and some keys stored in the Escrow keybag, which are used for device update and by MDM, are relevant for functions defined in [PP_MD_V3.1].

The system keybag is where the wrapped class keys used in normal operation of the device are stored. For example, when a passcode or biometric authentication factor is entered, the NSFileProtectionComplete key is loaded from the system keybag and unwrapped. It is a binary plist stored in the No Protection class, but whose contents are encrypted with a key held in Effaceable Storage. In order to give forward security to keybags, this key is wiped and regenerated each time a user changes their passcode.

The AppleKeyStore kernel extension manages the system keybag and can be queried regarding a device’s lock state. It reports that the device is unlocked only if all the class keys in the system keybag are accessible and have been unwrapped successfully.

The following table summarizes the storage for keys in persistent storage.

Key / Persistent Secret	Purpose	Storage (for all devices)
UID	REK for device Key entanglement	Secure enclave
Salt (128 bit)	Additional input to one way functions	AES encrypted in the system keybag
key 0x89B	Wrapping of EMF key	Block 0 of the flash memory. (Effaceable storage.) Secure enclave
key 0x835	Wrapping of DKey	Block 0 of the flash memory. (Effaceable storage.) Secure enclave
EMF key	A master key used for the encryption of file system metadata	Stored in wrapped form in persistent storage
NSFileProtectionCompleteUnlessOpen device-wide asymmetric key pair	Writing files while the device is locked	Stored in wrapped form in Persistent storage
CompleteUntilFirstUserAuthentication		Stored in wrapped form in persistent storage
NSFileProtectionCompleteUnlessOpen	Writing files while the device is locked: KDF static public keys	Stored in wrapped form in persistent storage
AfterFirstUnlock		Stored in wrapped form in persistent storage
AfterFirstUnlockThisDeviceOnly		Stored in wrapped form in persistent storage
WhenUnlocked		Stored in wrapped form in persistent storage
WhenUnlockedThisDeviceOnly		Stored in wrapped form in persistent storage
Dkey		Stored in wrapped form in persistent storage
NSFileProtectionNone		Stored in wrapped form in persistent storage
NSFileProtectionComplete class key	User device lock	Stored in wrapped form in persistent storage
Individual keys for files and keychains		Stored in wrapped form in persistent storage
Biometric templates (Touch ID and Face ID)		Stored in wrapped form in persistent storage

Table 6: Summary of keys and persistent secrets in iOS 11.2

8.2.1.1 Password based key derivation

The TOE implements PBKDF2 to derive a key from a user's passcode. The derived key is 256 bits in size. After the PBKDF2 operation, the derived key is entangled with the device's hardware UID key to form the root encryption key used to unwrap the user keybag holding the class keys for the file system data protection. Only when the unwrapping of the user keybag is successful, the user is considered authenticated.

The PBKDF2 is implemented as specified in SP800-132 following option 2.b. defined in section 5.4 of the mentioned standard. The password is inserted directly into the PBKDF2 function without any pre-processing. The PBKDF2 implementation uses HMAC SHA-256 as core.

8.2.2 Storage of Persistent Secrets and Private Keys by the Agent

The MD (Mobile Device) Agent calls the Apple iOS API on the device in order to store keys and persistent secrets in the keychain; which are therefore stored in wrapped form in persistent storage, as described above.

The following table summarizes the keys and persistent secrets stored for the Agent. They are used on all devices listed in this [ST].

Key / Persistent Secret	**Purpose**	**Storage (for all devices)**
TLS keys	Protecting MDM Protocol communications with the MDM Server	Stored on the device in wrapped form in persistent storage
Device Push Token	The device push token is received when registering with the Apple Push Notification Service (APNS) in order to have an unambiguous identifier in APNS.	The token is not stored on the device but sent to the MDM server. The MDM server stores it to be able to contact the device.
UDID	Unique Device ID	Stored in wrapped form in persistent storage
PushMagic	The magic string that must be included in the push notification message. This value is generated by the device.	Stored in wrapped form in persistent storage
Device identity certificate	The device presents its identity certificate for authentication when it connects to the check-in server.	Stored in wrapped form in persistent storage
Certificate Payload	https://developer.apple.com/library/ios/featuredarticles/iPhoneConfigurationProfileRef/Introduction/Introduction.html#//apple_ref/doc/uid/TP40010206-CH1-SW248	Stored in wrapped form in persistent storage
Profile encryption key	A profile can be encrypted so that it can only be decrypted using a private key previously installed on a device.	Stored in wrapped form in persistent storage
Guid	Volume Purchase Program (VPP) Account Protection A random UUID should be standard 8-4-4-4-12 formatted UUID string and	Stored in wrapped form in persistent storage

Key / Persistent Secret	Purpose	Storage (for all devices)
	must be unique for each installation of your product	

Table 7: Summary of keys and persistent secrets used by the Agent

The following figure provides an overview on the key management hierarchy implemented in iOS.

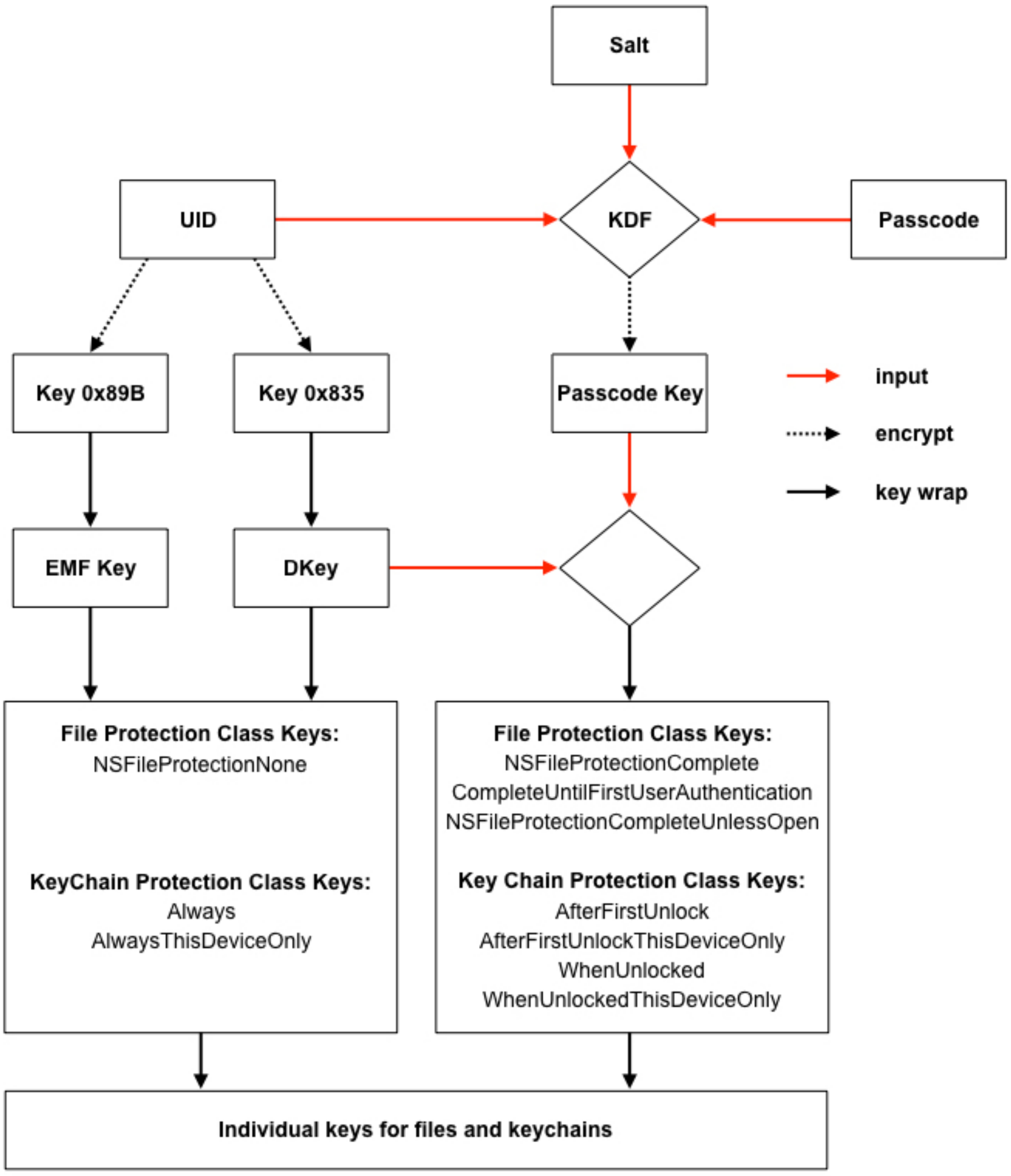
Salt
UID
KDF
Passcode
Key 0x89B
Key 0x835
Passcode Key
input
encrypt
key wrap
EMF Key
DKey
File Protection Class Keys:
NSFileProtectionNone
KeyChain Protection Class Keys:
Always
AlwaysThisDeviceOnly
File Protection Class Keys:
NSFileProtectionComplete
CompleteUntilFirstUserAuthentication
NSFileProtectionCompleteUnlessOpen
Key Chain Protection Class Keys:
AfterFirstUnlock
AfterFirstUnlockThisDeviceOnly
WhenUnlocked
WhenUnlockedThisDeviceOnly
Individual keys for files and keychains

Figure 4: Key Hierarchy in iOS

The Data Protection API can be used by applications to define the class a new file belongs to by using the NSFileProtectionKey attribute and setting its value to one of the classes described above. When the device is locked, a new file can only be created in the classes NSFileProtectionNone and NSFileProtectionCompleteUnlessOpen.

Note that the UID is not accessible by any software and that the two keys 'Key 0x89B' and 'Key 0x835' are both derived by encrypting defined values (identical for all devices) with the UID. Both are stored in the Secure Enclave. All other keys shown in the figure are stored in wrapped form in persistent storage and unwrapped when needed.

To summarize:

- All keys are managed and maintained in the Secure Enclave Processor, a dedicated execution environment with its own operating system that is completely separated from iOS. Both can interact with each other using a mailbox system detailed in section 8.1.1.
- All file system items and all key chain items are stored in encrypted form only.
- File system metadata is encrypted using the EMF key.
- Files and key chain items are encrypted with individual keys. Those keys are wrapped with the class key of the class, the file, or the keychain to which the item belongs.
- Files and key chain items belonging to the classes 'NSFileProtectionNone' (files) and 'Always' or 'AlwaysThisDeviceOnly' are encrypted with keys that are wrapped with the Dkey only. Those items can be accessed (decrypted) before the user is authenticated. For all other classes the passcode key (which is derived from the user's passcode) is used in the generation of the wrapping key used for those classes and therefore decrypting those items is only possible when the user has correctly entered his passphrase.
- All decryption errors are handled in compliance with NIST Special Publication 800-56B.
- When a wipe command is issued, protected data is wiped by erasing the top level KEKs. Since all data-at-rest is encrypted with one of those keys, the device is wiped.

iOS performs the following activities to protect the keys used for file encryption.

Every time the TOE is booted, the TOE does the following.

- An ephemeral AES key (256 bit) is created in the secure enclave using the random number generator of the secure enclave.
- The (wrapped) Dkey and (wrapped) EMF key (both 256-bit keys) are loaded by the iOS kernel from the effaceable storage and sent to the secure enclave.
- The secure enclave unwraps the Dkey and the EMF key.
- The secure enclave wraps the Dkey with the newly generated ephemeral key.
- The secure enclave stores the ephemeral key in the storage controller. This area is not accessible by the iOS kernel.

When iOS accesses a file, the following operations are performed.

- The iOS kernel first extracts the file metadata (which are encrypted with the EMF key) and sends them to the secure enclave.
- The secure enclave decrypts the file metadata and sends it back to the iOS kernel.
- The iOS kernel determines which class key to use and sends the class key (which is wrapped with the Dkey, or with the XOR of the Dkey and the Passcode Key) and the file key (which is wrapped with the class key) to the secure enclave.
- The secure enclave unwraps the file key and re-wraps it with the ephemeral key and sends this wrapped key back to the iOS kernel.
- The iOS kernel sends the file access request (read or write) together with the wrapped file key to the storage controller.
- The storage controller uses its internal implementation of AES, decrypts the file key, and then decrypts (when the operation is read) or encrypts (when the operation is write) the data during its transfer from/to the flash memory.

The following summarize the storage location for key material.

- The UID is stored in the firmware of the secure enclave in a section not accessible by any program in the secure enclave or the main processor. The processor in the secure enclave can only be used to encrypt and decrypt data (using AES256) using the UID as a key.
- Keys 0x89B and 0x835 are stored in the secure enclave.
- The EMF key, Dkey and the class keys are stored in the effaceable area, all in wrapped form only. As explained, they are never available in clear in the main processor system.
- File keys and keychain item keys are stored in non-volatile memory, but in wrapped form only. As explained they are never available in the clear in the main processor system.
- The system and the applications can store private keys in keychain items. They are protected by the encryption of the keychain item.
- Symmetric keys used for TLS, HTTPS, or Wi-Fi sessions are held in RAM only. They are generated and managed using one of the two libraries, Apple CoreCrypto Kernel Module v8 for ARM and Apple CoreCrypto Module v8 for ARM, or by the AES implementation within the Wi-Fi chip. The functions of those libraries, such as memset(0), also perform the clearing of those keys after use.

