
# Apple iOS 15: iPhones Assurance Activity Report

| | |
|---|---|
| **Version:** | 1.1 |
| **Date:** | 2022-11-03 |
| **Status:** | RELEASED |
| **Classification:** | Public |
| **Filename:** | VID11237_SER_AAR_Apple_iOS_15_v1.1 |
| **Product:** | Apple iOS 15: iPhones |
| **Sponsor:** | Apple Inc. |
| **Evaluation Facility:** | atsec information security corporation |
| **Validation ID:** | 11237 |
| **Validation Body:** | NIAP CCEVS |
| **Author(s):** | Trang Huynh |
| **Quality Assurance:** | King Ables |



atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759

Phone: +1 512-615-7300
www.atsec.com

# Classification Note

**Public Information (public)**

This classification level is for information that may be made available to the general public. No specific security procedures are required to protect the confidentiality of this information. Information classified “public” may be freely distributed to anyone inside or outside of atsec.

Information with this classification shall be clearly marked “public”, except that it is not required to mark “public” on printed marketing material obviously intended for publication.

# Revision History

| Version | Date | Author(s) | Changes to Previous Revision | Application Notes |
|---|---|---|---|---|
| 1.0 | 2022-10-13 | Trang Huynh | First version | |
| 1.1 | 2022-11-03 | Trang Huynh | Address validator comments. | |

# Table of Contents

# List of Tables



# 1 Evaluation Basis and Documents

This evaluation is based on the "Common Criteria for Information Technology Security Evaluation" Version 3.1 Revision 5 [CC] , the "Common Methodology for Information Technology Security Evaluation" [CEM] and the additional assurance activities defined in the following:

- [CFG_MDF-MDMA-VPNC-BT_V1.0]: PP-Configuration for Mobile Device Fundamentals (MDF), Mobile Device Management (MDM) Agents, and Virtual Private Network (VPN) Clients, Version 1.0, dated 2020-02-28, which consists of the following components:
  - [PP_MDF_V3.2]: Protection Profile for Mobile Device Fundamentals, Version 3.2, dated 2021-04-15
  - [MOD_MDM_AGENT_V1.0]: PP-Module for MDM Agents, Version 1.0, dated 2019-04-25
  - [MOD_BT_V1.0]: PP-Module for Bluetooth, Version 1.0, dated 2021-04-15
  - [MOD_VPNC_V2.3]: PP-Module for Virtual Private Network (VPN) Clients, Version 2.3, dated 2021-08-10
- [PP_WLAN_CLI_EP_V1.0]: General Purpose Operating Systems Protection Profile/ Mobile Device Fundamentals Protection Profile Extended Package (EP) Wireless Local Area Network (WLAN) Clients, dated 2016-02-08
- [PKG_TLS_V1.1]: Functional Package for Transport Layer Security (TLS), Version 1.1, dated 2019-02-12

This evaluation claims Exact Compliance with the above PP-Configuration, PP, PP-Modules, and Extended Packages.

The following scheme documents and interpretations have been considered:

- [CCEVS-TD0194]: "Update to Audit of FTP_ITC_EXT.1/WLAN", version as of 2017-04-11.
- [CCEVS-TD0439]: "EAP-TLS Revocation Checking", version as of 2019-08-29.
- [CCEVS-TD0470]: "Wireless Network Restrictions", version as of 2020-01-22.
- [CCEVS-TD0491]: "Update to FMT_SMF_EXT.4 Test 2", version as of 2020-01-15.
- [CCEVS-TD0492]: "TLS-EAP Ciphers and TLS versions for WLAN Client", version as of 2020-01-22.
- [CCEVS-TD0497]: "SFR Rationale, Consistency of SPD, and Implicitly Satisfied SFRs", version as of 2020-10-26.
- [CCEVS-TD0499]: "Testing with pinned certificates", version as of 2020-02-04.
- [CCEVS-TD0513]: "CA Certificate loading", version as of 2020-05-26.
- [CCEVS-TD0517]: "WLAN Client Corrections for X509 and TLSC", version as of 2020-06-19.
- [CCEVS-TD0596]: "VPN Traffic Permitted in FDP_IFC_EXT.1", version as of 2021-07-30.

- [CCEVS-TD0600]: "Conformance claim sections updated to allow for MOD_VPNC_V2.3", version as of 2021-08-10.
- [CCEVS-TD0622]: "VPNC MOD FTP_DIT_EXT.1 corrections", version as of 2022-02-07.
- [CCEVS-TD0623]: "TD0623: FIA_X509_EXT.2.1 Protocol Selection", version as of 2022-02-11.
- [CCEVS-TD0640]: "TD0640: Handling BT devices that do not support encryption ", version as of 2022-06-15.
- [CCEVS-TD0646]: "TD0623: FIA_X509_EXT.2.1 Protocol Selection", version as of 2022-02-11.
- [CCEVS-TD0653]: "TD0653: MDF v3.2 ASE References ", version as of 2022-06-21.
- [CCEVS-TD0660]: " TD0660: Mislabeled SFRs in MDM Agent Auditable Events Table", version as of 2022-07-11.
- [CCEVS-TD0663]: "TD0663: Audit Listing for MDF Moved to Guidance ", version as of 2022-08-26.
- [CCEVS-TD0671]: "TD0671: Bluetooth PP-Module updated to allow for new PP and PP-Module Versions", version as of 2022-09-27.
- [CCEVS-TD0673]: "TD0673: MDM-Agent PP-Module updated to allow for new PP and PP-Module Versions", version as of 2022-09-27.

# 2 Evaluation Results

The evaluator work units have been performed, including: evaluator actions and analysis explicitly stated in the CEM; evaluator actions implicitly derived from developer action elements described in the CC Part 3; and evaluator confirmation that requirements for content and presentation of evidence elements described in the CC Part 3 have been met.

The evaluation was performed by informal analysis of the evidence provided by the sponsor.

## 2.1 CAVP Summary

The following table summarizes the Cryptographic Algorithm Validation Program (CAVP) certificates that apply to the TOE, including specific standards, options, and implementations for each algorithm of each SFR, and the applicable CAVP certificate for each.

The CAVP operational environments (OEs) for the software and firmware modules are the following.

- iOS 15 with Apple processors:
  - A9, A10 Fusion, A11 Bionic, A12 Bionic, A13 Bionic, A14 Bionic, A15 Bionic

Below are the module (Mod) names used in Table 1.

**BC**
Broadcom Core

**KRN**
Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1] (Kernel Space)

**SEP**
SEP Hardware v2.0

**SKS**
Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2]

**USR**
Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1] (User Space)

Note that the Apple corecrypto Module is v12.0, but the SEP Hardware is v2.0.

**Table 1: SFRs to CAVP certificates for iPhones**

| SFR | Algorithm | Modes / Other | Mod | Implementation | CAVP |
|---|---|---|---|---|---|
| FCS_CKM.1 | ECDSA KeyGen and KeyVer<br><br>[FIPS 186-4] | P-256, P-384 | USR | vng_ltc | A2788 |
| | | | KRN | vng_ltc | A2797 |
| | | | SKS | vng_ltc | A2848 |
| | Safe Primes KeyGen<br><br>[SP800-56Ar3] | MODP-2048, MODP-3072 | USR | c_ltc | A2786 |
| FCS_CKM.1/VPN | ECDSA KeyGen and KeyVer<br><br>[FIPS 186-4] | P-256, P-384 | USR | vng_ltc | A2788 |

| SFR | Algorithm | Modes / Other | Mod | Implementation | CAVP |
|---|---|---|---|---|---|
| FCS_CKM.2/UNLOCKED | ECC Key Establishment (KAS-ECC-SSC Sp800-56Ar3)<br><br>[SP800-56Ar3] | P-256, P-384 | USR | c_ltc | A2786 |
| | | | SKS | c_ltc | A2845 |
| | FFC Key Establishment (KAS-FFC-SSC Sp800-56Ar3)<br><br>[SP800-56Ar3] | MODP-2048, MODP-3072 | USR | c_ltc | A2786 |
| | RSA Key Establishment<br><br>[SP800-56B] | 2048, 3072, 4096 | USR | c_ltc | Vender Affirmation as per PL#5 Add2 v2.0 |
| FCS_COP.1/ENCRYPT | AES<br><br>[FIPS 197] | CBC<br><br>128-bit, 256-bit encrypt, decrypt<br><br>[SP800-38A] (CBC) | USR | asm_arm | A2783 |
| | | | KRN | asm_arm | A2793 |
| | | | SKS | asm_arm | A2842 |
| | | CBC<br><br>128-bit, 256-bit encrypt only<br><br>[SP800-38A] (CBC) | SEP | A9 (skg) | C314 |
| | | | | A10 Fusion (skg) | C317 |
| | | | | A11 Bionic (skg) | C319 |
| | | | | A12 Bionic (skg) | C320 |
| | | | | A13 Bionic (skg) | A510 |
| | | | | A14 Bionic (skg) | A1469 |
| | | | | A15 Bionic (skg) | A2863 |
| | | CCM<br><br>128-bit<br><br>[SP800-38C] (CCM) | USR | vng_asm | A2787 |
| | | | KRN | vng_asm | A2796 |
| | | | SKS | vng_asm | A2847 |
| | | | BC | 43452, 4350 | AES 4035 |
| | | | | 4355 | AES 3678 |
| | | | | 4357 | AES 4152 |
| | | | | 4377 | AES 4791 |
| | | | | 4378 Aux | AES 5926 |
| | | | | 4378 Main | AES 5927 |

| SFR | Algorithm | Modes / Other | Mod | Implementation | CAVP |
|---|---|---|---|---|---|
| | | | | 4387 Aux | AES 5926 |
| | | | | 4387 Main | AES 5927 |
| | | CCM<br><br>256-bit<br><br>[SP800-38C] (CCM) | USR | vng_asm | A2787 |
| | | | KRN | vng_asm | A2796 |
| | | | SKS | vng_asm | A2847 |
| | | | BC | 4378 Aux | AES 5952 |
| | | | | 4378 Main | AES 5953 |
| | | | | 4387 Aux | AES 5952 |
| | | | | 4387 Main | AES 5953 |
| | | ECB<br><br>128-bit, 256-bit encrypt, decrypt<br><br>[SP800-38A] (ECB)<br><br>(Used by AES-KW) | SEP | A9 (skg) | C314 |
| | | | | A10 Fusion (skg) | C317 |
| | | | | A11 Bionic (skg) | C319 |
| | | | | A12 Bionic (skg) | C320 |
| | | | | A13 Bionic (skg) | A510 |
| | | | | A14 Bionic (skg) | A1469 |
| | | | | A15 Bionic (skg) | A2863 |
| | | ECB<br><br>256-bit encrypt only<br><br>[SP800-38A] (ECB)<br><br>(Prereq for the hardware CTR_DRBG(AES)) | SEP | A9 | AES 5275 |
| | | | | A10 Fusion | AES 5273 |
| | | | | A11 Bionic | AES 5261 |
| | | | | A12 Bionic | C323 |
| | | | | A13 Bionic (trng) | A501 |
| | | | | A14 Bionic (trng) | A1362 |
| | | | | A15 Bionic (trng) | A2864 |
| | | GCM<br><br>128-bit encrypt, decrypt<br><br>[SP800-38D] (GCM) | USR | vng_asm | A2787 |
| | | | KRN | vng_asm | A2796 |
| | | | SKS | vng_asm | A2847 |
| | | | BC | 4357 | AES 4152 |
| | | | | 4377 | AES 4791 |
| | | | | 4378 Aux | AES 5926 |
| | | | | 4378 Main | AES 5927 |

| SFR | Algorithm | Modes / Other | Mod | Implementation | CAVP |
|---|---|---|---|---|---|
| | | | | 4387 Aux | AES 5926 |
| | | | | 4387 Main | AES 5927 |
| | | GCM<br><br>256-bit<br>encrypt, decrypt<br><br>[SP800-38D] (GCM) | USR | vng_asm | A2787 |
| | | | KRN | vng_asm | A2796 |
| | | | SKS | vng_asm | A2847 |
| | | | BC | 4378 Aux | AES 5926 |
| | | | | 4378 Main | AES 5927 |
| | | | | 4387 Aux | AES 5926 |
| | | | | 4387 Main | AES 5927 |
| | | KW<br><br>128-bit, 256-bit<br>encrypt, decrypt<br><br>[SP800-38F] (KW) | USR | c_asm | A2784 |
| | | | KRN | c_asm | A2794 |
| | | | SKS | c_asm | A2843 |
| | | XTS<br><br>128-bit, 256-bit<br>encrypt, decrypt<br><br>[SP800-38E] (XTS) | USR | asm_arm | A2783 |
| | | | KRN | asm_arm | A2793 |
| | | | SKS | asm_arm | A2842 |
| FCS_COP.1/HASH | SHS<br>Byte-oriented mode<br><br>[FIPS 180-4] | SHA2-256 | USR | vng_neon | A2789 |
| | | | KRN | vng_neon | A2798 |
| | | | SKS | vng_neon | A2849 |
| | | SHA-1, SHA2-384, SHA2-512 | USR | vng_ltc | A2788 |
| | | | KRN | vng_ltc | A2797 |
| | | | SKS | vng_ltc | A2848 |
| FCS_COP.1/SIGN | RSA SigGen<br>PKCS 1.5 and PKCSPSS<br><br>[FIPS 186-4] | Modulo:<br>2048, 3072, 4096<br><br>using<br>SHA2-256, SHA2-384, SHA2-512 | USR | vng_ltc | A2788 |
| | | | KRN | vng_ltc | A2797 |
| | RSA SigVer<br>PKCS 1.5 and PKCSPSS<br><br>[FIPS 186-4] | Modulo:<br>2048, 3072, 4096<br><br>using<br>SHA-1, SHA2-256, SHA2-384, SHA2-512 | USR | vng_ltc | A2788 |
| | | | KRN | vng_ltc | A2797 |

| SFR | Algorithm | Modes / Other | Mod | Implementation | CAVP |
|---|---|---|---|---|---|
| | ECDSA SigGen<br><br>[FIPS 186-4] | P-256, P-384, P-521<br><br>using<br>SHA2-256, SHA2-384, SHA2-512 | USR | vng_ltc | A2788 |
| | | | KRN | vng_ltc | A2797 |
| | | | SKS | vng_ltc | A2848 |
| | ECDSA SigVer<br><br>[FIPS 186-4] | P-256, P-384, P-521<br><br>using<br>SHA-1, SHA2-256, SHA2-384, SHA2-512 | USR | vng_ltc | A2788 |
| | | | KRN | vng_ltc | A2797 |
| | | | SKS | vng_ltc | A2848 |
| FCS_COP.1/KEYHMAC | HMAC<br>Byte-oriented mode<br><br>[FIPS 198-1] | HMAC-SHA2-256 | USR | vng_neon | A2789 |
| | | | KRN | vng_neon | A2798 |
| | | | SKS | vng_neon | A2849 |
| | | HMAC-SHA-1, HMAC-SHA2-384, HMAC-SHA2-512 | USR | vng_ltc | A2788 |
| | | | KRN | vng_ltc | A2797 |
| | | | SKS | vng_ltc | A2848 |
| FCS_RBG_EXT.1 (Kernel and User space) | CTR_DRBG(AES)<br><br>[SP800-90A] | AES-256 | USR | vng_asm | A2787 |
| | | | KRN | vng_asm | A2796 |
| FCS_RBG_EXT.1 (SEP) | CTR_DRBG(AES)<br><br>[SP800-90A] | AES-256 | SEP | A9 | DRBG 2025 |
| | | | | A10 Fusion | DRBG 2023 |
| | | | | A11 Bionic | DRBG 2014 |
| | | | | A12 Bionic | C323 |
| | | | | A13 Bionic (trng) | A501 |
| | | | | A14 Bionic (trng) | A1362 |
| | | | | A15 Bionic (trng) | A2864 |

Table 2 maps iPhones to Broadcom cores and Broadcom CAVP certificates.

**Table 2: Broadcom core supported algorithms and CAVP certificates for iPhones**

| iPhone | Broadcom Core # (Implementation) | Supported Algorithms | CAVP |
|---|---|---|---|
| iPhone 6s (A9)<br>iPhone 6s Plus (A9) | 4350<br>(aes_core.vhd rev 9) | AES-CCM-128 | AES 4035 |
| iPhone SE (A9) | 43452<br>(aes_core.vhd rev 9) | AES-CCM-128 | AES 4035 |
| iPhone 7 (A10 Fusion)<br>iPhone 7 Plus (A10 Fusion) | 4355<br>(aes_core.vhd rev 10) | AES-CCM-128 | AES 3678 |

| iPhone | Broadcom Core # (Implementation) | Supported Algorithms | CAVP |
|---|---|---|---|
| iPhone 8 (A11 Bionic)<br>iPhone 8 Plus (A11 Bionic)<br>iPhone X (A11 Bionic) | 4357<br>(aes_core_gcm.vhd rev 2) | AES-CCM-128<br>AES-GCM-128 | AES 4152 |
| iPhone Xs (A12 Bionic)<br>iPhone Xs Max (A12 Bionic)<br>iPhone $X_R$ (A12 Bionic) | 4377<br>(aes_core_gcm.vhd rev 4) | AES-CCM-128<br>AES-GCM-128 | AES 4791 |
| iPhone 11 (A13 Bionic)<br>iPhone 11 Pro (A13 Bionic)<br>iPhone 11 Pro Max (A13 Bionic)<br>iPhone SE (2nd gen) (A13 Bionic) | 4378 Aux (2.4 GHz)<br>(aes_core_gcm.vhd rev 6) | AES-CCM-128<br>AES-GCM-128<br>AES-GCM-256 | AES 5926 |
| | | AES-CCM-256 | AES 5952 |
| | 4378 Main (5 GHz)<br>(aes_core_gcm_simult_enc_mic.vhd rev 6) | AES-CCM-128<br>AES-GCM-128<br>AES-GCM-256 | AES 5927 |
| | | AES-CCM-256 | AES 5953 |
| iPhone 12 mini (A14 Bionic)<br>iPhone 12 (A14 Bionic)<br>iPhone 12 Pro (A14 Bionic)<br>iPhone 12 Pro Max (A14 Bionic)<br>iPhone 13 mini (A15 Bionic)<br>iPhone 13 (A15 Bionic)<br>iPhone 13 Pro (A15 Bionic)<br>iPhone 13 Pro Max (A15 Bionic) | 4387 Aux (2.4 GHz)<br>(aes_core_gcm.vhd rev 6) | AES-CCM-128<br>AES-GCM-128<br>AES-GCM-256 | AES 5926 |
| | | AES-CCM-256 | AES 5952 |
| | 4387 Main (5 GHz)<br>(aes_core_gcm_simult_enc_mic.vhd rev 6) | AES-CCM-128<br>AES-GCM-128<br>AES-GCM-256 | AES 5927 |
| | | AES-CCM-256 | AES 5953 |

# 2.2 Security Functional Requirements

## 2.2.1 Security audit (FAU)

### 2.2.1.1 Agent Alerts (FAU_ALT_EXT.2)

#### TSS Assurance Activities

##### Assurance Activity AA-FAU_ALT_EXT.2-MDMA-ASE-01

*The evaluator shall examine the TSS and verify that it describes how the alerts are implemented.*

*The evaluator shall examine the TSS and verify that it describes how the candidate policy updates are obtained and the actions that take place for successful (policy update installed) and unsuccessful (policy update not installed) cases. The software components that are performing the processing must also be identified in the TSS and verified by the evaluator.*

*The evaluator also ensures that the TSS describes how reachability events are implemented, and if configurable are selected in FMT_SMF_EXT.4.2. The evaluator verifies that this description clearly indicates who (MDM Agent or MDM Server) initiates reachability events.*

*The evaluator shall ensure that the TSS describes under what circumstances, if any, the alert may not be generated (e.g., the device is powered off or disconnected from the trusted channel), how alerts are queued, and the maximum amount of storage for queued messages.*

**Summary**

Section 8.10.2 *MDM Agent Alerts* describes the agent alerts generated and received by the TOE. This section states the following:

- The MDM agent generates and sends alerts in response to an MDM server such as a request for applying a policy or receiving a reachability event.
- When candidate policies (defined in the Configuration Profile) are sent to an MDM agent, the agent responds in the form of "Alert". 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", CommandFormatError, "Idle", "NotNow".
- Periodic reachability events are initiated by the MDM Server using Push Notifications. The ST does not select "configure periodicity of reachability events" in FMT_SMF_EXT.4.2.
- In cases where the HTTPS 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. MDM Server requests may be processed, for example, when 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.

## Guidance Assurance Activities

No assurance activities defined.

## Test Assurance Activities

### Assurance Activity AA-FAU_ALT_EXT.2-MDMA-ATE-01

- ***Test 1:*** *The evaluator shall perform a policy update from the test environment MDM server. The evaluator shall verify the MDM Agent accepts the update, makes the configured changes, and reports the success of the policy update back to the MDM Server.*
- ***Test 2:*** *The evaluator shall perform each of the actions listed in FAU_ALT_EXT.2.1 and verify that the alert does in fact reach the MDM Server.*
- ***Test 3:*** *The evaluator shall configure the MDM Agent to perform a network reachability test, both with and without such connectivity and ensure that results reflect each.*
- ***Test 4:*** *The evaluator shall remove network connectivity from the MDM Agent and generate an alert/event as defined in FAU_ALT_EXT.2.1. The evaluator shall restore network connectivity to the MDM Agent and verify that the alert generated while the TOE was disconnected is sent by the MDM Agent upon re-establishment of the connectivity.*

**Summary**

Test 1: The evaluator configured a passcode length policy and verified that the policy was correctly enforced by the TOE.

Test 2: The evaluator notes that the testing of the successful application of policies to the mobile device is tested with Test 1. This test covers the submission of an alert after the receipt of periodic reachability events.

Test 3: The evaluator re-performed Test 2, but prior to sending the notification to the client, the clients are disconnected from the WLAN. Upon reconnection, the notification were automatically sent.

Test 4: The evaluator disconnected the device from the WLAN and re-ran Test 1 without the steps for verification of actions. He then verified a task is added to the list of Active Tasks and reconnected the device to the WLAN. Within a minute, the new password setting was found to be configured. The evaluator then verified on the Profile Manager web UI that the Completed Tasks list shows "Push Settings" with the date and time of the configuration.

## 2.2.1.2 Audit Data Generation (FAU_GEN.1)

### TSS Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-ASE-01

*The evaluator shall check the TSS and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP is described and that the description of the fields contains the information required in FAU_GEN.1.2.*

#### Summary

Section 8.10.1 *Audit Records* in the TSS of [ST] describes audit records. It references Annex C *Audit Events* which provides listing of the auditable events for the required SFRs and provide example audit reords. In addition, Annex C also describes the audit records along with the standard audit record format.

Each audit record, at a minium, contains the following information

- Date and time of the event
- Type of event (this is described as log level and log tag)
- Subject identity (this is described as PID and PPID)
- The outcome (success or failure) of the event
- Any applicable required additional information

Table 20 "Example audit log" shows an example of each field in an audit record that correponds with the format described above. These fields contain the information required in FAU_GEN.1.2. In addition, tables 21 through 24 list the auditable events for the required SFRs and provide example audit records which also show the format described above.

### Guidance Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-AGD-01

*The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP including those listed in the Management section. The evaluator shall examine the administrative guide and make a determination of which administrative commands are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP. The*

*evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are security relevant with respect to this PP. The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.*

**Summary**

[CCGUIDE] chapter 6 *Security Audit* describes the audit function

Section 6.1 *Audit Logging* provides a sample audit record in Table 16 "Example audit log" showing how the fields in this audit record map t o the required fields specified in FAU_GEN.1. The sample outlines the basic format for each audit record showing that the audit records include at minimum the following information:

- date and time of the event;
- type of event (this is described as log level and log tag)
- subject identity (this is described as PID and PPID)
- the outcome (success or failure) of the event and
- any applicable required additional information

As stated at the beginning of section 6.1 of [CCGUIDE] , the available commands and responses constitute audit records and must be configured by the TOE administrators using Configuration Profiles. The details for profile implementation and audit record collection are documented in [DEV_MAN] , [PROFS_LOGS] , and [LOGGING] . The evaluator examined these documents along with [ST] and [CCGUIDE] as well as through testing, the evaluator determined that the guidance contains all the administrative actions and their associated audit events that are relevant to [PP_MDF_V3.2] and [MOD_MDM_AGENT_V1.0] , [MOD_BT_V1.0] , [MOD_VPNC_V2.3] , [PP_WLAN_CLI_EP_V1.0] , and [PKG_TLS_V1.1] , and through use of the TOE. These administrative actions are found to be consistent with the actions specified in [ST] .