8.2.3 CAVS Certificates

The following Cryptographic Algorithm Validation System (CAVS) certificates, issued by the NIST Cryptographic Algorithm Validation Program (CAVP), apply to iOS 11.2 on the CPUs specified.

8.2.3.1 *User Space*

Row ID	SFR	Cryptographic Function	Algorithm	Modes/Options	CAVP Algorithm Certificate Number	
1	FCS_CKM_EXT.1 (MDF)	Random Number Generation; Symmetric Key Generation	[SP 800-90] DRBG	CTR_DRBG (AES Optimized Assembler Implementation) Key Size: 128-bit key size	A7	1714
	FCS_CKM_EXT.2 (MDF)				A8	1715
	FCS_CKM_EXT.6 (MDF)				A8X	1720
	FCS_RBG_EXT.1 (MDF)				A9	1717
					A9X	1716
					A10 Fusion	1721
					A10X Fusion	1718
					A11 Bionic	1719
2	FCS_CKM_EXT.3 (MDF) FCS_COP.1(1) (MDF)	Symmetric Encryption and Decryption	[FIPS 197] AES SP 800-38 A SP 800-38 D SP 800-38 E SP 800-38 F	Optimized Assembler Implementation: Key Sizes: 128/256 bits Modes: CBC, CCM, CTR, GCM, KW	A7	4870
					A8	4871
					A8X	4876
					A9	4873
					A9X	4872
					A10 Fusion	4877
					A10X Fusion	4874
					A11 Bionic	4875

3	FCS_CKM.1(1) (MDF
	FCS_CKM.2(1) (MDF)
	FCS_COP.1(3) (MDF)
		Digital Signature and Asymmetric Key Generation	[FIPS186-4]
			RSA
			PKCS #1.5
				Generic Software Implementation:
				KEY(gen)(2048/3072)
				SigGenPKCS1.5 (2048/3072)
				SigVerPKCS1.5 (1024/2048/3072)
					A7	2680

					A8	2679
					A8X	2682
					A9	2681
					A9X	2684
					A10 Fusion	2683
					A10X Fusion	2685
					A11 Bionic	2686

4	FCS_CKM.1(1) (MDF)
	FCS_CKM.2(1) (MDF)
	FCS_COP.1(3) (MDF)
			[FIPS 186-4]
			ECDSA
			ANSI X9.62
				Generic Software Implementation:
				PKG: curves P-256, P-384
				PKV: curves P-256, P-384
				SIG(gen): curves P-256, P-384 using (SHA-224, SHA-256, SHA384, SHA512)
					A7	1256

					A8	1255
					A8X	1258
					A9	1257
					A9X	1260
					A10 Fusion	1259

Row ID	SFR	Cryptographic Function	Algorithm	Modes/Options	CAVP Algorithm Certificate Number	
				SIG(ver): curves P-256, P-384 using (SHA-1, SHA-224, SHA-256, SHA384, SHA512)	A10X Fusion	1261
					A11 Bionic	1262
5	FCS_COP.1(2) (MDF)	Message Digest	[FIPS 180-4] SHS	Optimized Assembler Implementation using VNG: SHA-1, SHA-2 (256, 384, 512)	A7	4022
					A8	4021
					A8X	4024
					A9	4023
					A9X	4026
					A10 Fusion	4025
					A10X Fusion	4027
					A11 Bionic	4028
6	FCS_COP.1(4) (MDF) FCS_COP.1(5) (MDF)	Keyed Hash	[FIPS 198] HMAC	Optimized Assembler Implementation using VNG: SHA-1, SHA-2 (256, 384, 512) Key Size: at least 112 bits	A7	3282
					A8	3281
					A8X	3284
					A9	3283
					A9X	3286
					A10 Fusion	3285
					A10X Fusion	3287
					A11 Bionic	3288
7	FCS_CKM.2(2) (MDF) FCS_CKM_EXT.3 (MDF)	CVL KAS ECC	[SP800-56A] EC Diffie-Hellman Implementation follows SP800-56A for primitive only	Generic Software Implementation: 6.2.2.2 One-Pass Diffie-Hellman, C(1e, 1s, ECC CDH) Curves:P-256, P384	A7	1524
					A8	1522
					A8X	1529
					A9	1527
					A9X	1533
					A10 Fusion	1531
					A10X Fusion	1535
					A11 Bionic	1537
8	FCS_COP.1(5) (MDF)	Key Derivation	[SP 800-132] PBKDF	Password Based Key Derivation using HMAC with SHA-2	Vendor Affirmed	
9	FCS_CKM.2(1) (MDF)	RSA Key Wrapping	SP800-56B	KTS-OAEP	Vendor Affirmed	

Table 8: User Space CAVS Certificates

8.2.3.2 Kernel Space

Row ID	SFR	Cryptographic Function	Algorithm	Modes/Options	CAVP Algorithm Certificate Number	
10	FCS_COP.1(1) (MDF)	Symmetric Encryption and Decryption	[FIPS 197] AES SP800-38 A SP800-38 E SP800-38 F	Optimized Assembler Implementation: Key sizes: 128/256 bits Modes: CBC, KW	A7	4916
					A8	4917
					A8X	4920
					A9	4922
					A9X	4918
					A10 Fusion	4919
					A10X Fusion	4921
					A11 Bionic	4923
11	FCS_COP.1(2) (MDF)	Message Digest	[FIPS 180-4] SHS	Optimized Assembler Implementation using VNG: SHA-1, SHA-2 (256, 384, 512)	A7	4013
					A8	4014
					A8X	4016
					A9	4015
					A9X	4017
					A10 Fusion	4018
					A10X Fusion	4019
					A11 Bionic	4020
12	FCS_COP.1(4) (MDF)	Keyed Hash	[FIPS 198] HMAC	Optimized Assembler Implementation: KS<BS, KS=BS, KS>BS SHA-1, SHA-2 (256, 384, 512)	A7	3273
					A8	3274
					A8X	3276
					A9	3275
					A9X	3277
					A10 Fusion	3278
					A10X Fusion	3279
					A11 Bionic	3280

Table 9: Kernel Space CAVS Certificates

8.2.3.3 Broadcom Wi-Fi CAVS certificates

SFR	iOS Device Name	Broadcom Wi-Fi Chip Core #	AES CAVS Certificate Number Includes CCM, ECB, CBC, CTR modes

FCS_CKM.1(2) (WLAN)
FCS_CKM.2(3) (WLAN)
	iPhone 6	4345	4035

	iPhone 6 Plus	4345	4035
	iPhone 6s	4350	4035
	iPhone 6s Plus	4350	4035
	iPhone SE	43452	4035
	iPhone 7	4355	3678
	iPhone 7 Plus	4355	3678
	iPhone 8	4357	4152
	iPhone 8 Plus	4357	4152
	iPhone X	4357	4152
	iPad mini 4	4350	4035
	iPad mini 3	43342	N/A
	iPad Air 2	4350	4035
	iPad Pro 9.7"	4355	3678
	iPad Pro 12.9"	4355	3678
	iPad	4355	3678
	iPad Pro 10.5	4355	3678
	iPad Pro 12.9	4355	3678

Table 10: Wi-Fi CAVS Certificates

8.2.3.4 Hardware DRBG:

Row ID	SFR	Cryptographic Function	Algorithm	Modes/Options	Algorithm Certificate Number	

15	FCS_CKM.1(2) (WLAN)
	
	FCS_CKM.2(3)(WLAN)
	
	FCS_CKM_EXT.6 (MDF)
		DRBG offered by the device CPU for the Secure Enclave used by the Apple Key Store.
		
		Note that the A7, A8 and A8X devices use version 1.2 of the Secure Enclave. The other devices use version 2.0
			[SP800-90A]	CTR_DRBG	A7	2013

					A8	2020
					A8X	2021
					A9	2025
					A9X	2022
					A10 Fusion	2023
					A10X Fusion	2024
					A11 Bionic	2014
			[SP800-38A]	ECB	A7	5260
					A8	5270
					A8X	5271
					A9	5275
					A9X	5272
					A10 Fusion	5273
					A10X Fusion	5274
					A11 Bionic	5261

Table 11: Hardware DRBG CAVS Certificates

8.2.4 Randomness extraction step

“Concatenating the keys and using a KDF (as described in (SP 800- 56C)” is selected in FCS_CKM_EXT.3 Extended: Cryptographic Key Generation.

The TOE implements the KDF following the specification in RFC 5869. The KDF defined in this RFC complies with the extraction and expansion KDFs specified in SP800-56C. This RFC exactly specifies the order of the concatenation of the input data used for the extraction steps as well as the data concatenation and the counter maintenance of the expansion phase.

Extraction:

```
HKDF-Extract(salt, IKM) -> PRK

 Options:
    Hash     a hash function; HashLen denotes the length of
             the hash function output in octets

 Inputs:
    salt     optional salt value (a non-secret random value);
             if not provided, it is set to a string of
             HashLen zeros.
    IKM      input keying material

 Output:
    PRK      a pseudorandom key (of HashLen octets)

 The output PRK is calculated as follows:

 PRK = HMAC-Hash(salt, IKM)
```

Expansion:

```
HKDF-Expand(PRK, info, L) -> OKM

Options:

      Hash     a hash function; HashLen denotes the length
                 of the hash function output in octets


   Inputs:

      PRK      a pseudorandom key of at least HashLen octets

               (usually, the output from the extract step)

      info     optional context and application specific
                 information

               (can be a zero-length string)

      L        length of output keying material in octets

               (<= 255*HashLen)


   Output:

      OKM      output keying material (of L octets)


   The output OKM is calculated as follows:
```

```
N = ceil(L/HashLen)
T = T(1) | T(2) | T(3) | ... | T(N)
OKM = first L octets of T

where:
T(0) = empty string (zero length)
T(1) = HMAC-Hash(PRK, T(0) | info | 0x01)
T(2) = HMAC-Hash(PRK, T(1) | info | 0x02)
T(3) = HMAC-Hash(PRK, T(2) | info | 0x03)
...

(where the constant concatenated to the end of each T(n)
              is a single octet.)
```

The implementation of the KDF uses HMAC SHA-256 for both the extraction as well as the expansion phase.

The salt length as well as the output key length of the KDF are 256 bits, which in every case is larger than the parameters given in tables 1-3 of SP 800-56C.

8.3 User Data Protection (FDP)

The System Services available for user data protection are those of Protection of Files and Application access to Files, described below. These are applicable to all applications on the TOE which are all allowed access to these two System Services.

8.3.1 Protection of Files

When a new file is created on an iOS device, it’s assigned a class by the app that creates it. Each class uses different policies to determine when the data is accessible. As described above each class has a dedicated class key which is stored in wrapped form. Note that for the classes other than 'No Protection' to work the user must have an active passcode lock set for the device.

The basic classes and policies are described below.

Complete Protection (referred to as "class A" in some documents)

Files in this class can only be accessed when the device is unlocked.

Protected Unless Open (referred to as "class B" in some documents)

This class is for files that may need to be written while the device is locked.

Protected Until First User Authentication (referred to as "class C" in some documents)

This class is for files that are protected until the user has successfully authenticated. Unlike the 'Complete Protection' class, the class key for this class is not wiped when the device is locked, but after a re-boot the user has to authenticate before files in this class can be accessed. So, once the user has authenticated after reboot the key is available until the phone is shutdown or rebooted.

No Protection (referred to as "class D" in some documents)

Files in this class can be always accessed. Still the files themselves are encrypted using a file specific key, but this key can be unwrapped without using the passcode key derived from the user's passcode or biometric authentication factor.

Note: class A, class B and class C keys require that the user has defined a PIN. Unless he has done this only class D keys exist.

All data in files is considered private data, since all files are encrypted. Sensitive data is data protected with a class A or class B key since this data is not accessible when the device is locked.

8.3.2 Application Access to Files

An iOS app’s interactions with the file system are limited mostly to the directories inside the app’s sandbox. During installation of a new app, the installer creates a number of containers for the app. Each container has a specific role. The bundle container holds the app’s bundle, whereas the data container holds data for both the application and the user. The data container is further divided into a number of directories that the app can use to sort and organize its data. The app may also request access to additional containers—for example, the iCloud container—at runtime.

When an application is removed, all of its files are also removed.

8.3.3 Declaring the Required Device Capabilities of an App

All apps must declare the device-specific capabilities they need to run. The value of the UIRequiredDeviceCapabilities key is either an array or dictionary that contains additional keys identifying features your app requires (or specifically prohibits). If you specify the value of the key using an array, the presence of a key indicates that the feature is required; the absence of a key indicates that the feature is not required and that the app can run without it. If a dictionary is specified instead, each key in the dictionary must have a Boolean value that indicates whether the feature is required or prohibited. A value of true indicates the feature is required and a value of false indicates that the feature must not be present on the device.

8.3.4 App Groups

Apps and extensions owned by a given developer account can share content when configured to be part of an App Group. It is up to the developer to create the appropriate groups on the Apple Developer Portal and include the desired set of apps and extensions. Once configured to be part of an App Group, apps have access to the following.

- A shared container for storage, which will stay on the device as long as at least one app from the group is installed
- Shared preferences
- Shared keychain items

The Apple Developer Portal guarantees that App Group IDs are unique across the app ecosystem.

8.3.5 Restricting App Access to Services

The TOE allows a user to restrict the services an app can access. The services that can be restricted on a per-app basis are as follows.

- Location Services
- Contacts
- Calendar
- Reminders
- Photos
- Microphones
- Cameras

8.3.6 Keychain Data Protection

Many apps need to handle passwords and other short but sensitive bits of data, such as keys and login tokens. The iOS keychain provides a secure way to store these items.

The keychain is implemented as an SQLite database stored on the file system. There is only one database; the securityd daemon determines which keychain items each process or app can access. Keychain access APIs result in calls to the daemon, which queries the app's “keychain-access-groups” and the “application-identifier” entitlement. Rather than limiting access to a single process, access groups allow keychain items to be shared between apps.

Keychain items can only be shared between apps from the same developer. This is managed by requiring third-party apps to use access groups with a prefix allocated to them through the iOS Developer Program, or in iOS 11.2, via application groups. The prefix requirement and application group uniqueness are enforced through code signing, Provisioning Profiles, and the iOS Developer Program. iOS provides a user interface (UI) in the Settings dialog that allows importing of keys for use for Apple-provided applications such as Safari or VPN.

Keychain data is protected using a class structure similar to the one used in file Data Protection. These classes have behaviors equivalent to file Data Protection classes, but use distinct keys and are part of APIs that are named differently.

The following table shows the keychain classes and their equivalent file system classes.

Keychain data protection class	File data protection class
When unlocked	NSFileProtectionComplete
While locked	NSFileProtectionCompleteUnlessOpen
After first unlock	NSFileProtectionCompleteUntilFirstUserAuthentication
Always	NSFileProtectionNone

Table 12: Keychain to File-system Mapping

In addition, there are the Keychain data protection classes with the additional "ThisDeviceOnly" added to their class name. Keychain items in those classes cannot be moved to a different device using backup and restore; key chain items in those classes are bound to the device.

Among the data stored in keychain items are digital certificates used for setting up VPN connections and certificates and private keys installed by the Configuration Profile.

Keychains can use access control lists (ACLs) to set policies for accessibility and authentication requirements. Items can establish conditions that require user presence by specifying that they can't be accessed unless authenticated entering the device's passcode. ACLs are evaluated inside the Secure Enclave and are released to the kernel only if their specified constraints are met.

Further information is found in section 1.5.2.5 Protection of the TSF.

8.4 Identification and Authentication (FIA)

The user of the device is authenticated using a passcode, fingerprint, or facial authentication factor. Except for making emergency calls, using the cameras, and using the flashlight, users need to authenticate using an authentication factor provided above. All passcode entries are obscured by a dot symbol for each character as the user input occurs. Biometric authentication inputs do not produce feedback to the user unless an input is rejected. When an invalid fingerprint sample is given or cannot be authenticated, a simple error message is returned to the user to try again. If three invalid biometric samples are presented the device will offer passcode entry. After five invalid biometric samples are presented passcode authentication is required.

For Face ID, when an invalid facial sample is given or cannot be authenticated, the user needs to swipe up before a second attempt can occur and passcode entry will be presented to the user as an option. After five invalid Face ID attempts, the device will vibrate and passcode entry must be used.

The following passcode policies can be defined for managed devices.

- The minimum length of the passcode
- The minimum number of special characters a valid passcode must contain
- The maximum number of consecutive failed attempts to enter the passcode (which can be value between 2 and 10, the default is 10)
- The number of minutes for which the device can be idle before it gets locked by the system
- The maximum number of days a passcode can remain unchanged
- The size of the passcode history (the maximum value is 50)

Those parameters for the passcode policy can be defined in the Passcode Policy Payload section of a configuration profile defined by a system administrator for a managed device. For details see section 8.5, *Specification of Management Functions (FMT),* below.

Devices that support Touch ID do not support Face ID. The passcode and the device's biometric authentication method can **not** be combined for two-factor authentication. In addition, the following behavior applies to biometric authentication methods. A passcode must be supplied for additional security validation when:

- The device has just been turned on or restarted.
- The device hasn't been unlocked for more than 48 hours.
- The passcode hasn't been used to unlock the device in the last 156 hours (six and a half days) and Face ID hasn't unlocked the device in the last 4 hours.
- The device has received a remote lock command.
- After five unsuccessful attempts to match.
- After initiating power off/Emergency SOS.

The number of failed authentication attempts is maintained in a system file which will persist in the event of graceful or ungraceful loss of power to the TOE. The counter maintaining the number of failed consecutive logon attempts is increased by one immediately once the TOE has identified that the passcode is incorrect. The increment of the counter is completed before the UI informs the user about the failed logon attempt.

The time between consecutive authentication attempts, including biometric authentication factors, is at least the time it takes the PBKDF2 function to execute. This is calibrated to be at least 80 milliseconds. In addition, for Touch ID, the TOE enforces a 5 second delay between repeated failed authentication attempts. When a user exceeds the number of consecutive failed passcode login attempts, the user's partition is erased (by erasing the encryption key). The OS partition is mounted READONLY upon boot and is never modified during the use of the TOE except during a software update or restore. Note that entering the same incorrect passcode multiple times consecutively causes the failed login counter to increment only once for those multiple attempts even though these are all failed login attempts.

Additionally, authentication credentials which include biometric samples are not stored on the TOE in any location. Successful authentication attempts are achieved exclusively by a successful key derivation that decrypts the keybag in the Secure Enclave Processor with the respective class keys.

8.4.1 Biometric Authentication

8.4.1.1 Accuracy of Biometric Authentication

All validation of biometric authentication factors is conducted in the form of an off-line test. Fingerprint data used in these tests were collected separately for each sensor generation using multiple devices, the production version of the sensor and its firmware. The software (i.e. primarily the biometric algorithms) is based on production code corresponding to the given iOS release.

For efficiency reasons the test is not run on the production hardware, but it is instead emulated on a different platform in a cloud computation infrastructure. A special testing step is performed to assure equality of results between the emulated and production run on the same input data.

The False Acceptance Rate (FAR), which is further defined in [PP_MD_V3.1], test protocols differ between sensor generations as follows.

- For gen.1 a full cross-comparison scheme is used. Each user is enrolled and each template is attacked by all other user data.
- For gen.2 and gen.3 a partial cross-comparison is done. In this scheme test population is split between users and imposters. Users are enrolled and each template is attacked by all imposter data.

Validation of Face ID follows the methodology established for Touch ID also using offline testing. The FAR was evaluated by full cross-comparison of all subjects in the P3 dataset. The P3 dataset contained fully labeled data.

Data was collected from a wide range of subjects. Facial expression variations were collected from each subject as well as variations in environmental factors. Variations in subject age, ethnicity, and gender were also introduced into the dataset as well as subjects that exhibited familial relationships such as siblings. Offline testing was performed with data that simulates a normal presentation -- near frontal view, no obstructions, within nominal range (20-45 cm).

8.4.1.2 Rule of 3 Discussion

The size of the test population differs between sensor generations of Touch ID and also for Face ID, but in all cases, there are enough unique subjects to fulfill the Rule of 3.

The actual tests are run on datasets covering more fingers per subject and also more attempts per impostor. Therefore, the actual number of imposter attempts is much greater than number of unique attempts. Each imposter has a similar contribution to the result. We assume that in combination with, for Touch ID, a small sensor area this approach improves quality and variability of the database.

8.4.1.3 Accuracy of Biometric Authentication – Continued

The overall System Authentication False Acceptance Rate (SAFAR), which is further defined in [PP_MD_V3.1] is derived from tests run for FIA_BMG_EXT.1.1 as noted in section 8.4.1.1 above.

Since iOS 11.2 allows the use of 5 fingerprint attempts or 10 attempts of 6-digit passcode, the value of SAFAR is dominated by fingerprint SAFAR for 5 attempts.

Similarly, iOS 11.2 allows the use of 5 facial identification attempts or 10 attempts of 6-digit passcode, value of SAFAR is dominated by Face ID SAFAR for 5 attempts.

Since each mobile device contains no more than one BAF, the interaction of BAFs is not an issue for the calculation.

8.4.1.4 Biometric Sample Quality

In iOS 11.2, sample quality is inspected before it is passed to the matcher algorithm for both Touch ID and Face ID. In general, the inspection is based on the following.

For Touch ID:

- Deciding what portion of the sensor is covered by the finger. Sensor regions containing very weak signal are considered not covered. Samples with high number of such regions are rejected.
- Assessing the level of residual fixed-pattern noise. Samples where the noise could significantly alter the fingerprint pattern are rejected.
- Detection and removal of regions affected by image discontinuity caused by finger motion. Samples with many regions affected by motion are rejected.

For Face ID:

- User is attending the device
- No significant depth holes in the depth map
- Anti-Spoofing network to reject physical spoofs
- For enrollment
    - No occlusions detected (e.g. hand covering face)
    - Face within certain pose angles
    - User attending device
    - No significant depth holes in the depth map
    - Anti-spoofing network to reject physical spoofs

The validation of the discussed mechanism is performed regularly for each major iOS release. The test is based on specialized datasets containing different levels of coverage and different artifacts. These samples are fed to the biometric system and it is confirmed whether the sample is correctly passed or rejected from the processing as expected.

Additionally, the biometric system is tested by feeding artificially created images containing different geometric patterns.

8.4.2 Certificates

There are a number of certificates used by the TOE. First there is the Apple certificate used to verify the integrity and authenticity of software updates. This certificate is installed in ROM during manufacturing.

Other certificates used for setting up trusted channels or decrypt/verify protected e-mail can be imported by a user (if allowed by the policy) or installed using configuration profiles.

Certificates can be installed for the following.

- IPSec
- TLS
- EAP-TLS, other supported EAP protocols

Note that only IPSec, TLS, and EAP-TLS are addressed by [PP_MD_V3.1]. Certificates have a certificate type that defines their respective application area. This ensures that only certificates defined for a specific application area are used. In addition, the database containing trust anchors for all certificates is protected via integrity check and write protection. The certificate types supported by the TOE are as follows.

- AppleX509Basic
- AppleSSL
- AppleSMIME
- AppleEAP
- AppleIPsec
- AppleCodeSigning
- AppleIDValidation
- AppleTimeStamping

External entities can be authenticated using a digital certificate. Out of the box, iOS includes a number of preinstalled root certificates.

Code signing certificates need to be assigned by Apple and can be imported into a device. The issue of such a certificate can be by app developers or by enterprises that want to deploy apps from their MDM to managed devices. All apps must have a valid signature that can be verified by a code signing certificate before they are installed on a device.

iOS can update certificates wirelessly, if any of the preinstalled root certificates become compromised. To disable this, there is a restriction that prevents over-the-air certificate updates.

The list of supported certificate and identity formats are:

- X.509 certificates with RSA keys, and
- File extensions .cer, .crt, .der, .p12, and .pfx.

To use a root certificate that isn't preinstalled, such as a self-signed root certificate created by the organization managing the TOE, they can be distributed using one of the following methods.

- When reviewed and accepted by the user
- Using the configuration profile
- Downloaded from a web site

When attempting to establish a connection using a remote certificate, the certificate is first checked to ensure it is valid. Certificates are validated against the common name (CN) portion of the distinguished name (DN). If the CN does not match the corresponding domain name system (DNS) or IP Address of the server being accessed, validation and subsequently the connection will fail. If the certificate is valid, the attempt to establish the connection continues. If the certificate is invalid, the next step is up to the application. The application should provide an indication to the user that the certificate is invalid and options to accept or reject.

TLS is implemented as a stack that can be utilized by third-party applications. The API informs the calling application that the certificate is not valid. For example, Safari will display a message to the user that the remote certificate validation failed and allow the user to examine the certificate with the option to allow the connection or not.

iOS can be configured to disable the user option to accept invalid TLS certificates using the "Allow user to accept untrusted TLS certificates" setting.

8.4.3 MDM Server Reference ID

The initial MDM Payload contains a mandatory ServerURL string. The URL that the device contacts to retrieve device management instructions must begin with the https:// URL scheme and may contain a port number (for example ":1234").

The MDM check-in protocol is used during initialization to validate a device's eligibility for MDM enrollment and to inform the MD server that a device's Push Token has been updated. If a check-in server URL is provided in the MDM payload, the check-in protocol is used to communicate with that check-in server. If no check-in server URL is provided, the main MDM Server URL is used instead.

A managed Mobile Device uses an identity to authenticate itself to the MDM Server over TLS (Secure Sockets Layer (SSL)). This identity can be included in the profile as a Certificate payload or can be generated by enrolling the device with Simple Certificate Enrollment Protocol (SCEP)[4]. Each MDM Server must be registered with Apple at the MDM Server Device Enrollment Program (DEP) management portal. The DEP provides details about the server entity to identify it uniquely throughout the organization deploying the MDM Server. Each server can be identified by either its system-generated UUID or by a user-provided name assigned by one of the organization's users. Both the UUID and server name must be unique within the organization. Registered MDM servers can include third party servers. iOS devices automatically connect to the MDM Server during setup if the device is enrolled into the DEP and is assigned to an MDM Server. During the device enrollment, the MDM

[4] More information about SCEP can be found at https://datatracker.ietf.org/doc/draft-nourse-scep/

enrollment service returns a JavaScript Object Notation (JSON) dictionary with the keys to mobile devices shown in the following table.

Key	Value
server_name	An identifiable name for the MDM Server
server_uuid	A system-generated server identifier
admin_id	Apple ID of the person who generated the current tokens that are in use
facilitator_id	Legacy equivalent to the admin_id key. This key is deprecated and may not be returned in future responses.
org_name	The organization name
org_email	The organization email address
org_phone	The organization phone
org_address	The organization address

Table 13: MDM Server Reference Identifiers

8.5 Specification of Management Functions (FMT)

In FMT_SMF_EXT.3.1 “no additional functions" is selected and so these are not documented in this [ST] as supported by the platforms.