### Test Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-ATE-01

*The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the provided table and administrative actions. This should include all instances of an event. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. For administrative actions, the evaluator shall test that each action determined by the evaluator above to be security relevant in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields specified in FAU_GEN.1.2 are contained in each audit record.*

*Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.*

**Summary**

The evaluator triggered all events listed in the provided table and administrative actions and used the console provided by Xcode to view and save the devices' logs.

## 2.2.1.3 Audit Data Generation (FAU_GEN.1(2))

### TSS Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-2-MDMA-ASE-01

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

*If "invoke platform-provided functionality" is selected, the evaluator shall examine the TSS to verify that it describes (for each supported platform) how this functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this evaluation activity).*

**Summary**

Section 8.10.1 *Audit Records* states that specific audit record is determined via the Configuration Profile, that at minimum includes the following information:

- 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)

This section also states that 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 2 or to an MDM server on enrolled devices.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-2-MDMA-ATE-01

*The evaluator shall use the TOE to perform the auditable events defined in the Auditable Events table in FAU_GEN.1.1(2) and observe that accurate audit records are generated with contents and formatting consistent with those described in the TSS. Note that this testing can be accomplished in conjunction with the testing of the security mechanisms directly.*

**Summary**

This testing is performed is conjunction with FAU_GEN.1

The evaluator performed all audible actions and recorded the device logs through the Xcode console. The evaluator verified that accurate audit logs are generated with contents and formatting consistent with those described in the TSS.

## 2.2.1.4 Audit Data Generation (Bluetooth) (FAU_GEN.1/BT)

### TSS Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-BT-ASE-01

*There are additional auditable events that serve to extend the FAU_GEN.1 SFR found in each Base-PP.*

*This SFR is evaluated in the same manner as defined by the Evaluation Activities for the claimed Base-PP. The only difference is that the evaluator shall also assess the auditable events required for this PP-Module in addition to those defined in the claimed Base-PP.*

**Summary**

This assurance activity is performed in conjunction with the assurance activity for FAU_GEN.1 from the Base-PP MDF-PP.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.1.5 Audit Data Generation (FAU_GEN.1/VPN)

### TSS Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-VPN-ASE-01

*The evaluator shall examine the TSS to determine that it describes the auditable events and the component that is responsible for each type of auditable event.*

#### Summary

Per [CCEVS-TD0663], the audit listing for MDF as required by this assurance activity is moved to guidance (AGD). The evaluator confirmed that this information is indeed provided in admiistrative guidance.

### Guidance Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-VPN-AGD-01

*The evaluator shall check the operational guidance and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the VPN Client PP-Module is described and that the description of the fields contains the information required in FAU_GEN.1.2/VPN, and the additional information specified in the Auditable Events table of the VPN Client PP-PP-Module.*

*In particular, the evaluator shall ensure that the operational guidance is clear in relation to the contents for failed cryptographic events. In the Auditable Events table of the VPN Client PP-Module, information detailing the cryptographic mode of operation and a name or identifier for the object being encrypted is required. The evaluator shall ensure that name or identifier is sufficient to allow an administrator reviewing the audit log to determine the context of the cryptographic operation (for example, performed during a key negotiation exchange, performed when encrypting data for transit) as well as the non-TOE endpoint of the connection for cryptographic failures relating to communications with other IT systems.*

*The evaluator shall also make a determination of the administrative actions that are relevant in the context of the VPN Client PP-Module. The TOE may contain functionality that is not evaluated in the context of the VPN Client PP-Module because the functionality is not specified in an SFR. This functionality may have administrative aspects that are described in the operational guidance. Since such administrative actions will not be performed in an evaluated configuration of the TOE, the evaluator shall examine the operational guidance and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the VPN Client PP-Module, which thus form the set of “all administrative actions”. The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.*

*For each required auditable event, the evaluator shall examine the operational guidance to determine that it is clear to the reader where each event is generated (e.g. the TSF may generate its own audit logs in one location while the platform-provided auditable events are generated elsewhere).*

#### Summary

Audit related guidance is provided in chapter 6 *Security Audit* where the required audit events are listed along with description of audit records, audit fields and sample audit records. In addition, the evaluator examined these details when performing the related assurance activity for the Base-PP MDF PP where the evaluator determined that the necesssary guidance was provided.

### Test Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-VPN-ATE-01

*The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records in accordance with the EAs associated with the functional requirements in the VPN Client PP-Module. Additionally, the evaluator shall test that each administrative action applicable in the context of the VPN Client PP-Module is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.*

*Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.*

#### Summary

This testing is performed in conjunction with testing of FAU_GEN.1.

## 2.2.1.6 Audit Data Generation (Wireless LAN) (FAU_GEN.1/WLAN)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

#### Assurance Activity AA-FAU_GEN.1-WLAN-AGD-01

*The evaluator shall check the operational guidance and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the EP is described and that the description of the fields contains the information required in FAU_GEN.1.2, and the additional information specified in Table 2.*

*The evaluator shall in particular ensure that the operational guidance is clear in relation to the contents for failed cryptographic events. In Table 2, information detailing the cryptographic mode of operation and a name or identifier for the object being encrypted is required. The evaluator shall ensure that name or identifier is sufficient to allow an administrator reviewing the audit log to determine the context of the cryptographic operation (for example, performed during a key negotiation exchange, performed when encrypting data for transit) as well as the non-TOE endpoint of the connection for cryptographic failures relating to communications with other IT systems.*

*The evaluator shall also make a determination of the administrative actions that are relevant in the context of this EP. The TOE may contain functionality that is not evaluated in the context of this EP because the functionality is not specified in an SFR. This functionality may have administrative aspects that are described in the operational guidance. Since such administrative actions will not be performed in an evaluated configuration of the TOE, the evaluator shall examine the operational guidance and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the EP, which thus form the set of "all administrative actions". The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.*

#### Summary

[CCGUIDE] chapter 6 *Security Audit* describes the audit function and provides Tables 16 through table 21 listing the audit events corresponding with the requirements specified in [ST]. In particular, table 17 "{MDF} auditable events" and table 18 "{WLAN} auditable events" describe the audit records required by the base PP [PP_MDF_V3.2] and [PP_WLAN_CLI_EP_V1.0]

Additionally, section 6.1 *Audit Logging* provides a sample audit record showing how the fields in this audit record map to the required fields specified in FAU_GEN.1.2. The sample outlines the basic format for each audit record showing that the audit records include at minimum the following information:

- date and time of the event;
- type of event (this is described as log level and log tag)
- subject identity (this is described as PID and PPID)
- the outcome (success or failure) of the event and
- any applicable required additional information

As stated at the beginning of section 6.1 of [CCGUIDE] , the available commands and responses constitute audit records and must be configured by the TOE administrators using Configuration Profiles. The details for profile implementation and audit record collection are documented in [DEV_MAN] , [PROFS_LOGS] , and [LOGGING] . The evaluator examined these documents along with [ST] and [CCGUIDE] as well as through testing, the evaluator determined that the guidance contains all the administrative actions and their associated audit events that are relevant to [PP_WLAN_CLI_EP_V1.0] , and through use of the TOE. These administrative actions are found to be consistent with the actions specified in [ST] .

For [PP_WLAN_CLI_EP_V1.0], the evaluator determined from the [ST] that the TOE doesn't claim to generate audit records for failed cryptographic events related to cryptographic services, thus, [CCGUIDE] is not required to proivide guidance for such audit records For connection for cryptographic failures relating to communications with other IT systems, table 18 in [CCGUIDE] cover these events including failure to establish an EAP-TLS session (FCS_TLSC_EXT.1/WLAN {WLAN}, establish / terminiation of an EAP-TLS session (FCS_TLSC_EXT.1/WLAN {WLAN}), failture to validate X.509v3 certificate (FIA_X509_EXT.1/WLAN {WLAN}), and all attempts to establish a trsuted channel (FTP_ITC_EXT.1/WLAN {WLAN})

The evaluator made the following observations:

- The audit records described in tables 17 and 18 provide the required information such as timestamps and subject identity.
- The SFRs covered in tables 17 and 18 match with Table 3 "Combined mandatory auditable events from [PP_MDF_V3.2] and [PP_WLAN_CLI_EP_V1.0]" of [ST] that together cover the mandatory auditable events required by the base PP [PP_MDF_V3.2] and [PP_WLAN_CLI_EP_V1.0]
- Each SFR from Table 3 [ST] is covered by at least one audit record and vice versa.

## Test Assurance Activities

### Assurance Activity AA-FAU_GEN.1-WLAN-ATE-01

*The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records in accordance with the assurance activities associated with the functional requirements in this EP. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.*

*Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.*

**Summary**

This test is performed in conjunction with FAU_GEN.1. The evaluator loaded a configuration profile on the TOE to activate logging and then attempted connections to a wireless Access Point using both an invalid and valid password and collected the logs for each.

## 2.2.1.7 Security Audit Event Selection (FAU_SEL.1(2))

### TSS Assurance Activities

#### Assurance Activity AA-FAU_SEL.1-2-MDMA-ASE-01

*If "invoke platform-provided functionality" is selected, the evaluator shall examine the TSS of the ST to verify that it describes (for each supported platform) how this functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the MDM Agent; nonetheless, that mechanism will be identified in the TSS as part of this evaluation activity).*

**Summary**

The ST selects "implement functionality" in FAU_SEL.1(2) {AGENT}, therefore this assurance activity is not applicable.

### Guidance Assurance Activities

#### Assurance Activity AA-FAU_SEL.1-2-MDMA-AGD-01

*The evaluator shall examine the operational guidance to determine that it contains instructions on how to define the set of auditable events as well as explains the syntax for multi-value selection (if applicable). The evaluator shall also verify that the operational guidance shall identify those audit records that are always recorded, regardless of the selection criteria currently being enforced.*

**Summary**

[CCGUIDE] section 6.2 *Audit Storage* and section 6.3 *Configure the Auditable Items* provide guidance for how to select auditable events to be audited. These sections state the following:

- The TOE OS has a logging framework that is used to configure different logging levels for the various TOE OS subsystems. This framework is configured by creating and installing a logging configuration profile property list file (i.e., .plist file) into the appropriate directory. More information may be found in [LOGGING] .
- [AConfig] describes how to use the device console to see all logged information that occurs between the device, Apple Configurator 2, and possibly connections outside your network to your mobile device management (MDM) solution or Apple. Administrators can mark a selection, clear the window to view a specific event, or save the log for troubleshooting.
- Per [PROFS_LOGS] , additional logs can be specified by performing user actions on a device or through using a Configuration Profile.
- Table 21 "Additional Audit Logs" identifies all the logs that can be optionally selected and how they can be initiated where the majority of which requires a configuration profile.

### Test Assurance Activities

#### Assurance Activity AA-FAU_SEL.1-2-MDMA-ATE-01

- ***Test 1:*** *For each attribute listed in the requirement, the evaluator shall devise a test to show that selecting the attribute causes only audit events with that attribute (or those that are always recorded, as identified in the administrative guidance) to be recorded.*
- ***Test 2:*** *[conditional]: If the TSF supports specification of more complex audit pre-selection criteria (e.g., multiple attributes, logical expressions using attributes) then the evaluator shall devise tests showing that this capability is correctly implemented. The evaluator shall also, in the test plan, provide a short narrative justifying the set of tests as representative and sufficient to exercise the capability.*

**Summary**

Test 1: The evaluator performed this testing as part of FAU_GEN.1.

Test 2 is not applicable since the functionality is not claimed in [ST].