Since all the Mobile Devices specified in this [ST] use iOS, there are no differences between supported management functions and policies between the different mobile devices. The supported management functions for iOS are described in [iOS_CFG]

Table 5: Management Functions, describes all management functions of the devices as well as the MDM Agent that are available when the device is enrolled in MDM.

8.5.1 Enrollment

The methods by which an MDM Agent can be enrolled are as follows.

- Manually, using Apple's Profile Manager
- Manually, using Apple Configurator 2
- Distributing an enrollment profile via email, or a web site
- Device Enrollment Program (This is an automated and enforced method of automatically enrolling new devices.)

A more detailed description is found in section 8.4.3, above. In addition, the enrollment process is discussed in the MDM Protocol reference [iOS_MDM].

8.5.2 Configuration Profiles

The TOE can be configured using "Configuration Profiles" that are installed on the TOE. Configuration Profiles are XML files that may contain settings for a number of configurable parameters. For details on the different payloads and keys that can be defined see the document "Configuration Profile Key Reference" ([iOS_CFG]) that can be downloaded from the Apple web site.

Configuration profiles can be deployed in one of five different ways:

- using the Apple Configurator tool,
- via an email message,
- via a web page,
- using over-the-air configuration as described in [iOS OTAConfig], and

- using over-the-air configuration via a MDM Server.

To preserve the integrity, authenticity, and confidentiality of Configuration Profiles they can be required to be digitally signed and encrypted.

Managed items relevant for this [ST] that have to be configured using Configuration Profiles are as follows.

- The password policy—the administrator can define this using the Passcode Policy Payload and
  - define the minimum password length,
  - define requirements for the password complexity,
  - define the maximum password lifetime,
  - define the maximum time of inactivity after which the device is locked automatically, and
  - define the maximum number of consecutive authentication failures after which the device is wiped.
- The VPN policy as follows:
  - if VPN is always on,
  - define the authentication method (Shared Secret or Certificate), and
  - specification of certificates or shared keys.
- The Wi-Fi policy as follows:
  - the EAP types allowed,
  - the service set identifiers (SSIDs) allowed to connect to,
  - the encryption type(s) allowed, and
  - enabling/disabling Wi-Fi hotspot functionality.
- General restrictions as follows:
  - allowing or disallowing specific services (e. g. remote backup) or using devices like the cameras,
  - allowing or disallowing notifications when locked, and
  - allowing or disallowing a prompt when an untrusted certificate is presented in a TLS/HTTPS connection.

Note that the microphones cannot be disabled in general but a user can restrict access to the microphones on a per-app basis.

Other functions that can be enabled/disabled by an administrator are:

- the installation of applications by a user,
- the possibility to perform backups to iCloud,
- the ability to submit diagnostics automatically,
- the ability to use the fingerprint device (Touch ID) for user authentication,
- the ability to see notifications on the lock screen,
- the ability to take screen shots,
- the ability to accept untrusted TLS certificates, and

- the ability to perform unencrypted backups (via iTunes).

Further restrictions can be enforced for enrolled devices. Those include:

- the ability to modify the account,
- the ability to modify the cellular data usage,
- the ability to pair with a host other than the supervision host,
- the ability for the user to install configuration profiles or certificates interactively, and
- the ability for the user to use 'Enable Restrictions' interface.

A user can access management functions available to him via the menus of the graphical user interface. The functions he can perform are described in [iPhone_UG] and [iPad_UG].

Configuration profiles can also be deployed such that users are unable to override or remove restrictions set in place by Administrators or MDM Administrators. Depending on the behavior defined in the configuration profile, users will be unable to access, perform, or relax management functions defined in Table 5: Management Functions. In the most restrictive mode, users will not be able to access the options to alter the above functionality at all. In less restrictive modes, the user is only able to select more secure options.

8.5.3 Biometric Authentication Factors (BAFs)

Enrollment and management of biometric authentication factors and credentials is detailed in the [iPhone_UG] and [iPad_UG] respectively.

Enrollment for Touch ID is typically accomplished during initial device configuration but can also be performed using the Settings»Touch ID & Passcode menus. Multiple fingerprints may be enrolled, named, and deleted from this menu. In order to remove a specific finger, a user must tap the finger for removal followed by delete fingerprint. Users may place a finger on the Touch ID sensor to determine which biometric credential entry it is mapped. Users may also disable Touch ID selectively for applications or entirely from the Settings»Touch ID & Passcode menu and turning off one or more of the corresponding options.

- Unlock
- Apple Pay
- ITunes & App Store

Enrollment for Face ID is typically accomplished during initial device configuration but can also be performed using the Settings»Face ID & Passcode menu by selecting the “Set up Face ID” option. Face ID does not support multiple users and only one individual may establish a Face ID credential by providing a biometric sample for enrollment. Users may remove Face ID biometric samples from the Settings»Face ID & Passcode and selecting the “Reset Face ID” option. Users may also disable Face ID selectively for applications or entirely from the Settings»Touch ID & Passcode menu and turning off one or more of the corresponding options.

- Unlock
- Apple Pay
- iTunes & App Store
- Safari AutoFill

When enrolling, naming, and deleting BAFs the passcode must be successfully entered before changes can be made.

8.5.4 Unenrollment

The Configuration Profile Key Reference, [iOS_CFG], describes unenrollment options through specifying the key "PayloadRemovalDisallowed". This is an optional key. If present and set to true the user cannot delete the profile unless the profile has a removal password and the user provides it.

It is up to the MDM Server to ensure that this key is set appropriately.

In supervised mode the MDM payload can be locked to the device.

In addition, [iOS_MDM] describes the additional ability to restrict the installation and removal of configuration profiles from other sources. This is achieved using the AccessRights key which has a value of a logical OR of the following bits.

Required

- 1—Allow inspection of installed configuration profiles
- 2—Allow installation and removal of configuration profiles
- 4—Allow device lock and passcode removal
- 8—Allow device erase
- 16—Allow query of Device Information (device capacity, serial number)
- 32—Allow query of Network Information (phone/SIM numbers, MAC addresses)
- 64—Allow inspection of installed provisioning profiles
- 128—Allow installation and removal of provisioning profiles
- 256—Allow inspection of installed applications
- 512—Allow restriction-related queries
- 1024—Allow security-related queries
- 2048—Allow manipulation of settings
- 4096—Allow app management

Note that the AccessRights key may not be zero. If bit 2 is specified, then bit 1 must also be specified. If bit 128 is specified, then bit 64 must also be specified.

8.5.5 Radios

The following radios are found in the TOE:

- Cellular
- WiFi
- Bluetooth

These are fully described, including the frequencies employed, in Table 1: Devices Covered by the Evaluation and Table 2: Cellular Protocols Supported.

As indicated in Table 5: Management Functions, users can enable/disable these radios.

8.5.6 Collection devices

The following collection devices are found in the TOE:

- Cameras
- Microphones

Table 5: Management Functions describes the roles that can enable/disable them.

8.6 Protection of the TSF (FPT)

8.6.1 Secure Boot

Each step of the startup process contains components that are cryptographically signed by Apple to ensure integrity and that proceed only after verifying the chain of trust. This includes the bootloaders, kernel, kernel extensions, and baseband firmware.

When an iOS device is turned on, its application processor immediately executes code from read-only memory known as the Boot ROM. This immutable code, known as the hardware root of trust, is laid down during chip fabrication, and is implicitly trusted. The Boot ROM code contains the Apple Root CA public key, which is used to verify that the Low-Level Bootloader (LLB) is signed by Apple before allowing it to load. This is the first step in the chain of trust where each step ensures that the next is signed by Apple. When the LLB finishes its tasks, it verifies and runs the next-stage bootloader, iBoot, which in turn verifies and runs the iOS kernel.

This secure boot chain helps ensure that the lowest levels of software are not tampered with and allows iOS to run only on validated Apple devices.

For devices with cellular access, the baseband subsystem also utilizes its own similar process of secure booting with signed software and keys verified by the baseband processor.

For devices with an A7 or later A-series processor, the Secure Enclave coprocessor also utilizes a secure boot process that ensures its separate software is verified and signed by Apple.

If one step of this boot process is unable to load or verify the next process, startup is stopped and the device displays the “Connect to iTunes” screen. This is called recovery mode. If the Boot ROM is not able to load or verify LLB, it enters Device Firmware Upgrade (DFU) mode. In both cases, the device must be connected to iTunes via USB and restored to factory default settings.

8.6.2 Joint Test Action Group (JTAG) Disablement

The mobile devices of iPhones and iPads use a 2-staged interface that resembles the functionality of JTAG but does not implement the JTAG protocol.

The Apple development environment that is JTAG-like is based on the following.

- To use this JTAG-like interface, a development-fused device is required. This implies that certain hardware fuses were not blown during the manufacturing process. Only with these development-interface related fuses intact, the JTAG-like interface is technically reachable.
- When having a development-fused device, the Apple developers are given a special Lightning cable that contains some additional computing logic. This cable establishes a serial channel with the mobile device's JTAG-like interface

reachable on development-fused devices. This Lightning cable connects to the development machine's USB port and allows subsequent access by development tools. The serial link allows access to the serial console of the mobile device. The serial console, however, does not allow access on a production fused device. On a development-fused device, the root account is enabled and an SSH server is listening. The SSH server is accessible via the serial link and allows the developer to access the root account for development including uploading of software or modifying of installed software.

8.6.3 Secure Software Update

Software updates to the TOE are released regularly to address emerging security concerns and also provide new features; these updates are provided for all supported devices simultaneously. Users receive iOS update notifications on the device and through iTunes, and updates are delivered wirelessly, encouraging rapid adoption of the latest security fixes.

The startup process described above helps ensure that only Apple-signed code can be installed on a device. To prevent devices from being downgraded to older versions that lack the latest security updates, iOS uses a process called System Software Authorization. If downgrades were possible, an attacker who gains possession of a device could install an older version of iOS and exploit a vulnerability that’s been fixed in the newer version.

On a device with an A7 or later A-series processor, the Secure Enclave coprocessor also utilizes System Software Authorization to ensure the integrity of its software and prevent downgrade installations.

iOS software updates can be installed using iTunes or over-the-air (OTA) on the device via HTTPS trusted channel. With iTunes, a full copy of iOS is downloaded and installed. OTA software updates download only the components required to complete an update, improving network efficiency, rather than downloading the entire OS. Additionally, software updates can be cached on a local network server running the caching service on OS X Server so that iOS devices do not need to access Apple servers to obtain the necessary update data.

During an iOS upgrade, iTunes (or the device itself, in the case of OTA software updates) connects to the Apple installation authorization server and sends it a list of cryptographic measurements for each part of the installation bundle to be installed (for example, LLB, iBoot, the kernel, and OS image), a random anti-replay value (nonce), and the device’s unique ID (ECID).

The authorization server checks the presented list of measurements against versions for which installation is permitted and, if it finds a match, adds the ECID to the measurement and signs the result. The server passes a complete set of signed data to the device as part of the upgrade process. Adding the ECID “personalizes” the authorization for the requesting device. By authorizing and signing only for known measurements, the server ensures that the update takes place exactly as provided by Apple.

The boot-time, chain-of-trust evaluation verifies that the signature comes from Apple and that the measurement of the item loaded, combined with the device’s ECID, matches what was covered by the signature.

These steps ensure that the authorization is for a specific device and that an old iOS version from one device can’t be copied to another. The nonce prevents an attacker from saving the server’s response and using it to tamper with a device or otherwise alter the system software.

Note that this ensures the integrity and authenticity of software updates. A TLS trusted channel is provided for this process.

8.6.4 Security Updates

Apple generally does not disclose, discuss, or confirm security issues until a full investigation is complete and any necessary patches or releases are available. Once an issue has been confirmed and a patch has been made available references containing technical details on the patches / Common Vulnerabilities and Exposures (CVEs), etc. are released. Apple also distributes information about security issues in its products through security advisories. Advisories are provided through the security-announce mailing list. Resources include the following.

Contact Apple About Security Issues
https://support.apple.com/en-us/HT201220

Apple Security Updates (Advisories)
https://support.apple.com/en-us/HT201222

Security-Announce Mailing List (receive Apple security advisories through)
https://lists.apple.com/mailman/listinfo/security-announce/

8.6.5 Domain Isolation

The TOE does not support Unstructured Supplementary Service Data (USSD) or Man-Machine Interface (MMI) codes and also does not support auxiliary boot modes.

All applications are executed in their own domain or 'sandbox' which isolates the application from other applications and the rest of the system5. Stack-based buffer overflow protection is implemented for every sandbox. They are also restricted from accessing files stored by other applications or from making changes to the device configuration. Each application has a unique home directory for its files, which is randomly assigned when the application is installed. If a third-party application needs to access information other than its own, it does so only by using services explicitly provided by iOS.

System files and resources are also shielded from the user’s apps. The majority of iOS runs as the non-privileged user “mobile,” as do all third-party apps. The entire OS partition is mounted as read-only. Unnecessary tools, such as remote login services, aren’t included in the system software, and APIs do not allow apps to escalate their own privileges to modify other apps or iOS itself.

Access by third-party apps to user information and features is controlled using declared entitlements. Entitlements are key value pairs that are signed in to an app and allow authentication beyond runtime factors like a UNIX user ID. Since entitlements are digitally signed, they cannot be changed. Entitlements are used extensively by system apps and daemons to perform specific privileged operations that would otherwise require the process to run as root. This greatly reduces the potential for privilege escalation by a compromised system application or daemon.

Address space layout randomization (ASLR) protects against the exploitation of memory corruption bugs. All TSF binaries and libraries use ASLR to ensure that all memory regions are randomized upon launch. Xcode, the iOS development environment, automatically compiles third-party programs with ASLR support turned on. Address space layout randomization is used for every sandbox used to execute

[5] This means that the inventory of TSF binaries and libraries, indicating those that implement stack-based buffer overflow protections as well as those that do not is effectively a null list, since all the TSF binaries and libraries are protected from stack-based buffer overflow.

applications in. There are 8 bits of randomness taken from the application processor TRNG involved in the randomization, the seed for the RNG comes from the seed that also feeds the approved DRBG for cryptographic use.

In addition, The Memory Management Unit (MMU) supports memory address translation using a translation table maintained by the OS kernel. For each page, the MMU maintains flags that allow or deny the read, write or execution of data. Execution in this case allows the CPU to fetch instructions from a given page.

8.6.6 Device Locking

The TOE is locked after a configurable time of user inactivity or upon request of the user. When the device is locked, the class key for the 'Complete Protection' class is wiped 10 seconds after locking, making files in that class inaccessible. This also applies for the class key of the 'Accessible when unlocked' keychain class.

The lock screen of a device can be defined and set for supervised devices by an administrator using Apple Configurator or an MDM.

The TOE can be locked remotely either via the iCloud "Lost Mode" function or by an MDM system if the device is enrolled in management.

8.6.7 Time

The following security functional requirements make use of time:

- ALC_TSU_EXT
- FAU_GEN.1.2
- FIA_UAU.7
- FIA_X509_EXT.1.1
- FIA_X509_EXT.3.1
- FMT_SMF_EXT.1.1 Function 1:
- FMT_SMF_EXT.1.1 Function 2:
- FPT_STM.1.1
- FTA_SSL_EXT.1

When the device starts and the "Set Automatically" setting is set on the device the following services are used to synchronize the real-time clock on the device:

For A7 and later platforms the devices set time by:

1. GPS.
2. If GPS is unavailable, NTP will be used.

If the "Set Automatically" setting is not set on the device the user can manually set the clock using an appropriate reference facility.

When configured and maintained according to the Network, Identity and Time Zone (NITZ), Global Positioning Satellites (GPS), Network Time Protocol (NTP) standards or the cellular carrier time service the time may be considered reliable.

8.6.8 Inventory of TSF Binaries and Libraries

All user space binaries (applications as well as shared libraries) are subject to address space layout randomization. The logic is implemented in the binary loader and agnostic of a particular file or its contents. An inventory of TSF binaries and libraries has been provided to NIAP since the list is considered proprietary.

8.6.9 Self-Tests

Self-tests are performed by the Apple CoreCrypto modules for ARM.

8.6.9.1 Apple iOS CoreCrypto Module V8 for ARM

The module performs self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition, the random bit generator requires continuous verification. The FIPS Self Tests application runs all required module self-tests. This application is invoked by the iOS startup process upon device power on.

The execution of an independent application for invoking the self-tests in the libcorecrypto.dylib makes use of features of the iOS architecture: the module, implemented in libcorecrypto.dylib, is linked by libcommoncrypto.dylib which is linked by libSystem.dylib. The libSystem.dylib is a library that must be loaded into every application for operation. The library is stored in the kernel cache and therefore is not available in the file system as directly visible files. iOS ensures that there is only one physical instance of the library and maps it to all application linking to that library. In this way, the module always stays in memory. Therefore, the self-test during startup time is sufficient as it tests the module instance loaded in memory which is subsequently used by every application on iOS.

All self-tests performed by the module are listed and described in this section.

Power-Up Self Tests

The following tests are performed each time the Apple CoreCrypto Module v8 for ARM starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the following tests fail, the device powers itself off. To rerun the self-tests on demand, the user must reboot the device.

Cryptographic Algorithm Tests

Algorithm	Modes	Test
AES implementations selected by the module for the corresponding environment AES-128	ECB, CBC, GCM, XTS	KAT Separate encryption / decryption operations are performed
DRBG (CTR_DRBG and HMAC_DRBG; tested separately)	N/A	KAT
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT
RSA	SIG(ver), SIG(gen)	Pair-wise consistency checks
	Encrypt / decrypt	KAT, Separate encryption /decryption operations are performed
ECDSA	Signature Generation, Signature Verification	Pair-wise consistency checks
Diffie-Hellman “Z” computation	N/A	KAT
EC Diffie-Hellman “Z” computation	N/A	KAT

Table 14: Apple CoreCrypto Module v8 for ARM Cryptographic Algorithm Tests

Software / Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple iOS CoreCrypto Module, v8 for ARM. The CoreCrypto's HMAC-SHA-256 is used as an Approved algorithm for the integrity test. If the test fails, then the device powers itself off.

Critical Function Tests

No other critical function test is performed on power up.

Conditional Tests

The following sections describe the conditional tests supported by the Apple CoreCrypto Module v8 for ARM.

- **Continuous Random Number Generator Test**
  The Apple CoreCrypto Module v8 for ARM performs a continuous random number generator test, whenever CTR_DRBG is invoked.
- **Pair-wise Consistency Test**
  The Apple CoreCrypto Module v8 for ARM does generate asymmetric keys and performs all required pair-wise consistency tests, the encryption/decryption as well as signature verification tests, with the newly generated key pairs.
- **SP 800-90A Assurance Tests**
  The Apple CoreCrypto Module v8 for ARM performs a subset of the assurance tests as specified in section 11 of SP 800-90A, in particular it complies with the mandatory documentation requirements and performs know-answer tests and prediction resistance.
- **Critical Function Test**
  No other critical function test is performed conditionally.

8.6.9.2 Apple iOS CoreCrypto Kernel Module V8 for ARM

The module performs self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. In addition, the DRBG requires continuous verification. The FIPS Self Tests functionality runs all required module self-tests. This functionality is invoked by the iOS Kernel startup process upon device initialization. If the self-tests succeed, the Apple CoreCrypto Cryptographic module v8 for ARM instance is maintained in the memory of the iOS Kernel on the device and made available to each calling kernel service without reloading. All self-tests performed by the module are listed and described in this section.

Power-Up Tests

The following tests are performed each time the Apple CoreCrypto Cryptographic module v8 for ARM starts and must be completed successfully for the module to operate in the FIPS Approved Mode. If any of the following tests fails the device shuts down automatically. To run the self- tests on demand, the user may reboot the device.

Cryptographic Algorithm Tests

Algorithm	Modes	Test
AES implementations selected by the module for the corresponding environment AES-128	ECB, CBC, XTS	KAT Separate encryption / decryption operations are performed
DRBG (CTR_DRBG and HMAC_DRBG; tested separately)	N/A	KAT
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	N/A	KAT
ECDSA	Signature Generation, Signature Verification	pair-wise consistency test
RSA	Signature Verification	KAT

Table 15:Apple CoreCrypto Kernel Module V8 for ARM Cryptographic Algorithm Tests

Software / Firmware Integrity Tests

A software integrity test is performed on the runtime image of the Apple CoreCrypto Kernel Module, v8 for ARM. The CoreCrypto's HMAC-SHA256 is used as an Approved algorithm for the integrity test. If the test fails, then the device powers itself off.

Critical Function Tests

No other critical function test is performed on power up.

Conditional Tests

The following sections describe the conditional tests supported by the Apple CoreCrypto Cryptographic module v8 for ARM.

- **Continuous Random Number Generator Test**
  The Apple CoreCrypto Cryptographic module v8 for ARM performs a continuous random number generator test, whenever CTR_DRBG is invoked. In addition, the seed source implemented in the operating system kernel also performs a continuous self-test.
- **Pair-wise Consistency Test**
  The Apple CoreCrypto Cryptographic module v8 for ARM generates asymmetric ECDSA key pairs and performs all required pair-wise consistency tests (signature generation and verification) with the newly generated key pairs.
- **SP800-90A Assurance Tests**
  The Apple CoreCrypto Cryptographic module v8 for ARM performs a subset of the assurance tests as specified in section 11 of SP800-90A, in particular it complies with the mandatory documentation requirements and performs know-answer tests and prediction resistance.
- **Critical Function Test**
  No other critical function test is performed conditionally.

8.7 TOE Access (FTA)

8.7.1 Session Locking

iOS devices can be configured to transit to a locked state after a configurable time interval of inactivity. This time can be defined by an administrator using a Configuration Profile.

Displaying notifications when in the locked state can be prohibited via the allowLockScreenNotificationsView key in the Restrictions Payload of a Configuration Profile.

8.7.2 Restricting Access to Wireless Networks

Users and administrators can restrict the wireless networks an iOS device connects to. Using the configuration profile administrators can define the wireless networks the device is allowed to connect to and the EAP types allowed for authentication. This also includes the following attributes:

- Specification of the CA(s) from which the TSF will accept WLAN authentication server certificates(s)
- Security type
- Authentication protocol
- Client credentials to be used for authentication

For the list of radios supported by each device see Table 1: Devices Covered by the Evaluation and Table 2: Cellular Protocols Supported. The standards listed there define the frequency ranges.

8.7.3 Lock Screen Banner Display

An advisory warning message regarding unauthorized use of the TOE can be defined using an image that is presented during the lock screen. Configuration for this is described in FMT_SMF_EXT.1.1 Function 36.

Since the banner is an image there are no character limitations, information is restricted to what can be included in an image appropriate to the device display.

8.8 Trusted Path/Channels (FTP)

The TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of 802.11-2012, 802.1X, and EAP-TLS, TLS, and IPsec.

IPsec supports authentication using shared keys or certificate-based authentication.

8.8.1 EAP-TLS and TLS

The TOE supports EAP-TLS using TLS version 1.0, TLS 1.1 and TLS 1.2 and supports the following ciphersuites:

- Mandatory Ciphersuites in accordance with RFC 5246:
  - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC5246
- Optional Ciphersuites:
  - TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC5246
  - TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC5246

- TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC5246

EAP-TLS can be configured as one of the EAP types accepted using the AcceptEAPTypes key in the Wi-Fi payload of the configuration profile.

When configuring the TOE to utilize EAP-TLS as part of a Wi-Fi Protected Access 2 (WPA2) protected Wi-Fi-network, the CA certificate(s) to which the server's certificate must chain can be configured using the PayLoadCertificateAnchorUUID key in the Wi-Fi payload of the configuration profile.

Using the TLSAllowTrustExceptions key in the Wi-Fi payload of the configuration profile the administrator can enforce that untrusted certificates are not accepted and the authentication fails if such an untrusted certificate is presented.

The TOE provides mobile applications API service for TLS version 1.2 including support for following ciphersuites from FCS_TLSC_EXT.1:

- Mandatory Ciphersuites:
  - TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- Optional Ciphersuites
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC5289
  - 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

In addition, the following ciphersuites are enabled within iOS 11.2 and can be disabled in the Operational Environment:

1. TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
2. TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
3. TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
4. TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
5. TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
6. TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
7. TLS_RSA_WITH_AES_128_GCM_SHA256
8. TLS_RSA_WITH_AES_256_CBC_SHA
9. TLS_RSA_WITH_AES_128_CBC_SHA
10. TLS_RSA_WITH_AES_256_GCM_SHA384
11. TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
12. TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
13. TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

14. TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Furthermore, the elliptic curve cipher suites above may utilize the following supported elliptic curve extensions by default:

1. secp256r1
2. secp384r1

Additional supported elliptic curve extensions below are also enabled by iOS and may be disabled in the Operational Environment:

1. x25519

When an application uses the provided APIs to attempt to establish a trusted channel, the TOE will compare the DN contained within the peer certificate (specifically the Subject CN, as well as any Subject Alternative Name fields, IP Address, or Wildcard certificate if applicable) to the DN of the requested server. If the DN in the certificate does not match the expected DN for the peer, then the application cannot establish the connection.

Applications can request from the TOE the list of ciphersuites supported and then define which of the supported ciphersuites they enable for the TLS protected session they are going to set up. Once the connection has been set up the application can retrieve the ciphersuite negotiated with the communication partner.

When setting up a TLS session, the library function for the handshake (SSLHandshake) will indicate, via a result code, any error that occurred during the certificate chain validation.

Communication between the Agent and the MDM Server is protected by TLS using the above supported cipher specifications.

Certificate pinning is supported and is described in [HTTPSTN2232]. The user of the TLS framework can use this certificate pinning support. However, existing TLS clients in the TOE, such as the Safari browser, do not support certificate pinning.

8.8.2 AlwaysOn VPN

For managed and supervised devices, the TOE has the option to be configured with an 'AlwaysOn' VPN where the organization has full control over device traffic by tunneling all IP traffic back to the organization using an Internet Key Exchange (IKE) v2 based IPSec tunnel. A specific set of configuration key values dedicated to the VPN type 'AlwaysOn' allows the specification of the interfaces (cellular or Wi-Fi) for which the VPN is 'AlwaysOn' (default is: for both), the specification of exceptions from this service (only VoiceMail and Airprint can be listed as exceptions), and the definition of exceptions for Captive networking (if any).