## 2.2.1.8 Audit Storage Protection (FAU_STG.1)

### TSS Assurance Activities

#### Assurance Activity AA-FAU_STG.1-ASE-01

*The evaluator shall ensure that the TSS lists the location of all logs and the access controls of those files such that unauthorized modification and deletion are prevented.*

**Summary**

Section 8.10.1 *Audit Records* describes audit records performed by the TOE. This section specifies the storage location for the audit records which depends on operating system of the trusted workstation or MDM server, can be the following:

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

This section also states that 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 2 or to an MDM server on enrolled devices.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FAU_STG.1-ATE-01

- ***Test 1:*** *The evaluator shall attempt to delete the audit trail in a manner that the access controls should prevent (as an unauthorized user) and shall verify that the attempt fails.*
- ***Test 2:*** *The evaluator shall attempt to modify the audit trail in a manner that the access controls should prevent (as an unauthorized application) and shall verify that the attempt fails.*

**Summary**

Test 1 & Test 2: During the test for FAU_GEN.1, the evaluator observed that the interface to obtain the audit log information is read-only. This means that the interface is not capable of allowing a modify/write/delete operation. The audit trail is stored by the TOE such that a user application can only access the provided interface allowing the read operation. The audit log is kept in a storage location on the TOE devices where only the operating system can write to. The audit data is protected from modification attempts by user processes using the standard TOE OS protection mechanisms. The reading is limited to the log framework. That log framework can be accessed using tools provided with Xcode. Thus, the tester uses the Xcode log framework tool to access the audit logs to verify the audit log entries.

While obtaining the audit log, the evaluator tried to both modify and delete the log and verified that both the modification and deletion did not succeed.

### 2.2.1.9 Prevention of Audit Data Loss (FAU_STG.4)

#### TSS Assurance Activities

##### Assurance Activity AA-FAU_STG.4-ASE-01

*The evaluator shall examine the TSS to ensure that it describes the size limits on the audit records, the detection of a full audit trail, and the action(s) taken by the TSF when the audit trail is full. The evaluator shall ensure that the action(s) results in the deletion or overwrite of the oldest stored record.*

**Summary**

Section 8.10.1 *Audit Records* describes the audit function. This section states that the audit storage capacity is defined by the TOE administrators using the Configuration Profiles, as well as the default action when the storage capacity is reached.

#### Guidance Assurance Activities

No assurance activities defined.

#### Test Assurance Activities

No assurance activities defined.

## 2.2.2 Cryptographic support (FCS)

### 2.2.2.1 Cryptographic Key Generation (FCS_CKM.1)

#### TSS Assurance Activities

##### Assurance Activity AA-FCS_CKM.1-ASE-01

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

**Summary**

Section 8.3.4 *Explanation of usage for cryptographic functions* in the TSS of [ST] provides Table 10 "Explanation of usage for cryptographic functions in the cryptographic modules". Table entry "FCS_CKM.1 Cryptographic Key Generation" identifies the following schemes for asymmetric key pair generation:

- ECC scheme - ECDSA KeyGen and KeyVer with curves P-256, P-384, and Curve25519
- FFC scheme - Diffie-Hellman Group 14 with key size 2048-bits
- “safe-prime” groups that meet the following: ‘NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”’

The evaluator found this information to be consistent with the defintion of FCS_CKM.1 {MDF} {VPN}.

## Guidance Assurance Activities

### Assurance Activity AA-FCS_CKM.1-AGD-01

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

### Summary

The developer provided the following document as the main administrative guide for configuring and using the TOE in the evaluated configuration:

- Apple iOS 15: iPhones and Apple iPadOS 15: iPads Common Criteria Configuration Guide as the main guidance document, hereafter shorthanded as [CCGUIDE] .

Section 3 *Introduction* of this guidance document contains the following important statement:

> *This document is written for administrators and users of Apple mobile devices that are managed using a mobile device management (MDM) solution. The “Apple iOS 15: iPhones Security Target” [IOS_ST] and “Apple iPadOS 15: iPads Security Target” [IPADOS_ST] includes specifications for security where the mobile device operating environment includes a Wi-Fi network and includes evaluation of the Always-On virtual private network (VPN) provided by iOS/iPadOS. This configuration guide applies to both evaluations VID11237 (Apple iOS 15: iPhones) and VID11238 (Apple iPadOS 15: iPads). Where applicable, the guide shall point out the differences between iPhones and iPads.*

For this report, the evaluator will focus on information relevant to VID11237 (Apple iOS 15: iPhones).

[CCGUIDE] section 3.7 *Security Functional Requirements (SFRs) in the STs requiring configuration* contains Table 4: "SFR Configuration Requirements". Table entry for FCS_CKM.1(1) {MDF} {VPN} indicates that there is no configuration needed. This table entry also references section 5.2.1 *Key Generation, Signature Generation and Verification* which states:

> " *For the evaluated configuration, no configuration is required from the mobile device user.* "

and

> " *For the evaluated configuration, no configuration is required from the mobile device administrator.* "

## Test Assurance Activities

### Assurance Activity AA-FCS_CKM.1-ATE-01

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

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

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

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

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

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

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

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

***Key Generation for FIPS 186-4 Elliptic Curve Cryptography (ECC)***

*FIPS 186-4 ECC Key Generation Test*

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

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

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

***Key Generation for Curve25519***

*The evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated as specified in RFC 7748 using an approved random bit generator (RBG) and shall be written in little-endian order (least significant byte first). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.*

*Note: Assuming the PKV function of the good implementation will (using little-endian order):*

1. *confirm the private and public keys are 32-byte values*
2. *confirm the three least significant bits of the first byte of the private key are zero*
3. *confirm the most significant bit of the last byte is zero*

*4. confirm the second most significant bit of the last byte is one*

*5. calculate the expected public key from the private key and confirm it matches the supplied public key*

*The evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify 5 of the public key values so that they are incorrect, leaving five values unchanged (i.e. correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.*

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

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

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

*Cryptographic and Field Primes:*

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

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

*Cryptographic Group Generator:*

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

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

*Private Key:*

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

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

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

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

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

*for each FFC parameter set and key pair.*

***Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups***

*Testing for FFC Schemes using Diffie-Hellman group 14 and/or "safe-prime" groups is done as part of testing in FCS_CKM.2/UNLOCKED.*

## Summary

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

The evaluator performed CAVS-like testing using a tool developed by the lab for Curve25519.

## 2.2.2.2 VPN Cryptographic Key Generation (IKE) (FCS_CKM.1/VPN)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM.1-VPN-ASE-01

*The evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.*

#### Summary

Section 8.9.4.3 *VPN* in the TSS of [ST] describes that the key generation and key establishment for VPN are handled in the user space, while the bulk encryption is handled in the kernel space. The VPN seeds its private copy of the CTR_DRBG present in the corecrypto instance used by the VPN client with 384 bits of entropy, invokes the corecrypto ECDH or DH APIs to generate the appropriate keys using key material generated by the CTR_DRBG, and follows the key establishment algorithms defined in [SP800-56A], ECDSA key generation algorithm in [FIPS 186-4], Diffie-Hellman Group (MODP) key generation [RFC 3526], and IKEv2 key derivation function in [RFC 5996].

### Guidance Assurance Activities

#### Assurance Activity AA-FCS_CKM.1-VPN-AGD-01

*There are no AGD EAs for this requirement.*

#### Summary

There is no AGD evaluation activity for this SFR.

### Test Assurance Activities

#### Assurance Activity AA-FCS_CKM.1-VPN-ATE-01

*If this functionality is implemented by the TSF, refer to the following EAs, depending on the TOE's claimed Base-PP:*

- *GPOS PP: FCS_CKM.1*
- *MDF PP: FCS_CKM.1*
- *App PP: FCS_CKM.1(1)*
- *MDM PP: FCS_CKM.1*

#### Summary

[MOD_VPNC_V2.3] refers to EAs for FCS_CKM.1 specified in [PP_MDF_V3.2] and defines no additional tests.

## 2.2.2.3 Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) (FCS_CKM.1/WLAN)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM.1-WLAN-ASE-01

*The evaluator shall verify that the TSS describes how the primitives defined and implemented by this EP are used by the TOE in establishing and maintaining secure connectivity to the wireless clients. The TSS shall also provide a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed.*

**Summary**

Section 8.9.3 *Wireless LAN (WLAN)* in the TSS of [ST] describes the WLAN protocol. It states the the TOE implements the WLAN protcol as defined in the IEEE 802.11 (2012) as follow:

- The TOE implements the CTR with CBC-MAC protocol (CCMP) with AES (128-bit key) as defined in section 11.4.3 of IEEE 802.11 (2012)
- The TOE implements AES key wrapping as defined in [SP800-38F] 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). Newer device models support AES-CCMP-256 with 256-bit keys following IEEE 802.11ac-2013 as well as AES-GCMP-256 with 256-bit key sizes, respectively, following IEEE 802.11ac-2013
- AES key unwrapping used to unwrap the GTK is performed as described in [SP800-38F] section 6.1, Algorithm 2: W^-1(C), and in section 6.2, Algorithm 4: KW-AD(C).
- 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.

In addition, the TOE implements WPA2 Enterprise which received Wi-Fi Alliance certificates listed in Annex B *Wi-Fi Alliance Certificates* .

## Guidance Assurance Activities

No assurance activities defined.

## Test Assurance Activities

### Assurance Activity AA-FCS_CKM.1-WLAN-ATE-01

*The evaluator shall perform the following tests:*

- *Test 1: The evaluator shall configure the access point so the cryptoperiod of the session key is 1 hour. The evaluator shall successfully connect the TOE to the access point and maintain the connection for a length of time that is greater than the configured cryptoperiod. The evaluator shall use a packet capture tool to determine that after the configured cryptoperiod, a re-negotiation is initiated to establish a new session key. Finally, the evaluator shall determine that the renegotiation has been successful and the client continues communication with the access point.*
- *Test 2: The evaluator shall perform the following test using a packet sniffing tool to collect frames between the TOE and a wireless LAN access point:*
  - *Step 1: The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.*
  - *Step 2: The evaluator shall configure the TOE to communicate with a WLAN access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-f) pre-shared key. The pre-shared key is only used for testing.*
  - *Step 3: The evaluator shall start the sniffing tool, initiate a connection between the TOE and the access point, and allow the TOE to authenticate, associate, and successfully complete the 4 way handshake with the client.*
  - *Step 4: The evaluator shall set a timer for 1 minute, at the end of which the evaluator shall disconnect the TOE from the wireless network and stop the sniffer.*

- *Step 5: The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and derive the PTK from the 4-way handshake frames and pre- shared key as specified in IEEE 802.11-2012.*
- *Step 6: The evaluator shall select the first data frame from the captured packets that was sent between the TOE and access point after the 4-way handshake successfully completed, and without the frame control value 0x4208 (the first 2 bytes are 08 42). The evaluator shall use the PTK to decrypt the data portion of the packet as specified in IEEE 802.11-2012, and shall verify that the decrypted data contains ASCII-readable text.*
- *Step 7: The evaluator shall repeat Step 6 for the next 2 data frames between the TOE and access point and without frame control value 0x4208.*

**Summary**

Test 1: The evaluator connected the TOE to a WiFi Access point (AP) for over one hour and verified with Wireshark that the 4-way EAPOL handshake was automatically triggered after one hour.

Test 2: The evaluator connected the TOE to the AP and used Wireshark to obtain the anonce, snonce and MAC address from both devices to derive the PTK, KCK, KEK and TK keys. The evaluator then verified that the keys were consistent with the ones found in the logs.

## 2.2.2.4 Cryptographic Key Establishment (FCS_CKM.2/LOCKED)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_CKM.2-LOCKED-ATE-01

*The test for SP800-56A Revision 3 and SP800-56B key establishment schemes is performed in association with FCS_CKM.2/UNLOCKED.*

***Curve25519 Key Establishment Schemes***

*The evaluator shall verify a TOE's implementation of the key agreement scheme using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specification. These components include the calculation of the shared secret K and the hash of K.*

***Function Test***

*The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement role and hash function combination, the tester shall generate 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.*

*The evaluator shall obtain the shared secret value K, and the hash of K.*

*The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value K and compare the hash generated from this value.*

***Validity Test***

*The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results. To conduct this test, the evaluator generates a set of 30 test vectors consisting of data sets including the evaluator's public keys and the TOE's public/private key pairs.*

*The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value K or the hash of K. At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).*

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

**Summary**

An ACVP test harness extension was developed by the lab to implement ACVP-like tests for Curve25519. This tool generates ACVP-compliant test vectors in JSON format following the test definition in the PP. The evaluator used this tool to perform testing comparable to ACVP testing for Curve25519.