For IKEv2 the configuration contains (among other items):

- The IP address or hostname of the VPN server
- The identifier of the IKEv2 client in one of the supported formats
- The authentication method (shared key or certificate)
- The server certificate (for certificate-based authentication)
- If extended authentication is enabled
- The encryption algorithm (allows for AES-128, and AES-256. Single DES and 3DES shall not be used in the evaluated configuration)

- The integrity algorithm (allows for SHA1-96 (default), SHA1-128, SHA2-256, SHA2-384, SHA2-512)

8.8.3 Bluetooth

The TOE supports Bluetooth 4.0 with the following Bluetooth profiles.

- Hands-Free Profile (HFP 1.6)
- Phone Book Access Profile (PBAP)
- Advanced Audio Distribution Profile (A2DP)
- Audio/Video Remote Control Profile (AVRCP 1.4)
- Personal Area Network Profile (PAN)
- Human Interface Device Profile (HID)
- Message Access Profile (MAP)

Users can pair their device with a remote Bluetooth device using the 'Set up Bluetooth Device' option from the Bluetooth status menu. They can also remove a device from the device list.

Manual authorization is implicitly configured since pairing can only occur when explicitly authorized through the Settings»Bluetooth interface. During the pairing time, another device (or the iOS) can send a pairing request. Commonly, a six-digit number is displayed on both sides which must be manually matched by a user, i.e. the PIN is shown and the user must accept it before the pairing completes. If one device does not support this automatic exchange of a PIN, a window for entering a manual PIN is shown. That PIN must match on both sides.

iOS requires that remote Bluetooth devices support an encrypted connection. Devices that want to pair with the TOE via Bluetooth are required by Apple to use Secure Simple Pairing, which uses ECDH based authentication and key exchange. See the Specification of the Bluetooth System 4.0, Volume 2, Part H, chapter 7 [BT] for details. No data can be transferred via Bluetooth until pairing has been completed.

The only time the device is Bluetooth discoverable is when the Bluetooth configuration panel is active and in the foreground (there is no toggle switch for discoverable or not discoverable—unless the configuration panel is the active panel, the device is not discoverable).

Connections via Bluetooth/LE are secured using AES-128 in counter with CBC-MAC (CCM) mode. A local database is kept of all Bluetooth device addresses for paired devices which is checked prior to any automatic connection attempt. Additionally, Bluetooth devices may not establish more than one connection. Multiple connection attempts from the same BD_ADDR for an established connection will be discarded. For details of the security of Bluetooth/LE see the Specification of the Bluetooth System 4.0 [BT], Volume 6, Part E, chapter 1.

8.8.4 Wireless LAN

The TOE implements the wireless LAN protocol as defined in IEEE 802.11 (2012). The TOE uses the random number generators of the CoreCrypto cryptographic modules for the generation of keys and other random values used as part of this protocol.

As required by IEEE 802.11 (2012), the TOE implements the CTR with CBC-MAC protocol (CCMP) with AES (128-bit key) as defined in section 11.4.3 of 802.11. This protocol is mandatory for IEEE 802.11 (2012) and is also the default protocol for providing confidentiality and integrity for wireless LANs that comply with IEEE 802.11. The implementation of the AES algorithm is performed by the bulk encryption operation of the WLAN chip.

AES key wrapping as defined in NIST SP 800-38F is used to wrap the Group Temporal Key (GTK), which is sent in an Extensible Authentication Protocol (EAPOL) key frame in message three of the 4-way handshake defined in section 11.6.2 of IEEE 802.11 (2012).

AES key unwrapping is performed as described in NIST SP 800-38F section 6.1, Algorithm 2: W^-1(C), and in section 6.2, Algorithm 4: KW-AD(C).

W^-1

- Initialize the variables
- Let s = 6(n-1)
  - Let C_1, C_2, ..., C_n be the semiblocks such that C = C_1 || C_2 || ... || C_n
  - Let A^s = C_1
  - For i = 2, ..., n: let _i^s = C i
- Calculate the intermediate values
  - A^(t-1) = MSB_(64)(CIPH^(-1)_K ((A^t ⊕ [t] 64) || R_n^t))
  - R_2^(t-1) = LSB_(64)(CIPH^(-1)_K ((A^t ⊕ [t] 64) || R_n^t))
  - For i = 2, ... , n-1, R_(i+1)^(t-1) = R_i^t
- Output the results
  - Let S_1 =A^0
  - For i = 2, ... , n: S_i = R_i^0
  - Return S_1 || S_2 || ... || S_n

Figure 3 of SP 800-38F depicts how the unwrapping is accomplished.

KW-AD(C)

- Let ICV1 = 0xA6A6A6A6A6A6A6A6
- Let S = W^-1(C)
- If MSB_64(S) != ICV1, then return FAIL and stop
- Return P = LSB_(64(n-1))(S)

The TOE implements WPA2 Enterprise, certified by the Wi-Fi Alliance, as shown in the following table.

Device Name	WiFi Alliance certificate
iPhone 5s A1533/A1453/A1457/A1530	WFA 20103
iPhone 6 Plus/ iPhone 6 A1549/A1522 A1586/A1524	WFA 55890 WFA 55892
iPhone 7 Plus/ iPhone 7 A1784/A1778 A1661/A1660	WFA 66687 WFA 66691
iPhone 8 Plus/iPhone 8 A1864/A1898/A1899/A1897 A1863/A1906/A1907/A1905	WFA 72375 WFA 72355
iPhone X (A1865/A1902/A1901)	WFA 72376 WFA 72356
iPhone SE (A1662 / A1723)	WFA 64671 WFA 64670
iPad mini 3 (A1599/A1600/A1601)	WFA 20607
iPad mini 4 (A153/A1550)	WFA 61740 WFA 61719
iPad Air 2 (A1566/A1567)	WFA 56012 WFA 56011
iPad Pro 12.9” (A1584/A1652)	WFA 61740 WFA 61525
iPad Pro 9.7” (A1673/A1674)	WFA 64672 WFA 64673
iPad (A1822/A1823)	WFA 70147 WFA 70146
iPad Pro 12.9” (A1670/A1671)	WFA 70651 WFA 70412
iPad Pro 10.5” (A1701/A1709)	WFA 70652 WFA 70413

Table 16: WiFi Alliance certificates

8.9 Security Audit (FAU)

8.9.1 Audit Records

iOS logging capabilities are able to collect a wide array of information concerning TOE usage and configuration. The available commands and responses constitute audit records and must be configured by TOE administrators using profiles that are further explained in section 8.5.2. These profiles must also be used to determine the audit storage capacity as well as default action when capacity is reached.

Although the specific audit record format is determined via configuration profile, the following attributes form the baseline:

- Date and time the audit record was generated
- Process ID or information about the cause of the event
- Information about the intended operation
- Success or failure (where appropriate)

Audit record information is not available to TOE users or administrators on TOE devices and is only accessible externally on trusted workstations via the Apple Configurator or to an MDM server on enrolled devices.

Depending on the underlying OS of the trusted workstation or MDM server, the audit records are transferred to the following locations.

- macOS
  - ~/Library/Logs/CrashReporter/MobileDevice/[Your_Device_Name]/
- Windows
  - C:\Users\[Your_User_Name]\AppData\Roaming\Apple Computer\Logs\CrashReporter\MobileDevice\[Your_Device_Name]\

8.9.2 MDM Agent Alerts

The MDM agent generates and sends an alert in response to an MDM server request (i.e., applying a policy, receiving a reachability event). The Status key field in Table 9: Wi-Fi CAVS Certificates is used as the alert message to satisfy the FAU_ALT_EXT.2 requirements. The MDM Agent's response is being used as the alert transfer mechanism.

When a Configuration Profile is sent to an MDM Agent, the MDM Agent responds using an "Alert", the *MDM Result Payload*, a plist-encoded dictionary containing the following keys, as well as other keys returned by each command.

Key	Type	Content	
Status	String	Status. Legal values are described as:	
		Status value	Description
		Acknowledged	Everything went well.
		Error	An error has occurred. See the ErrorChain for details.
		CommandFormatError	A protocol error has occurred. The command may be malformed.
		Idle	The device is idle (there is no status).
		NotNow	The device received the command, but cannot perform it at this time. It will poll the server again in the future.
UDID	String	UDID of the device	
CommandUUID	String	UUID of the command that this response is for (if any)	
ErrorChain	Array	*Optional*—Array of dictionaries representing the chain of errors that occurred	

Table 17: MDM Agent Status Commands

During installation:

- The user or administrator tells the device to install an MDM payload. The structure of this payload is described in the Structure of MDM Payloads section of [iOS_MDM].
- The device connects to the check-in server. The device presents its identity certificate for authentication, along with its UDID and push notification topic.

Note: Although UDIDs are used by MDM, the use of UDIDs is deprecated for iOS apps.

If the server accepts the device, the device provides its push notification device token to the server. The server should use this token to send push messages to the device.

This check-in message also contains a PushMagic string. The server must remember this string and include it in any push messages it sends to the device.

During normal operation:

- The server (at some point in the future) sends out a push notification to the device.
- The device polls the server for a command in response to the push notification.
- The device performs the command.
- The MDM Agent contacts the server to report the result of the last command and to request the next command.

From time to time, the device token may change. When a change is detected, the device automatically checks in with the MDM Server to report its new push notification token.

Note: The device polls only in response to a push notification; it does not poll the server immediately after installation. The server must send a push notification to the device to begin a transaction.

The MDM Agent initiates communication with the MDM Server in response to a push notification by establishing a TLS connection to the MDM Server URL. The MDM Agent validates the server's certificate, and then uses the identity specified in its MDM payload as the client authentication certificate for the connection.

When an MDM Server wants to communicate with iPhone or iPad, a silent notification is sent to the MDM Agent via the Apple Push Notification (APN) service, prompting it to check in with the server. The process of notifying the MDM Agent does not send any proprietary information to or from the APNS. The only task performed by the push notification is to wake the device so it checks in with the MDM Server.

8.9.2.1 Queuing of Alerts

In cases where the TLS channel is unavailable, for example because the device is out of range of a suitable network, an alert in regard to the successful installation of policies is queued until the device is able to communicate with the server again. The queue cannot become long, because if the device is out of communication with the MDM server no additional requests can be received. If the MDM Server does not receive the alert, the MDM Server should re-initiate the transfer until a response is received from the device.

There are certain times when the device is not able to do what the MDM Server requests. For example, databases cannot be modified while the device is locked with Data Protection. When a device cannot perform a command due to these types of situations, it will send the NotNow status without performing the command. The server may send another command immediately after receiving this status, but chances are the following command will also be refused.

After sending a NotNow status, the device will poll the server at some future time. The device will continue to poll the server until a successful transaction is completed.

The device does not cache the command that was refused. If the server wants the device to retry the command, it must send the same command again later, when the device polls the server.

The server does not need to send another push notification in response to this status. However, the server may send another push notification to the device to have it poll the server immediately.

The following commands are guaranteed to execute on iOS, and never return NotNow.

- DeviceInformation
- ProfileList
- DeviceLock
- EraseDevice
- ClearPasscode
- CertificateList
- ProvisioningProfileList
- InstalledApplicationList
- Restrictions

8.9.2.2 Alerts on successful application of policies

Candidate policies are generated by the administrator and disseminated as a configuration profile using one of the methods already described in section 8.5.2 above.

The protocol for managing configuration profiles between the MDM Server and the MDM Agent is defined by the MDM Protocol [iOS_MDM].

When the application of policies to a mobile device is successful the MDM Agent replies with an MDM Result Payload with Status value "Acknowledged".

If a policy update is not successfully installed then the MDM Agent replies with an MDM Result Payload with Status value "Error" or CommandFormatError, "Idle" and "NotNow".

8.9.2.3 Alerts on receiving periodic reachability events

Periodic reachability events are initiated by the MDM Server using Push Notifications. When a periodic reachability event is received the MDM Agent contacts the server in the manner described in section 8.9.1, above.

8.10 Mapping to the Security Functional Requirements

The following table provides a mapping of the SFRs defined in chapter 6 of this [ST] to the functions implemented by the TOE, referring to the previous sections of this TSS where the additional information is given.

SFR	TSS Section /Reference	Comment
FAU_ALT_EXT.2 (AGENT)	8.9.2 MDM Agent Alerts 8.8 Trusted Path/Channels (FTP)	Alerts are implemented as described in the instantiation of the MDM protocol, especially MDM Command payloads and MDM Result payloads.
FAU_GEN.1(1) (MDF)	8.9.1 Audit Records Table 3: Combined mandatory auditable events from [PP_MD_V3.1] and [PP_WLAN_CLI_EP_V1.0]	Audit data generation is provided by iOS logging functionality in accordance with events enumerated in the table.
FAU_GEN.1(2) (AGENT)	8.9.1 Audit Records	Format for audit records
FAU_SEL.1(2) (AGENT)	NA	No TSS requirement
FAU_STG.1 (MDF)	8.9.1 Audit Records	Audit data is not accessible on TOE devices. Locations of logs on MDM servers (outside of the TOE) are given.
FAU_STG.4 (MDF)	8.9.1 Audit Records 8.5.2 Configuration Profiles	Audit capacity is determined by configuration profile and can remove the oldest audit record when the audit trail is full.
FCS_CKM.1(1) (MDF)	8.2.3 CAVS Certificates Table 8: User Space CAVS Certificates (Rows 3 and 4)	RSA and elliptic curve cryptography (ECC) key generation are functions of the Apple CoreCrypto Module v8 for ARM. The modules can generate RSA key pairs with modulus sizes of 2048 to 3072, as provided by the PP, in increments of 32 bits. ECC key pairs can be generated for NIST curves P-256, P-384.
FCS_CKM.1(2) (WLAN)	8.2.3 CAVS Certificates Table 10: Wi-Fi CAVS Certificates Table 11: Hardware DRBG CAVS Certificates 8.8.4 Wireless LAN (WiFi Alliance certiicates)	PRF-384 is implemented as defined in IEEE 802.11-2012, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", section 11.6.1.2. It is implemented in the TOE as part of the WPA implementation and is used for the key generation of AES keys when the Counter Mode CBC-MAC Protocol (CCMP) cipher (defined in section 11.4.3.1 of IEEE 802.11-2012) is used. The Random Bit Generator used is the one of the main device. Compliance to the requirements of FCS_RBG_EXT.1 has been analyzed in the proprietary EAR which has been provided to NIAP.
FCS_CKM.2(1) (MDF)	8.2.1 Overview of Key Management 8.2.1.1Password based key derivation 8.2.3 CAVS Certificates Table 8: User Space CAVS Certificates (Rows 3, 4 and 9)	Two key establishment schemes are defined: RSA based and elliptic curve based.

SFR	TSS Section /Reference	Comment
FCS_CKM.2(2) (MDF)	8.2.3 CAVS Certificates RSA – Vendor affirmed EC - Table 8: User Space CAVS Certificates (Row 7 and 9)	No TSS requirement
FCS_CKM.2(3) (WLAN)	8.8.4 Wireless LAN 8.2.3 CAVS Certificates Table 10: Wi-Fi CAVS Certificates Table 11: Hardware DRBG CAVS Certificates 8.8.4 Wireless LAN (WiFi Alliance certificates)	Describes how the GTK is unwrapped
FCS_CKM_EXT.1 (MDF)	8.2.3 CAVS Certificates Table 8: User Space CAVS Certificates (Row 1) Entropy assessment report (EAR) On file with NIAP	The TOE supports a device unique REK (256-bit AES keys) that is generated in the production environment and installed during production. This REK is protected by the hardware such that it can only be used for encryption and decryption operation and only in the Secure Enclave. It cannot be read or modified by any software or firmware in the TOE. The proprietary EAR has analyzed the random bit generator (RBG) used in the production environment for compliance to the requirements defined in FCS_RBG_EXT.1
FCS_CKM_EXT.2 (MDF)	8.1 Hardware Protection Functions 8.1.1 The Secure Enclave 8.2 Cryptographic Support 8.2.1 Overview of Key Management Table 8: User Space CAVS Certificates (Row 1)	All the data encryption keys (DEKs) described in the key hierarchy diagram in Figure 4: Key Hierarchy in iOS are generated using the RBG of the secure enclave and are AES keys with 256-bit key size or Curve25519 keys with 256-bit key size using the functions of the Apple CoreCrypto Modules v8 for ARM. The Key Generation functions are pre-programmed to use only this RBG and key-size, which cannot be changed.

SFR	TSS Section /Reference	Comment
FCS_CKM_EXT.3 (MDF)	8.2.1 Overview of Key Management Figure 4: Key Hierarchy in iOS Table 8: User Space CAVS Certificates (Row 2) (Row 8)	All the KEKs described in the key hierarchy diagram in Figure 4: Key Hierarchy in iOS are generated using the RBG of the secure enclave and are AES keys with 256-bit key size using the functions of the Apple CoreCrypto kernel Module v8 for ARM. The Passcode Key is generated using PBKDF with the UID as the key and a salt that is generated using the RBG in the Secure Enclave. This RBG in the Secure Enclave has been analyzed for compliance with the requirements of FCS_RBG_EXT.1 in the proprietary EAR, which has been provided to NIAP.
FCS_CKM_EXT.4 (MDF)	8.2.1 Overview of Key Management Table 6: Summary of keys and persistent secrets in Table 7: Summary of keys and persistent secrets used by the Agent	Keys in volatile memory are overwritten by zeros when no longer needed. Keys in non-volatile flash memory are cleared by a block erase when no longer needed.
FCS_CKM_EXT.5 (MDF)	8.2.1 Overview of Key Management	A wipe operation is performed after exceeding the limit number of failed authentication attempts or upon receiving a request from an authorized administrator. Wiping is performed by clearing the block with the highest level of KEKs shown in the diagram in Figure 4: Key Hierarchy in iOS using a block erase of the 'effaceable area' of the non-volatile flash memory.
FCS_CKM_EXT.6 (MDF)	8.1 Hardware Protection Functions Table 8: User Space CAVS Certificates (row 1)	The salt used for the password-based key derivation function is generated using the RBG in the Secure Enclave, which (as noted before) has been analyzed for compliance with the requirements of FCS_RBG_EXT.1 in the proprietary EAR, which has been provided to NIAP. Other salts and random values are generated using the RBG of the main device.
FCS_CKM_EXT.7 (MDF)	See FCS_CKM_EXT.1 (MDF)	See FCS_CKM_EXT.1 (MDF)
FCS_COP.1(1) (MDF)	8.2 Cryptographic Support Table 8: User Space CAVS Certificates (row 2) Table 9: Kernel Space CAVS Certificates (row 10)	AES-CBC and AES-GCM are implemented as approved functions in the Apple CoreCrypto Modules v8 for ARM as well as in the Wi-Fi chip. AES-CCM and AES Key Wrap are implemented as non-approved functions in the Apple CoreCrypto Modules v8 for ARM. AES-CCMP is implemented as defined in IEEE 802.11-2012, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", section 11.4.3. See section 8.8.4 for Wi-Fi Alliance certificates. Cryptographic key sizes are 128-bit and 256-bit.

SFR	TSS Section /Reference	Comment
FCS_COP.1(2) (MDF)	Table 8: User Space CAVS Certificates (row 5)	No TSS requirement
FCS_COP.1(3) (MDF)	Table 8: User Space CAVS Certificates (row 3 and 4)	No TSS requirement
FCS_COP.1(4) (MDF)	8.2 Cryptographic Support Table 8: User Space CAVS Certificates (row 6) Table 9: Kernel Space CAVS Certificates (row 12)	HMAC-SHA1, HMAC-SHA-256, HMAC-SHA384, and HMAC-SHA-512 are implemented as approved functions in the Apple iOS CoreCrypto Modules. The key attributes used are defined in the SFR, FCS_COP.1.1(4).
FCS_COP.1(5) (MDF)	section 8.2.1 Overview of Key Management Table 8: User Space CAVS Certificates (row 6 and 8)	PBKDF is implemented as a *vendor affirmed* function in the Apple iOS CoreCrypto Modules. The password is fed as a binary string to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) are described in this [ST]. The output of the hash function is used to form the sub-mask and is conditioned to be the same length as the KEK, 256-bits, that is specified in FCS_CKM_EXT.3.
FCS_HTTPS_EXT.1 (MDF)	NA	No TSS requirement
FCS_IV_EXT.1 (MDF)	8.2.1 Overview of Key Management	All initialization vectors are implemented as defined in table 14 of [PP_MD_V3.1] and [EP_MDM_AGENT_V3.0].

SFR	TSS Section /Reference	Comment
FCS_RBG_EXT.1 (MDF)	Entropy Assessment Report (EAR) On file with NIAP. Table 8: User Space CAVS Certificates (row 1) Table 11: Hardware DRBG CAVS Certificates	There are two Random Bit Generators implemented in the TOE that are used for security functionality specified in this [ST]: one in the Secure Enclave (used to generate the salt used in the PBKDF for generating the Passcode Key), and one in the main device (used for the generation of all other random bits used for security functionality specified in the [ST]). Both RBGs are based on hardware noise sources and both have been analyzed in the proprietary EAR, which has been provided to NIAP, to have a minimum of 256 bits of entropy. The cryptographic library provides an interface for application programs that those programs can use to request random bits from the RBG in the main device. The CAVS Certificates listed in Table 11: Hardware DRBG CAVS Certificates.
FCS_SRV_EXT.1 (MDF)	NA	No TSS requirement
FCS_STG_EXT.1 (MDF)	8.2.1 Overview of Key Management	Cryptographic keys are stored in keychains. Keys may be installed by an administrator as part of a configuration profile of may be installed by an application. Access control to keychain items is described in section 8.3.6, *Keychain Data Protection* in this [ST].
FCS_STG_EXT.2 (MDF)	8.2.1 Overview of Key Management	The key hierarchy implemented by the TOE is explained in section 8.2.1 and Figure 4 of this [ST]. All keys are wrapped using AES in Key Wrap mode with 256-bit keys.
FCS_STG_EXT.3 (MDF)	8.2.1 Overview of Key Management	The integrity of wrapped keys is implemented through the MAC before the unwrapped key is used.
FCS_STG_EXT.4 (AGENT)	8.2.1 Overview of Key Management 8.2.1.1 Password based Key derivation	iOS provides platform-provided key storage for keys which is utilized by the MDM Agent by calling the iOS API.
FCS_TLSC_EXT.1 (MDF)	8.8.1 EAP-TLS and TLS	EAP TLS with TLS 1.0 and TLS 1.1 only. TLS 1.2 is implemented by the TOE including all mandatory and some optional ciphersuites listed in [PP_MD_V3.1]. A client certificate can be installed on the device via a configuration profile. The information includes which types of reference identifiers are supported.
FCS_TLSC_EXT.1/ WLAN (WLAN)	8.4.2 Certificates	A description of the certificates used is provided, including client-side certs for mutual authentication.

SFR	TSS Section /Reference	Comment
FCS_TLSC_EXT.2 (MDF)	8.8.1 EAP-TLS and TLS	The use of Elliptic Curve ciphersuites selected in FCS_TLSC_EXT.1 are implemented by default without configuration.
FDP_ACF_EXT.1 (MDF)	8.3.1 Protection of Files 8.3.2 Application Access to Files	The TOE implements an access control policy to data in order to control sharing.
FDP_DAR_EXT.1 (MDF)	8.3.6 Keychain Data Protection	All data-at-rest is protected using the class key of the class the file belongs to. Similar all keychain items are protected by their class key. Figure 4 explains the KEKs used to protect the class keys.
FDP_DAR_EXT.2 (MDF)	8.3.6 Keychain Data Protection Table 12: Keychain to File-system Mapping	The class key for the NSFileProtectionCompleteUnlessOpen is used for the case of protecting data when the device is locked. The description of the NSFileProtectionCompleteUnlessOpen class describes the use of the asymmetric key pair over Curve25519 for this purpose.
FDP_IFC_EXT.1 (MDF)	8.8.2 AlwaysOn VPN	The TOE can be configured (using a Configuration Profile) such that all IP traffic flows through an IPsec VPN client (AlwaysOn VPN).
FDP_PBA_EXT.1 (MDF)	8.5.3 Biometric Authentication Factors	Describes the activities that happen during biometric authentication. When accessing the settings for BAFs (e.g. enrolling a new fingerprint/face) the password is required to authenticate the change.
FDP_STG_EXT.1 (MDF)	8.4.2 Certificates	The Trust Anchor Database is stored as a protected item.
FDP_UPC_EXT.1 (MDF)	8.8 Trusted Path/Channels (FTP)	The TOE provides a library that applications can use to set up a TLS protected connection, set up a HTTPS protected connection, use Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) and Bluetooth Low Energy (LE) to set up a protected communication channel and initiate communication via this channel.
FIA_AFL_EXT.1 (MDF)	8.5.2 Configuration Profiles 8.4 Identification and Authentication (FIA)	The TSF can be configured (using a Configuration Profile) to limit the number of consecutive failed authentication attempts to any number between 2 and 10. Authentication attempts are further described in section 8.4 Identification and Authentication (FIA)
FIA_BLT_EXT.1 (MDF)	8.8.3 Bluetooth	The TSF can be configured to require explicit user authorization before pairing with a remote Bluetooth device.
FIA_BLT_EXT.2 (MDF)	NA	No TSS requirement specified
FIA_BLT_EXT.3 (MDF)	NA	No TSS requirement specified
FIA_BLT_EXT.4 (MDF)	NA	No TSS requirement specified

SFR	TSS Section /Reference	Comment
FIA_BMG_EXT.1 (MDF)	8.4.1 Biometric Authentication	Describes the testing and calculations completed to determine the FAR and FRR
FIA_BMG_EXT.2 (MDF)	8.4.1.4 Biometric Sample Quality	Describes the biometric sample quality used on the TOE for deriving the authentication template enrolled.
FIA_BMG_EXT.3 (MDF)	8.4.1.4 Biometric Sample Quality	Describes the biometric sample quality used on the TOE for deriving the authentication template enrolled.
FIA_BMG_EXT.5 (MDF)	8.4.1.4 Biometric Sample Quality	How the matching algorithm addresses properly formatted templates with unusual data properties, incorrect syntax, or low quality is addressed.
FIA_ENR_EXT.2 (AGENT)	8.4.3 MDM Server Reference ID Table 13: MDM Server Reference Identifiers	The MDM Agent records the reference ID of the MDM Server in flash memory. The table specifies the reference identifies used by the MDM server
FIA_PAE_EXT.1 (WLAN)	NA	No TSS requirement
FIA_PMG_EXT.1 (MDF)	NA	No TSS requirement
FIA_TRT_EXT.1 (MDF)	8.4 Identification and Authentication (FIA)	The TSF enforces a delay of at least 5 seconds between user authentication attempts.
FIA_UAU.5 (MDF)	8.4 Identification and Authentication (FIA)	The mechanisms and rules for each authentication type (Password, Touch ID and Face ID are described.)
FIA_UAU.6(1) (MDF)	NA	No TSS requirement
FIA_UAU.6(2) (MDF)	NA	No TSS requirement
FIA_UAU.7 (MDF)	8.4 Identification and Authentication (FIA)	When a password is entered, entries are obscured by dot symbol for each character as the user input occurs. Biometric Authentication factors are inherently obscured as no feedback is presented to the user during input.
FIA_UAU_EXT.1 (MDF)	8.2.1 Overview of Key Management	The passcode is used, combined with the UID and the salt, to derive the passcode key (KEK) which is then used to decrypt the class keys for protected data. That passcode key is erased whenever the device is locked, and reconstructed whenever the device is unlocked with the right passcode.