## 2.2.2.5 Cryptographic Key Establishment (FCS_CKM.2/UNLOCKED)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM.2-UNLOCKED-ASE-01

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

*If Diffie-Hellman group 14 is selected from FCS_CKM.2/UNLOCKED, the TSS shall describe how the implementation meets RFC 3526 Section 3.*

**Summary**

Section 8.3.4 *Explanation of usage for cryptographic functions* in the TSS of [ST] provides Table 10 "Explanation of usage for cryptographic functions in the cryptographic modules". Table entry "FCS_CKM.2/UNLOCKED" identifies the following schemes for key establishment:

- RSA
- ECC
- FFC - Diffie-Hellman group 14.
- FFC

The evaluator found this information corresponds with the key generation schemes specified in FCS_CKM.1.1 {MDF} {VPN}.

Section 8.9.4.3 in the TSS of [ST] states the following:

- Each of these cryptographic mechanisms are provided by one of the following two cryptographic modules: Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1] or Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1].
- The key generation and key establishment for VPN are handled in the user space, while the bulk encryption is handled in the kernel space. The VPN seeds its private copy of the CTR_DRBG present in the corecrypto instance used by the VPN client with 384 bits of entropy, invokes the corecrypto ECDH or DH APIs to generate the appropriate keys using key material generated by the CTR_DRBG, and follows the key establishment algorithms defined in [SP800-56A], ECDSA key generation algorithm in [FIPS 186-4], Diffie-Hellman Group (MODP) key generation [RFC 3526], and IKEv2 key derivation function in [RFC 5996].

## Guidance Assurance Activities

### Assurance Activity AA-FCS_CKM.2-UNLOCKED-AGD-01

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

### Summary

[CCGUIDE] Table 4 "SFR Configuration Requirements" identifies which SFR can be or needs to be configured in the evaluated configuration. For FCS_CKM.2/UNLOCKED {MDF} {VPN}, there is no configuration necessary for the selection of key establishment schemes by the TOE users or administrators. Table entry FCS_CKM.2/UNLOCKED {MDF} {VPN} references section 5.2.2 which also states:

" *For the evaluated configuration, no configuration is required from the mobile device user.* "

and

" *For the evaluated configuration, no configuration is required from the mobile device administrator.* "

## Test Assurance Activities

### Assurance Activity AA-FCS_CKM.2-UNLOCKED-ATE-01

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

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

***SP800-56A Revision 3 Key Establishment Schemes***

*The evaluator shall verify a TOE's implementation of SP800-56A Revision 3 key establishment schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.*

***Function Test***

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

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

*If the TOE does not use a KDF defined in SP 800-56A Revision 3, the evaluator shall obtain only the public keys and the hashed value of the shared secret.*

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

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

***Validity Test***

*The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A Revision 3 key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.*

*The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).*

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

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

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

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

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

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

*To conduct this test the evaluator shall generate or obtain test vectors FCS_CKM.2.1/LOCKED from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.*

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

***RSAES-PKCS1-v1_5 Key Establishment Schemes***

*The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses RSAES-PKCS1-v1_5.*

***Diffie-Hellman Group 14***

*The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses Diffie-Hellman Group 14.*

***FFC Schemes using "safe-prime" groups***

*The evaluator shall verify the correctness of the TSF's implementation of "safe-prime" groups by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses "safe-prime" groups. This test must be performed for each "safe-prime" group that each protocol uses.*

**Summary**

For ECDH, the evaluator confirmed that testing was performed in accordance with the CAVP test procedures. These results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

For RSA2, an ACVP test harness extension was developed by the lab to implement ACVP-like tests. This tool generates ACVP-compliant test vectors in JSON format following the test definition in the PP. The evaluator used this tool to perform testing comparable to ACVP testing for RSA2.

## 2.2.2.6 Cryptographic Key Distribution (GTK) (FCS_CKM.2/WLAN)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM.2-WLAN-ASE-01

*The evaluator shall check the TSS to ensure that it describes how the GTK is unwrapped prior to being installed for use on the TOE using the AES implementation specified in this EP.*

**Summary**

Section 8.9.3 *Wirelss LAN(WLAN)* in the TSS of [ST] describes the WLAN protocol. It states that the AES key unwrapping used to unwrap the GTK is performed as described in [SP800-38F] section 6.1, Algorithm 2: W^-1(C), and in section 6.2, Algorithm 4: KW-AD(C).

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_CKM.2-WLAN-ATE-01

*The evaluator shall perform the following test using a packet sniffing tool to collect frames between the TOE and a wireless access point (which may be performed in conjunction with the assurance activity for FCS_CKM.1.1/WLAN).*

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

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

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

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

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

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

*Step 7: The evaluator shall repeat Step 6 for the next 2 data frames with frame control value 0x4208.*

**Summary**

Performed in conjunction with FCS_CKM.1.1/WLAN Test 2: The evaluator connected the TOE to the AP and used Wireshark to obtain the anonce, snonce and MAC address from both devices to derive the PTK, KCK, KEK and TK keys. The evaluator then verified that the keys were consistent with the ones found in the logs.

## 2.2.2.7 Cryptographic Key Support (FCS_CKM_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.1-ASE-01

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

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

*The evaluator shall verify that the description includes how the OS is prevented from accessing the memory containing REK key material, which software is allowed access to the REK, how any other software in the execution environment is prevented from reading that key material, and what other mechanisms prevent the REK key material from being written to shared memory locations between the OS and the separate execution environment.*

*If key derivation is performed using a REK, the evaluator shall ensure that the TSS description includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to FCS_CKM_EXT.3.2.*

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

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

**Summary**

Section 8.2 *Hardware Protection Functions* and section 8.3 *Cryptographic Support* in the TSS of [ST] provide relevant description about key management.

Section 8.3.1 *Overview of Key Management* describes an overview of key management performed by the TOE. This description includes details on REK, the TOE UID used by the TOE.

Together these sections describe the TOE UID (the REK) which is stored in hardware-protected memory in the SEP and is not accessible to any other part of the system such as the application processors. The UID is not even known to Apple. It is generated during production using the hardware SP800-90A DRBG CTR_DRBG(AES) of the SEP and then etched into the silicon. When the TOE device starts up, an ephemeral key is created, entangled with the UID which is used to encrypt the Secure Enclave's portion of the device's memory space. The UID is never passed to any other part of the system. Even the software within the Secure Enclave cannot read the UID. It can only request encryption/decryption operations performed by a dedicated AES engine accessible only from the SEP. The UID is used to derive two other keys: 0x89B and 0x835. These keys are derived during first boot by encrypting defined constants with the UID. These keys are used to wrap the EMF key, which is the file system master key, and the DKey, which is the device key. The AES Key Wrap algorithm is used to encrypt the 0x89B and 0x835 keys. As stated above, the UID is not generated on the device, but rather at manufacturing time. It is generated in the production environment using a protected system with an RNG that complies with the requirements of NIST SP 800-90A (i.e., an approved SP 800-90A DRBG implementation), as specified in FCS_RBG_EXT.1.1 (SEP) {MDF} and FCS_RBG_EXT.1.2 (SEP) {MDF}.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.8 Cryptographic Key Random Generation (FCS_CKM_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.2-ASE-01

*The evaluator shall ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.*

*The evaluator shall also examine the key hierarchy section of the TSS to ensure that the formation of all DEKs is described and that the key sizes match that described by the ST author. The evaluator shall examine the key hierarchy section of the TSS to ensure that each DEK is generated or combined from keys of equal or greater security strength using one of the selected methods.*

- *If the symmetric DEK is generated by an RBG, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.*
- *If the DEK is formed from a combination, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR or a KDF to justify that the effective entropy of each factor is preserved. The evaluator shall also verify that each combined value was originally generated from an Approved DRBG described in FCS_RBG_EXT.1.*
- *If "concatenating the keys and using a KDF (as described in (SP 800-56C)" is selected, the evaluator shall ensure the TSS includes a description of the randomness extraction step.*

*The description must include how an approved untruncated MAC function is being used for the randomness extraction step and the evaluator must verify the TSS describes that the output length (in bits) of the MAC function is at least as large as the targeted security strength (in bits) of the parameter set employed by the key establishment scheme (see Tables 1-3 of SP 800-56C).*

*The description must include how the MAC function being used for the randomness extraction step is related to the PRF used in the key expansion and verify the TSS description includes the correct MAC function:*

- *If an HMAC-hash is used in the randomness extraction step, then the same HMAC-hash (with the same hash function hash) is used as the PRF in the key expansion step.*
- *If an AES-CMAC (with key length 128, 192, or 256 bits) is used in the randomness extraction step, then AES-CMAC with a 128-bit key is used as the PRF in the key expansion step.*
- *The description must include the lengths of the salt values being used in the randomness extraction step and the evaluator shall verify the TSS description includes correct salt lengths:*
  - *If an HMAC-hash is being used as the MAC, the salt length can be any value up to the maximum bit length permitted for input to the hash function hash.*
  - *If an AES-CMAC is being used as the MAC, the salt length shall be the same length as the AES key (i.e. 128, 192, or 256 bits).*

*(conditional) If a KDF is used, the evaluator shall ensure that the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 or SP 800-56C.*

**Summary**

Section 8.3 *Cryptographic Support* in the TSS of [ST] states the following:

- 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 SEP (i.e., FCS_RBG_EXT.1(SEP)). Files are encrypted using this key with AES in Xor-Encrypt-Xor-based tweaked-codebook mode with ciphertext stealing (XTS) where the initialization vector (IV) is calculated with the block offset into the file, encrypted with the SHA-1 hash of the per-file key, and follows [SP800-38E]. On Apple A14 and later devices and M1 and later devices, the encryption uses AES-256 in XTS mode in the SEP. On Apple A9 through A13 devices, the encryption uses AES-128 in XTS mode in the SEP where the 256-bit per file key is split to provide a 128-bit tweak and a 128-bit cipher key. (This is a general Apple ARM processor statement and all aforementioned processors may not be used by the devices in this TOE.
- 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, the TOE OS generates another asymmetric key pair within the SEP (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 Diffie-Hellman (DH) as described in [SP800-56A]). The KDF is Concatenation Key Derivation Function (Approved Alternative 1) as described in 5.8.1 of [SP800-56A]. 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.
- “Concatenating the keys and using a KDF (as described in (SP 800- 56C)” is selected in FCS_CKM_EXT.3 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. The implementation of the KDF uses HMAC-SHA-256 for both the extraction as well as the expansion phase. The salt length and the output key length of the KDF are each 256 bits.

The evaluator determined that the description is sufficiently detailed, the key management hiearchy both in text description as well as illustrated in figure 5 is clear and meets the requirements to ensure the keys are adequately protected. Also, the description expands in multiple paragraphs and includes a flow-chart diagram in figure 5 which clearly depicts the key hierarchy in the TOE OS.

## Guidance Assurance Activities

### Assurance Activity AA-FCS_CKM_EXT.2-AGD-01

*The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being generated or combined is identical to the key size and mode to be used for the encryption/decryption of the data.*

### Summary

[CCGUIDE] Table 4 "SFR Configuration Requirements" identifies which SFR can be or needs to be configured in the evaluated configuration. For FCS_CKM_EXT.2 {MDF}, there is no configuration necessary by the TOE users or administrators since generation and maintenance of DEK is hard coded.

This assurance activity is also performed in conjunction with the test assurance activities.