SFR	TSS Section /Reference	Comment
FIA_UAU_EXT.2 (MDF)	8.4 Identification and Authentication (FIA)	The TSF can be configured to allow no actions to be performed before the user is authenticated. Except for making emergency calls, users need to authenticate to the device to perform any other services.
FIA_X509_EXT.1 (MDF)	8.4.2 Certificates	X.509 certificates are validated using the certificate path validation algorithm defined in RFC 5280. Certificate paths must terminate in the Trust Anchor Database for a certificate to be regarded as trusted. The extendedKeyUsage field in the certificate is validated to ensure that the key is used in accordance with its usage specification. Certificate Authority (CA) certificates are only accepted if the basicConstraints extension exists and the CA flag in this extension is set. The Online Certificate Status Protocol (OCSP) is supported to determine if a certificate has been revoked.
FIA_X509_EXT.2 (MDF)	8.4.2 Certificates	X.509v3 certificates are supported for authentication for EAP-TLS, IPsec, TLS, HTTPS, and code signing for software updates, code signing for mobile applications.
FIA_X509_EXT.2 (WLAN)	8.8.4 Wireless LAN	The library provided for certificate handling and management provides a service for certificate validation. The return value indicates the success or failure of the validation with detailed failure codes indicating why the validation failed.
FIA_X509_EXT.3 (MDF)	NA	No TSS requirements
FMT_MOF_EXT.1 (MDF)	Table 5: Management Functions	Table 5: Management Functions in this [ST] indicates the management functions and the entity (user or administrator) that can perform the function.
FMT_POL_EXT.2 (AGENT)	8.5.2 Configuration Profiles	MDM policies can be validated against enterprise signatures.
FMT_SMF_EXT.1 (MDF)	8.5 Specification of Management Functions (FMT) Table 5: Management Functions	Table 5: Management Functions indicates the management functions that can be performed by either the administrator, the user, or both. In the case of the administrator functions most of them (except for remote wipe and remote locking) are performed by installing Configuration Profiles on the TOE. Administration functions for the user are available through the user interface.

SFR	TSS Section /Reference	Comment
	8.4 Identification and Authentication (FIA)	Function 1:
	NA	Function 2:
	NA	Function 3:
	8.5.5 Radios Table 1: Devices Covered by the Evaluation Table 2: Cellular Protocols Supported Table 5: Management Functions	Function 4:
	8.5.2 Configuration Profiles 8.5.6 Collection devices Table 5: Management Functions	Function 5:
	NA	Function 6:
	NA	Function 7:
	8.5.2 Configuration Profiles	Function 8:
	8.2.1 Overview of Key Management Table 6: Summary of keys and persistent secrets in	Function 9:
	8.2.1 Overview of Key Management Table 6: Summary of keys and persistent secrets in	Function 10:
	NA	Function 11:
	8.4.2 Certificates	Function 12:
	8.5.2 Configuration Profiles	Function 13:
	8.5.2 Configuration Profiles	Function 14:
	NA	Function 15:
	NA	Function 16:
	NA	Function 17:
	8.8.3 Bluetooth	Function 18:
	NA	Function 19:
	NA	Function 22:
	8.5.3 Biometric Authentication Factors	Function 23:
	NA	Function 28:
	NA	Function 30:
	NA	Function 33:
	8.7.3 Lock Screen Banner Display	Function 36:
	NA	Function 37:

SFR	TSS Section /Reference	Comment
FMT_SMF_EXT.1/ WLAN (WLAN)	NA	No specific TSS requirements
FMT_SMF_EXT.2 (MDF)	8.2.1 Overview of Key Management	Upon unenrollment and when a remote wipe command is issued protected data is wiped by erasing the top level KEKs. Since all data-at-rest is encrypted with one of those keys, the device is wiped.
FMT_SMF_EXT.3 (AGENT)	8.5.1 Enrollment	The MDM Agent is integrated with the platform and provides an interface to ensure that administrator-provided management functions and certificate import are performed.
FMT_UNR_EXT.1 (AGENT)	8.5.4 Unenrollment	Unenrollment is handled as described.
FPT_AEX_EXT.1 (MDF)	8.6.5 Domain Isolation	Address space layout randomization is used for every sandbox used to execute applications in. There are 8 bits of randomness taken from the application processor TRNG involved in the randomization, the seed for the RNG comes from the seed that also feeds the approved DRBG for cryptographic use
FPT_AEX_EXT.2 (MDF)	8.6.5 Domain Isolation	Hardware enforced memory protection is used for every memory page.
FPT_AEX_EXT.3 (MDF)	8.6.5 Domain Isolation	Stack-based buffer overflow protection is implemented for every sandbox.
FPT_AEX_EXT.4 (MDF)	8.6.5 Domain Isolation	The TSF protects itself as well as applications from modification by untrusted subjects. Applications execute in their own address space separated from the address space of other applications and separated from the address space used by the TSF. The TOE does not support Unstructured Supplementary Service Data (USSD) or Man-Machine Interface (MMI) codes and also does not support auxiliary boot modes.
FPT_JTA_EXT.1 (MDF)	8.6.2 Joint Test Action Group (JTAG) Disablement	JTAG per se is not implemented. JTAG-like ports and authentication to them are described in the TSS.
FPT_KST_EXT.1 (MDF)	8 TOE Summary Specification (TSS) 8.2.1 Overview of Key Management 8.1.1 The Secure Enclave	Keys are stored in keychains and are therefore encrypted by one of the KEKs for keychain objects.
FPT_KST_EXT.2 (MDF)	8 TOE Summary Specification (TSS) 8.2.1 Overview of Key Management 8.1.1 The Secure Enclave	The TOE security boundary is described in section 0. Activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data are described Key material is never transmitted in plaintext outside the TOE by the TSF.

SFR	TSS Section /Reference	Comment
FPT_KST_EXT.3 (MDF)	8.1.1 The Secure Enclave	Plaintext keys cannot be exported by TOE users. There is no function that allows this.
FPT_NOT_EXT.1 (MDF)	8.6.9 Self-Tests	In the case of a self-test failure or software integrity verification failure the device will not start. Self-tests for cryptographic functions are performed by the cryptographic modules.
FPT_STM.1 (MDF)	8.6.7 Time	The TSF maintains a time stamp for its own use.
FPT_TST_EXT.1 (MDF)	8.6.9 Self-Tests	The TSF runs a suite of self-tests during initial start-up, as required by FIPS 140-2, for all cryptographic functionality.
FPT_TST_EXT.1/ WLAN (WLAN)	8.6.9 Self-Tests	The TSF runs a suite of self-tests during initial start-up, as required by FIPS 140-2, for all cryptographic functionality.
FPT_TST_EXT.2 (MDF)	8.6.1 Secure Boot	The TSF verifies the integrity of the boot chain.
FPT_TST_EXT.3 (MDF)	N/A	No specific TSS requirements
FPT_TUD_EXT.1 (MDF)	N/A	No specific TSS requirements
FPT_TUD_EXT.2 (MDF)	8.6.3 Secure Software Update	All software updates and all applications need to be digitally signed. This signature is verified before installing any update or application.
FPT_TUD_EXT.3 (MDF)	N/A	No specific TSS requirements
FPT_TUD_EXT.4 (MDF)	8.4.2 Certificates	All apps must have a valid signature that can be verified by a code signing certificate before they are installed on a device.
FTA_SSL_EXT.1 (MDF)	8.6.6 Device Locking 8.5.2 Configuration Profiles 8.2.1 Overview of Key Management	A Configuration Profile can be used to define the maximum time of inactivity before the TSF initiates a transition to the locked state. In addition, both the user and the administrator can initiate a transition to the locked state. During such a transition the display is overwritten and the decrypted class key for the NSFileProtection Complete class is zeroized.
FTA_TAB.1 (MDF)	8.7.3 Lock Screen Banner Display	An advisory warning message regarding unauthorized use of the TOE can be defined using an image that is presented during the lock screen. Configuration for this is described in FMT_SMF_EXT.1.1 Function 36.
FTA_WSE_EXT.1 (WLAN)	8.5.2 Configuration Profiles 8.7.3 Lock Screen Banner Display	Management of acceptable WLAN attributes can be configured via profile
FTP_ITC_EXT.1(2) (AGENT)	8.4.2 Certificates 8.4.3 MDM Server Reference ID	

SFR	TSS Section /Reference	Comment
FTP_ITC_EXT.1(3) (WLAN)	8.8.4 Wireless LAN 8.8.1 EAP-TLS and TLS	These sections specify which protocols are implemented according to the IEEE 802.11 (2012) standard for WLAN and which algorithms are supported by the TOE.

Table 18: Mapping of SFRs

Abbreviations and Acronyms

A2DP	Advanced Audio Distribution Profile
ACL	Access Control List
AES	Advanced Encryption Standard
API	Application Programmer Interface
APN	Apple Push Notification
APNS	Apple Push Notification Service
ARM	Advanced RISC Machine
ASLR	Address Space Layout Randomization
AVRCP	Audio/Video Remote Control Profile
BR/EDR	Basic Rate/Enhanced Data Rate
CAVS	Cryptographic Algorithm Validation System
CBC	Cypher Block Chaining
CC	Common Criteria
CCM	Counter with CBC-MAC
CCMP	Counter Mode CBC-MAC Protocol
CDMA	Code Division Multiple Access
CN	Common Name
CTR	Counter
CVE	Common Vulnerabilities and Exposures
DC-HSDPA	Dual-Carrier High Speed Packet Access
DEK	Data Encryption Key
DEP	Device Enrollment Program
DES	Data Encryption Standard
DFU	Device Firmware Upgrade
DH	Diffie-Hellman
DN	Distinguished Name
DNS	Domain Name Server
DRBG	Deterministic Random Bit Generator
DRNG	Deterministic Random Number Generator
EAL	Evaluation Assurance Level
EAPOL	Extensible Authentication Protocol Over LAN
EAP-TLS	Extensible Authentication Protocol Transport Layer Security
EAR	Entropy Assessment Report
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signal Algorithm
ECID	Electronic Chip Identification
EP	Extended Package (for a Protection Profile)
EV-DO	Evolution-Data Optimized
FDD-LTE	Frequency-Division Duplex-Long Term Evolution
FIA	Identification and Authentication
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GID	Group Key
GPS	Global Positioning Satellites
GSM	Global System for Mobile Communication
GTK	Group Temporal Key
HFP	Hands-Free Profile
HID	Human Interface Device Profile
HMAC	Keyed-hash Message Authentication Code
HSPA+	High Speed Packet Access Plus
IKE	Internet Key Exchange
IPsec	Internet Protocol Security
IV	Initialization Vector
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group

KAT	Known Answer Test
KDF	Key Derivation Function
KEK	Key Encryption Key
KW	Key Wrap
LE	Low Energy
LLB	Low-Level Bootloader
LTE	Long Term Evolution
MAC	Message Authentication Code
MAP	Message Access Profile
MD	Mobile Device
MDF	Mobile Device Fundamentals
MDFPP	Mobile Device Fundamentals Protection Profile
MDM	Mobile Device Management
MMI	Man-Machine Interface
MMU	Memory Management Unit
NITZ	Network, Identity and Time Zone
NTP	Network Time Protocol
OCSP	Online Certificate Status Protocol
OSP	Organizational Security Policy
PAE	Port Access Entity
PAN	Personal Area Network Profile
PBAP	Phone Book Access Profile
PBKDF	Password-Based Key Derivation Function
PHY	Physical Layer
PKCS	Public Key Cryptography Standards
PRF	Pseudorandom Function
RBG	Random Bit Generator
REK	Root Encryption Key
RFC	Request for Comment
RISC	Reduced Instruction Set Computing
RSA	Rivest-Shamir-Adleman
S/MIME	Secure/Multipurpose Internet Mail Extensions
SAR	Security Assurance Requirement
SCEP	Simple Certificate Enrollment Protocol
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SIG	Signature
SP	Special Publication
SP	Security Policy
SSID	Service Set Identifier
SSL	Secure Sockets Layer
ST	Security Target
TD-LTE	Time Division Long-Term Evolution
TD-SCDMA	Time Division Synchronous Code Division Multiple Access
TLS	Transport Layer Security
TOE	Target of Evaluation
TRNG	True Random Number Generators
TSF	TOE Security Functionality
TSS	TOE Summary Specification
UI	User Interface
UMTS	Universal Mobile Telecommunications System
USSD	Unstructured Supplementary Service Data
UUID	Universally Unique ID
VPN	Virtual Private Network
VPP	Volume Purchase Program
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access 2
XN	Execute Never