## Test Assurance Activities

### Assurance Activity AA-FCS_CKM_EXT.2-ATE-01

*If a KDF is used, the evaluator shall perform one or more of the following tests to verify the correctness of the key derivation function, depending on the mode(s) that are supported. Table 3 maps the data fields to the notations used in SP 800-108 and SP 800-56C.*

***Table 3: Notations used in SP 800-108 and SP 800-56C***

| ***Data Fields*** | ***Notations*** | |
|---|---|---|
| | ***SP 800-108*** | ***SP 800-56C*** |
| *Pseudorandom function* | *PRF* | *PRF* |
| *Counter length* | *r* | *r* |
| *Length of output of PRF* | *h* | *h* |
| *Length of derived keying material* | *L* | *L* |
| *Length of input values* | *l length* | *l length* |
| *Pseudorandom input values I* | *K1 (key derivation key)* | *Z (shared secret)* |
| *Pseudorandom salt values* | *n/a* | *s* |
| *Randomness extraction MAC* | *n/a* | *MAC* |

***Counter Mode Tests:***

*The evaluator shall determine the following characteristics of the key derivation function:*

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*

- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Location of the counter relative to fixed input data: before, after, or in the middle.*
  - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values I.*

*For each supported combination of I_length, MAC, salt, PRF, counter location, value of r, and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it. For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output.*

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

***Feedback Mode Tests:***

*The evaluator shall determine the following characteristics of the key derivation function:*

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Whether or not zero-length IVs are supported.*
- *Whether or not a counter is used, and if so:*
  - *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
  - *Location of the counter relative to fixed input data: before, after, or in the middle.*
    - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values I.*

*For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I and pseudorandom salt values. If the KDF supports zero-length IVs, five of these test vectors will be accompanied by pseudorandom IVs and the other five will use zero-length IVs. If zero-length IVs are not supported, each test vector will be accompanied by an pseudorandom IV. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.*

*For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.*

***Double Pipeline Iteration Mode Tests:***

*The evaluator shall determine the following characteristics of the key derivation function:*

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Whether or not a counter is used, and if so:*

- *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
- *Location of the counter relative to fixed input data: before, after, or in the middle.*
  - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*

- *The length (l_length) of the input values I.*

*For each supported combination of l_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.*

*For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.*

**Summary**

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.9 Cryptographic Key Generation (FCS_CKM_EXT.3)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.3-ASE-01

*The evaluator shall examine the key hierarchy section of the TSS to ensure that the formation of all KEKs are described and that the key sizes match that described by the ST author. The evaluator shall examine the key hierarchy section of the TSS to ensure that each key (DEKs, software-based key storage, and KEKs) is encrypted by keys of equal or greater security strength using one of the selected methods.*

*The evaluator shall review the TSS to verify that it contains a description of the conditioning used to derive KEKs. This description must include the size and storage location of salts. This activity may be performed in combination with that for FCS_COP.1/CONDITION.*

*(conditional) If the symmetric KEK is generated by an RBG, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.*

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

*(conditional) If the KEK is formed from a combination, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR, a KDF, or encryption.*

*(conditional) If a KDF is used, the evaluator shall ensure that the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108.*

*(conditional) If "concatenating the keys and using a KDF (as described in (SP 800-56C)" is selected, the evaluator shall ensure the TSS includes a description of the randomness extraction step. The description must include*

- *How an approved untruncated MAC function is being used for the randomness extraction step and the evaluator must verify the TSS describes that the output length (in bits) of the MAC function is at least as large as the targeted security strength (in bits) of the parameter set employed by the key establishment scheme (see Tables 1-3 of SP 800-56C).*
- *How the MAC function being used for the randomness extraction step is related to the PRF used in the key expansion and verify the TSS description includes the correct MAC function:*
  - *If an HMAC-hash is used in the randomness extraction step, then the same HMAC-hash (with the same hash function hash) is used as the PRF in the key expansion step.*
  - *If an AES-CMAC (with key length 128, 192, or 256 bits) is used in the randomness extraction step, then AES-CMAC with a 128-bit key is used as the PRF in the key expansion step.*
- *The lengths of the salt values being used in the randomness extraction step and the evaluator shall verify the TSS description includes correct salt lengths:*
  - *If an HMAC-hash is being used as the MAC, the salt length can be any value up to the maximum bit length permitted for input to the hash function hash.*
  - *If an AES-CMAC is being used as the MAC, the salt length shall be the same length as the AES key (i.e. 128, 192, or 256 bits).*

*The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.*

**Summary**

Section 8.3.1 *Overview of Key Management* of the [ST] describes the key management.

The evaluator reviewed section 8.3.1 and verified that all keys used for data encryption are 256-bit AES keys. Key Encryption Keys (KEKs) are also 256-bit AES keys. The passcode key is derived from the UID, the passcode and the salt using the concatenation Key Derivation Function (Approved Alternative 1) as described in 5.8.1 of NIST SP 800-56A. The KEKs are generated using an Approved NIST SP 800-90A as described in FCS_RBG_EXT.1(SEP).

Although KDF is used, the SP 800-56C KDF is implemented by the TOE and not the SP 800-108 KDF.

The ST selects "concatenating the keys and using a KDF (as described in (SP 800- 56C)" in FCS_CKM_EXT.3.

Section 8.3.3 *Randomness extraction and expansion step* in the TSS of [ST] describes the randomness extraction step. According to this section, the TOE implements the Key Derivation Function (KDF) based on RFC5869. The KDF defined in the RFC complies with SP 800-56C, as far as the extraction and expansions steps. The RFC specifies the order of the concatenation. The TOE's implementation of the KDF uses HMAC-SHA-256 for both the extraction and the expansion phase. Specifically, the evaluator verified that the extraction and expansion steps are explicitly described in this section which the evaluator determined to be a reproduction of sections 2.2 and 2.3 of RFC 5869. This section also states that the salt length and output key of the KDF are 256 bits.

The evaluator determined that the description is sufficiently detailed, the key management hiearchy both in text description as well as illustrated in figure 5 is clear and meets the requirements to ensure the keys are adequately protected. Also, the description expands in multiple paragraphs and includes a flow-chart diagram in figure 5 which clearly depicts the key hierarchy in the TOE OS.

## Guidance Assurance Activities

No assurance activities defined.

# Test Assurance Activities

## Assurance Activity AA-FCS_CKM_EXT.3-ATE-01

*If a KDF is used, the evaluator shall perform one or more of the following tests to verify the correctness of the key derivation function, depending on the mode(s) that are supported. Table 4 maps the data fields to the notations used in SP 800-108 and SP 800-56C.*

***Table 4: Notations used in SP 800-108 and SP 800-56C***

| Data Fields | Notations | |
|---|---|---|
| | SP 800-108 | SP 800-56C |
| Pseudorandom function | PRF | PRF |
| Counter length | r | r |
| Length of output of PRF | h | h |
| Length of derived keying material | L | L |
| Length of input values | I_length | I_length |
| Pseudorandom input values I | $K_1$ (key derivation key) | Z (shared secret) |
| Pseudorandom salt values | n/a | s |
| Randomness extraction MAC | n/a | MAC |

***Counter Mode Tests:***

*The evaluator shall determine the following characteristics of the key derivation function:*

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Location of the counter relative to fixed input data: before, after, or in the middle.*
  - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
  - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values I.*

*For each supported combination of I_length, MAC, salt, PRF, counter location, value of r, and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it. For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output.*

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

***Feedback Mode Tests:***

*The evaluator shall determine the following characteristics of the key derivation function:*

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *The length (in bits) of the output of the PRF (h).*

- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Whether or not zero-length IVs are supported.*
- *Whether or not a counter is used, and if so:*
  - *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
  - *Location of the counter relative to fixed input data: before, after, or in the middle.*
    - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values I.*

*For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I and pseudorandom salt values. If the KDF supports zero-length IVs, five of these test vectors will be accompanied by pseudorandom IVs and the other five will use zero-length IVs. If zero-length IVs are not supported, each test vector will be accompanied by an pseudorandom IV. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.*

*For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.*

***Double Pipeline Iteration Mode Tests:***

*The evaluator shall determine the following characteristics of the key derivation function:*

- *One or more pseudorandom functions that are supported by the implementation (PRF).*
- *The length (in bits) of the output of the PRF (h).*
- *Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.*
- *Up to two values of L that are NOT evenly divisible by h.*
- *Whether or not a counter is used, and if so:*
  - *One or more of the values {8, 16, 24, 32} that equal the length of the binary representation of the counter (r).*
  - *Location of the counter relative to fixed input data: before, after, or in the middle.*
    - *Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.*
    - *Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.*
- *The length (I_length) of the input values I.*

*For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.*

*For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.*

## Summary

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.10 Key Destruction (FCS_CKM_EXT.4)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.4-ASE-01

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

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

*The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored.*

#### Summary

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] how the TOE performs key management.

This section provides Table 8 "Summary of keys and persistent secrets in the TOE OS" and table 9 "Summary of keys and persistent secrets used by the MDM Agent" which describe all the key material, their purposes, and storage locations. Keys that are encrypted are not cleared unless the data they protect is erased. The DKey (device key) and the EMF key (file system master key) are stored wrapped, in block 0 of the flash memory (effaceable storage) and thus erasable very quickly if needed. The UID is stored unencrypted but is not accessible by any other part of the TOE system other than the SEP. All other keys are stored in non-volatile memory wrapped.

Also, section 8.3.2 *Storage of Persistent Secrets and Private Keys by the MDM Agent* in the TSS states that all clearing operations of keys stored in volatile memory are performed by the Apple corecrypto Module [Apple ARM, User, Software, SL1], Apple corecrypto Module [Apple ARM, Kernel, Software, SL1] or by the AES implementation within the Wi-Fi chip. Also, this section specifies that this clearing is performed using the memset(0) function to overwrite the memory containing keys and sensitive parameters. That applies to symmetric keys used for TLS, HTTPS, or Wi-Fi. All other keys are stored encrypted and therefore do not need to be wiped. When a wipe is triggered, the KEKs are erased and therefore all other encrypted keys do not need be zeroized.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.4-ATE-01

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

*For each software and firmware key clearing situation (including on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state) the evaluator shall repeat the following tests.*

*For these tests the evaluator shall utilize appropriate development environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.*

- ***Test 1:*** *Applied to each key held as plaintext in volatile memory and subject to destruction by overwrite by the TOE (whether or not the plaintext value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:*
  1. *Record the value of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Cause the TOE to stop the execution but not exit.*
  5. *Cause the TOE to dump the entire memory of the TOE into a binary file.*
  6. *Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.*
  7. *Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.*

  *Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails. Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a minuscule chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is found the test fails.*
- ***Test 2:*** *Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:*
  1. *Record the value of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Search the non-volatile memory the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.*
  5. *Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for test 1 above), and if a fragment is found in the repeated test then the test fails.*
- ***Test 3:*** *Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:*
  1. *Record the storage location of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Read the storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.*

  *The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.*

**Summary**

This test is performed as part of the zeroization testing in FIPS 140-3 conformance testing which is also performed by the evaluator. As all software requiring cryptographic support use CoreCrypto, the zeroization tests performed as part of FIPS 140-3 cover this test. These test results are included in evidence.

# 2.2.2.11 TSF Wipe (FCS_CKM_EXT.5)

## TSS Assurance Activities

### Assurance Activity AA-FCS_CKM_EXT.5-ASE-01

*The evaluator shall check to ensure the TSS describes how the device is wiped, the type of clearing procedure that is performed (cryptographic erase or overwrite) and, if overwrite is performed, the overwrite procedure (overwrite with zeros, overwrite three or more times by a different alternating pattern, overwrite with random pattern, or block erase).*

*If different types of memory are used to store the data to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, data stored on flash are cleared by overwriting once with zeros, while data stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write).*

**Summary**

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes how the TOE performs key management.

This section states that 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 effectively wiped. The overwriting of the KEKs is done through a memset(0) operation that overwrites the memory as detailed in the assurance activity for FCS_CKM_EXT.4 above.

## Guidance Assurance Activities

### Assurance Activity AA-FCS_CKM_EXT.5-AGD-01

*The evaluator shall verify that the AGD guidance describes how to enable encryption, if it is not enabled by default. Additionally the evaluator shall verify that the AGD guidance describes how to initiate the wipe command.*

**Summary**

[CCGUIDE] Table 4 "SFR Configuration Requirements" identifies which SFR can be or needs to be configured in the evaluated configuration. Table entry FCS_CKM_EXT.2 {MDF} references section 5.4.1 and section 5.4.3 for how to enable encryption and how to wipe the device, respectively.

For enabling encryption, section 5.4.1 states the following:

- To ensure data at rest protection, establishment of a passcode on the mobile device is required.
- Users can check that data at rest protection is enabled on their device by checking whether passcode is enabled on their device. This screen allows the user to enable data protection by enabling these ID features.
- Mobile device administrators must ensure that mobile device users set a passcode by using the forcePIN key in the Passcode Payload.
- Mobile device administrators can restrict USB drive access in the Files app if desired by setting the allowFilesUSBDriveAccess key to ‘false’ in the “Restrictions” section of the Configuration Profile.
- Mobile device administrators must ensure through organizational policies that mobile device users only use external storage devices formatted in the APFS format with encrypted volumes. Unencrypted volumes and other formats are not allowed in the evaluated configuration.

- The TOE only supports external storage encryption with storage devices formatted in the APFS format, other formats with encryption or encrypted volumes are not supported by the TOE. In the evaluated configuration, external storage devices must be formatted in the APFS file format and volumes must be encrypted. All other storage formats are not allowed in the evaluated configuration. Instructions to format devices in the APFS format with encrypted volumnes is provided in [APFS_DOC]

For wiping protected data, section 5.4.3 staes the following:

- A wipe operation is performed after the mobile device user exceeds the limit of the number of failed authentication attempts or upon receiving a request from an authorized administrator. The administrator can configure the number of failed attempts by using the following Configuration Profile key in the Passcode Payload: maxFailedAttempts.
- It is mandatory that the mobile device administrator can issue a remote wipe command from the MDM server using the MDM protocol. Subsection 5.4.3.3 provides guidance for administrator to configure and issue a remote wipe using a Configuration Profile with an MDM Payload and specific keys.
- The mobile device user can wipe the device themselves via the device UI: Settings » General » Transfer or Reset iPhone/iPad » Erase All Content and Settings

### Test Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.5-ATE-01

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

- ***Test 1:*** *The evaluator shall perform one of the following tests. The test before and after the wipe command shall be identical. This test shall be repeated for each type of memory used to store the data to be protected.*
  - ***Test 1.1: For File-based Methods:*** *The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a user data (protected data or sensitive data) file, for example, by using an application. The evaluator shall use a tool provided by the developer to examine this data stored in memory (for example, by examining a decrypted files). The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF_EXT.1. The evaluator shall use a tool provided by the developer to examine the same data location in memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).*
  - ***Test 1.2: For Volume-based Methods:*** *The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a unique data string, for example, by using an application. The evaluator shall use a tool provided by the developer to search decrypted data for the unique string. The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF_EXT.1. The evaluator shall use a tool provided by the developer to search for the same unique string in decrypted memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).*
- ***Test 2:*** *The evaluator shall cause the device to wipe and verify that the wipe concludes with a power cycle.*

#### Summary

When effaceable storage is reserved, a handle is provided for access. No pointer directly to the storage location is provided, therefore the memory cannot be read directly. The debugging of the AppleKeystore contents done for FCS_CKM_EXT.4 after a complete wipe of the device shows that new keys are generated by SEP and therefore loaded onto the new effaceable memory.

## 2.2.2.12 Salt Generation (FCS_CKM_EXT.6)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.6-ASE-01

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

**Summary**

Section 8.2.1 *The Secure Enclave Processor (SEP)* describes how the SEP performs salt generation. The SEP uses its own phyiscal noise source and RNG (defined in FCS_RBG_EXT.1(SEP) {MDF} to generate the 128-bit salt value for the PBKDF (specifically PBKDF2), which uses AES with the TOE device UID as the key for the psuedo-random function (PRF). This salt is regenerated every time the passcode changes. All other salt values used in the TOE OS are generated using the TRNG of the application processor, post-processed by a CTR-based SP 800-90A DRBG implementation claimed in FCS_RBG_EXT.1. This includes nonces used in the generation of digital signature algorithm (DSA) signatures as well as nonces required for the Wi-Fi and TLS protocol.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.13 Cryptographic Key Support (REK) (FCS_CKM_EXT.7)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.7-ASE-01

*The evaluation activity for this component is performed in conjunction with the evaluation activity for FCS_CKM_EXT.1.*

**Summary**

This assurance activity was performed in conjunction with the assurance activity for FCS_CKM_EXT.1.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.14 Bluetooth Key Generation (FCS_CKM_EXT.8)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.8-ASE-01

*The evaluator shall ensure that the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs. In particular, the evaluator shall ensure that the implementation does not permit the use of static ECDH key pairs.*

**Summary**

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- The TOE generates a new ephemeral ECDH key pair for every new connection attempt. No data can be transferred via Bluetooth until pairing has been completed. The TOE terminates the connection if the remote device stops encryption while connected to the TOE.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_CKM_EXT.8-ATE-01

*The evaluator shall perform the following steps:*

*Step 1: Pair the TOE to a remote Bluetooth device and record the public key currently in use by the TOE. (This public key can be obtained using a Bluetooth protocol analyzer to inspect packets exchanged during pairing.)*

*Step 2: Perform necessary actions to generate new ECDH public/private key pairs. (Note that this test step depends on how the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs.)*

*Step 3: Pair the TOE to a remote Bluetooth device and again record the public key currently in use by the TOE.*

*Step 4: Verify that the public key in Step 1 differs from the public key in Step 3.*

#### Summary

The evaluator deployed the bluetooth logging profile to the TOE and rebooted to enable bluetooth logging. He then performed the following steps with several bluetooth devices.

- pair the TOE with the bluetooth device
- dump system memory to obtain the ECDH key in the bluetooth log data
- unpair the TOE and the bluetooth device
- pair the TOE and the bluetooth device a second time
- dump system memory as done in the first step
- unpair the TOE and the bluetooth device

For each bluetooth device, the evaluator confirmed the ECDH key used in the second pairing operation was not the same as the first.

## 2.2.2.15 Cryptographic Operation (FCS_COP.1/CONDITION)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_COP.1-CONDITION-ASE-01

*The evaluator shall check that the TSS describes the method by which the password is first encoded and then fed to the SHA algorithm and verify the SHA algorithm matches the first selection.*

*If a key stretching function, such as PBKDF2, is selected the settings for the algorithm (padding, blocking, etc.) shall be described. The evaluator shall verify that the TSS contains a description of how the output of the hash function or key stretching function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_CKM_EXT.3.*

*If any manipulation of the key is performed in forming the submask that will be used to form the KEK, that process shall be described in the TSS.*

#### Summary

Section 8.3.1.1 *Password based key derivation* in the TSS of [ST] describes the password-based key derivation. It states the following:

- The TOE implements PBKDF2 to derive a 256-bit key from a user's passcode. The PBKDF2 is implemented as specified in [SP800-132] following “Option 2b” defined in section 5.4 of the standard. It uses HMAC-SHA-256 as the pseudorandom function (PRF).
- The input to the PBKDF2 is the 128-bit random salt generated by the SEP (as described in section 8.2.1), the user’s passcode without any pre-processing, and an iteration count of one. The output is the 256-bit key mentioned above.
- Next, the output of the PBKDF2 is repeatedly encrypted with the AES-CBC-256 hardware cipher using the 256-bit UID as the encryption key to generate 256 bits of data with each loop iteration. The loop is performed as often as needed to reach a duration between 100 and 150 milliseconds on that device.
- The output after all AES iterations have completed forms the 256-bit root encryption key used to unwrap the user’s keybag that holds the class keys for the file system data protection. Only when the unwrapping of the user’s keybag is successful is the user considered authenticated.
- The number of AES-CBC-256 iterations is calibrated to take at least 100 to 150 milliseconds and is a minimum of 50,000. The number of iterations is device-specific and may be greater than 50,000 on some devices.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.16 Cryptographic Operation (FCS_COP.1/ENCRYPT)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_COP.1-ENCRYPT-ATE-01

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

***AES-CBC Tests***

- ***Test 1:*** *AES-CBC Known Answer Tests There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.*
  - ***Test 1.1:*** *KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.*

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

- ***Test 1.2:*** *KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.*
  *To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.*
- ***Test 1.3:*** *KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].*
  *To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.*
- ***Test 1.4:*** *KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].*
  *To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.*

- ***Test 2:*** *AES-CBC Multi-Block Message Test*
  *The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.*
  *The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.*
- ***Test 3:*** *AES-CBC Monte Carlo Tests*
  *The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:*

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

  *The ciphertext computed in the 1000 $^{th}$ iteration (i.e. CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.*
  *The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.*

***AES-CCM Tests***

- ***Test 1:*** *The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:*

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

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

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

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

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

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

- ***Test 1.1:*** *For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- ***Test 1.2:*** *For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- ***Test 1.3:*** *For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.*
- ***Test 1.4:*** *For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*

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

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

***AES-GCM Test***

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

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

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

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

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

- ***Test 1:*** *The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.*
- ***Test 2:*** *The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.*

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

***XTS-AES Test***

- ***Test 1:*** *The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:*

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

  ***Three data unit (i.e. plaintext) lengths.*** *One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.*

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

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

- ***Test 2:*** *The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.*

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

- ***Test 1:*** *The evaluator shall test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:*

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

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

  *using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall use the AES-KW authenticated-encryption function of a known good implementation.*
- ***Test 2:*** *The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.*
- ***Test 3:*** *The evaluator shall test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:*
  - *One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).*
  - *One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).*
- ***Test 4:*** *The evaluator shall test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.*

**Summary**

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.17 Cryptographic Operation (FCS_COP.1/HASH)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_COP.1-HASH-ASE-01

*The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS. The evalutator shall check that the TSS indicates if the hashing function is implemented in bit-oriented and/or byte-oriented mode.*

**Summary**

Section 8.3 *Cryptographic Support* in the TSS of [ST] describes cryprographic services supported by the TOE.

The TSS in section 8.3.4 *Explanation of usage for cryptographic functions* provides table 10 "Explanation of usage for cryptographic functions in the cryptographic modules" which describes the hashing functions specified in FCS_COP.1/HASH. It states that the hashing functions are implemented in byte-oriented mode.

## Guidance Assurance Activities

### Assurance Activity AA-FCS_COP.1-HASH-AGD-01

*The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present.*

### Summary

[CCGUIDE] Table 4 "SFR Configuration Requirements" identifies which SFR can be or needs to be configured in the evaluated configuration. For FCS_COP.1/HASH {MDF}, there is no configuration necessary by the TOE users or administrators for TLS and EAP-TLS connections, however, for VPN connections configuration is possible via a Configuration Profile. Additional guidance is provided in [CCGUIDE] section 5.2.3 *Hashing* which states the following:

- Functions to perform hashing are provided as part of the Apple corecrypto libraries. The invoking function dictates which SHA function is used. Neither the mobile device user nor the mobile device administrator has the ability to configure this choice.
- Similarly, each TLS ciphersuite uses a specific and appropriate SHA function. Neither the mobile device user nor the mobile device administrator has the ability to configure this choice.
- For VPN connections with IKEv2, the integrity algorithm to be used is selectable by the mobile device administrator by setting the IntegrityAlgorithm key in the VPN payload. Note that setting IntegrityAlgorithm to ‘SHA1-96’ is not allowed in the evaluated configuration.

## Test Assurance Activities

### Assurance Activity AA-FCS_COP.1-HASH-ATE-01

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

*The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP. As there are different tests for each mode, an indication is given in the following sections for the bitoriented vs. the byteoriented testmacs.*

- ***Test 1:*** *Short Messages Test: Bit-oriented Mode The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages ranges sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 2:*** *Short Messages Test: Byte-oriented Mode The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 3:*** *Selected Long Messages Test: Bit-oriented Mode The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the $i^{th}$ message is 512 + 99*i, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 4:*** *Selected Long Messages Test: Byte-oriented Mode The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the $i^{th}$ message is 512 + 8*99*i, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

- ***Test 5:*** *Pseudorandomly Generated Messages Test: Byte-oriented Mode This test is for byteoriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of SHAVS. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.*

**Summary**

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.18 Cryptographic Operation (FCS_COP.1/KEYHMAC)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_COP.1-KEYHMAC-ASE-01

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

**Summary**

Section 8.3.4 *Explanation of usage for cryptographic functions* in the TSS of [ST] provides table 10 "Explanation of usage for cryptographic functions in the cryptographic modules". The table entry FCS_COP.1/KEYHMAC specifies the following HMAC functions and associated information:

**Table 5: HMAC Functions**

| HMAC Function | Key Length | Block Size | Output MAC Length |
|---|---|---|---|
| HMAC-SHA-1 | greater than or equal to 112 bits | 512 bits | 160 bits |
| HMAC-SHA-256 | greater than or equal to 112 bits | 512 bits | 256 bits |
| HMAC-SHA-384 | greater than or equal to 112 bits | 1024 bits | 384 bits |
| HMAC-SHA-512 | greater than or equal to 112 bits | 1024 bits | 512 bits |

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_COP.1-KEYHMAC-ATE-01

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

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

**Summary**

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.19 Cryptographic Operation (FCS_COP.1/SIGN)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_COP.1-SIGN-ATE-01

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

- ***Test 1: [conditional] If "ECDSA schemes..." is selected in FCS_COP.1.1/SIGN***
  - ***Test 1.1: ECDSA FIPS 186-4 Signature Generation Test*** *For each supported NIST curve (i.e. P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.*
  - ***Test 1.2: ECDSA FIPS 186-4 Signature Verification Test*** *For each supported NIST curve (i.e. P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.*
- ***Test 2: [conditional] If "RSA schemes..." is selected in FCS_COP.1.1/SIGN***
  - ***Test 2.1: Signature Generation Test*** *The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.*
  - ***Test 2.2: Signature Verification Test*** *The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure. The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.*

**Summary**

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.20 HTTPS Protocol (FCS_HTTPS_EXT.1)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_HTTPS_EXT.1-ATE-01

- ***Test 1:*** *The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS. Other tests are performed in conjunction with testing in the Package for Transport Layer Security. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:*
- ***Test 2:*** *The evaluator shall demonstrate that using a certificate without a valid certification path results in an application notification. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the application is notified of the validation failure.*

#### Summary

Test 1: The evaluator loaded a configuration profile on the TOE containing all the information necessary to connect to the WiFi Access Point (WAP). The evaluator started a TLS web server, connected the TOE to it, and verified using Wireshark that the connection used TLS 1.2 and was successful.

Test 2: The evaluator reproduced the procedure in Test 1, but accessed a different server instead. The evaluator was prompted with the option to accept or deny the certificate.

The evaluator disabled the trust for the certificate in
`Settings -> General -> About -> Certificate Trust Settings`
and re-performed Test 1. The evaluator verified that the warning about the certificate was displayed once again.

Test 1 (repeat): redo of Test 1 except that the CA is not trusted in the keychain. The tester notices that the connection to the web page is interrupted by Safari prompting to either accept or reject the certificate before connecting.

## 2.2.2.21 IPsec (FCS_IPSEC_EXT.1)

### FCS_IPSEC_EXT.1.1

#### TSS Assurance Activities

#### Assurance Activity AA-FCS_IPSEC_EXT.1.1-ASE-01

*In addition to the TSS EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:*

*If the TOE boundary includes a general-purpose operating system or mobile device, the evaluator shall examine the TSS to ensure that it describes whether the VPN client capability is architecturally integrated with the platform itself or whether it is a separate executable that is bundled with the platform.*

**Summary**

Section 8.9.4.2 *IPsec General* in the TSS of [ST] states the following:

- IPsec is implemented in the TOE natively, as part of the TOE OS.
- There is no separate “client” application; the VPN tunnels are configured and controlled by Network Extension Framework, which is a part of the Core OS Layer

**Assurance Activity AA-FCS_IPSEC_EXT.1.1-ASE-02**

*The evaluator shall examine the TSS and determine that it describes how the IPsec capabilities are implemented.*

*If the TOE is a standalone software application, the evaluator shall ensure that the TSS asserts that all IPsec functionality is implemented by the TSF. The evaluator shall also ensure that the TSS identifies what platform functionality the TSF relies upon to support its IPsec implementation, if any (e.g. does it invoke cryptographic primitive functions from the platform’s cryptographic library, enforcement of packet routing decisions by low-level network drivers).*

*If the TOE is part of a general-purpose desktop or mobile operating system, the evaluator shall ensure that the TSS describes at a high level the architectural relationship between the VPN client portion of the TOE and the rest of the TOE (e.g. is the VPN client an integrated part of the OS or is it a standalone executable that is bundled into the OS package). If the SPD is implemented by the underlying platform in this case, then the TSS describes how the client interacts with the platform to establish and populate the SPD, including the identification of the platform's interfaces that are used by the client.*

*In all cases, the evaluator shall also ensure that the TSS describes how the client interacts with the network stack of the platform(s) on which it can run (e.g., does the client insert itself within the stack via kernel mods, does the client simply invoke APIs to gain access to network services).*

*The evaluator shall ensure that the TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how the available rules and actions form the SPD using terms defined in RFC 4301 such as BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301. As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.*

**Summary**

Section 8.9.4.2 *IPsec General* in the TSS of [ST] states the following:

- IPsec is implemented by the TOE natively, as part of the TOE OS. The TOE implements IPsec in accordance to RFC4301. Packets are processed by the TOE in little-endian order. There is no separate client application, the VPN tunnels are configured and controlled by the Network Extension Framework which is part of the TOE OS' Core OS Layer.
- The SPD is implemented by the TOE, which is configured as a managed device using a configuration profile (either manually, through the Apple Configurator 2, or through an MDM). This requires the use of the VPN payload (described in [DEV_MAN] which specifies how a packet is processed against the SPD and includes IPsec Dictionary Keys, IKEv2 Dictionary Keys, DNS Dictionary Keys, Proxies Dictionary Keys, and AlwaysOn Dictionary Keys. The available rules are:
    - PROTECT: allow traffic via tunnel

- BYPASS: allow traffic outside tunnel
- DISCARD: drop traffic

- The TOE enforces an Always-On VPN which requires all traffic entering and leaving the TOE platform interfaces to be protected via an IPsec VPN connection. The only exceptions are Captive Networking apps, Voice Mail, Cellular Services, and AirPrint.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.1-AGD-01

*In addition to the Operational Guidance EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:*

*If the configuration of the IPsec behavior is from an environmental source, most notably a VPN gateway (e.g through receipt of required connection parameters from a VPN gateway), the evaluator shall ensure that the operational guidance contains any appropriate information for ensuring that this configuration can be properly applied.*

*Note in this case that the implementation of the IPsec protocol must be enforced entirely within the TOE boundary; i.e. it is not permissible for a software application TOE to be a graphical front-end for IPsec functionality implemented totally or in part by the underlying OS platform. The behavior referenced here is for the possibility that the configuration of the IPsec connection is initiated from outside the TOE, which is permissible so long as the TSF is solely responsible for enforcing the configured behavior. However, it is allowable for the TSF to rely on low-level platform-provided networking functions to implement the SPD from the client (e.g., enforcement of packet routing decisions).*

### Summary

According to section 8.9.4.2 *IPsec General* of [ST] , IPsec is implemented in the TOE natively, as part of the TOE OS. There is no seperate "client" application; the VPN tunnels are configured and controlled by Network Extension Framework, which is part of the Core OS Layer of the TOE. The TOE enforces an "always on" configuratin meaning that all traffic entering and leaving the TOE platform interfaces is protected via an IPsec VPN connection. The Security Policy Database (SPD) is implemented by the TOE, which as a managed device is configured using a Configuration Profile tiehter manually or through an Apple Configurator 2, or via an MDM solution.

[CCGUIDE] section 5.3.3 *IPsec Configuration* provides related guidance for IPsec which will be further assessed in the next few work units.

### Assurance Activity AA-FCS_IPSEC_EXT.1.1-AGD-02

*The evaluator shall examine the operational guidance to verify it describes how the SPD is created and configured. If there is an administrative interface to the client, then the guidance describes how the administrator specifies rules for processing a packet. The description includes all three cases - a rule that ensures packets are encrypted/decrypted, dropped, and allowing a packet to flow in plaintext. The evaluator shall determine that the description in the operational guidance is consistent with the description in the TSS, and that the level of detail in the operational guidance is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.*

*If the client is configured by an external application, such as the VPN gateway, then the operational guidance should indicate this and provide a description of how the client is configured by the external application. The description should contain information as to how the SPD is established and set up in an unambiguous fashion. The description should also include what is configurable via the external application, how ordering of entries may be expressed, as well as the impacts that ordering of entries may have on the packet processing.*

*In either case, the evaluator ensures the description provided In the TSS is consistent with the capabilities and description provided in the operational guidance.*

### Summary

[CCGUIDE] section 5.3.3 *IPsec Configuration* provides related guidance for IPsec. It states the following:

- The mobile devices implement IPsec natively, as part of their operating system, so any processing of packets used in IPsec communication takes place on the mobile device. IPsec VPN tunnels are configured and controlled by the Network Extension Framework, which is a part of the Core OS Layer of the mobile devices' operating system.
- The Security Policy Database (SPD) is created and configured by defining exceptions for IP traffic routing in a Configuration Profile. By default, all IP traffic is sent through a protected channel between the devices and the desired endpoint (PROTECT in the SPD). Any deviations from the default routing behavior must be explicitly specified as exceptions in the Configuration Profile, using the Wi-Fi Payload.
- Packet processing exceptions can be created for applications which make use of Captive Networking Identifiers (Captive Networking Apps), as well as for VoiceMail, AirPrint, and CellularServices. The mobile device administrator will need to refer to their organization's security policies to determine whether exceptions should be created and how those exceptions should be configured.
- Exceptions for Captive Networking Apps can be configured in the Wi-Fi Payload to allow traffic for these apps to pass outside the tunnel (BYPASS in the SPD). Exceptions for voicemail, AirPrint, and CellularServices can allow traffic to pass unencrypted outside the tunnel (BYPASS in the SPD) or drop the traffic entirely (DISCARD in the SPD).
- When the VPN is configured as Always-On, the mobile device uses IKEv2 for security association (SA) establishment. Since the mobile device must be configured with Always-On VPN in order to be in the evaluated configuration, the use of IKEv2 does not need to be configured separately.
- To configure exceptions for Voicemail, AirPrint, and CellularServices, the mobile device administrator can specify a ServiceExceptions array in the AlwaysOn dictionary of the VPN payload. Each entry in a ServiceExceptions array lists a ServiceName key and a corresponding Action key. The allowed values for ServiceName and Action can be found in Table 12: Essential Keys for the VPN Payload. For each ServiceName, the corresponding Action can be set to 'Allow' (BYPASS in the SPD) or 'Drop' (DISCARD in the SPD).
- Likewise, exceptions for Captive Network Apps can be defined by using *AllowCaptiveWebSheet* , *AllowAllCaptiveNetworkPlugins* , and *AllowedCaptiveNetworkPlugins* keys in the Configuration Profile as described in [DEV_MAN] .

The evaluator found the provided guidance to be consistent with the description in the TSS.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.1-ATE-01

*As a prerequisite for performing the Test EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall do the following:*

*The evaluator shall minimally create a test environment equivalent to the test environment illustrated below. It is expected that the traffic generator is used to construct network packets and will provide the evaluator with the ability manipulate fields in the ICMP, IPv4, IPv6, UDP, and TCP packet headers. The evaluator shall provide justification for any differences in the test environment.*

*Note that the evaluator shall perform all tests using the VPN client and a representative sample of platforms listed in the ST (for TOEs that claim to support multiple platforms).*

### Summary

IPsec testing is performed using a wireless network, therefore all test equipment and TOE platforms are on the same network as depicted in the illustration in the [MOD_VPNC_V2.3] Supporting Document.

### Assurance Activity AA-FCS_IPSEC_EXT.1.1-ATE-02

*Depending on the implementation, the evaluator may be required to use a VPN gateway or some form of application to configure the client. For Test 2, the evaluator is required to choose an application that allows for the configuration of the full set of capabilities of the VPN client (in conjunction with the platform). For example, if the client provides a robust interface that allows for specification of wildcards, subnets, etc., it is unacceptable for the evaluator to choose a VPN Gateway that only allows for specifying a single fully qualified IP addresses in the rule.*

*The evaluator shall perform the following tests:*

- ***Test 1:** : The evaluator shall configure an SPD on the client that is capable of the following: dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the client with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule. The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed through without modification, was encrypted by the IPsec implementation.*
- ***Test 2:** The evaluator shall devise several tests that cover a variety of scenarios for packet processing. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and operational guidance. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.*

#### Summary

The evaluator implemented an automatic test for both tests in this assurance activity. The program will perform the activities specified in the assurance activity and the traffic will be sniffed using a traffic analyzer. The result shows the Always-On-VPN configuration with a passing and a failing connection attempt. The test result is obtained by performing the test defined for FDP_IFC_EXT.1 Test 1. With the PCAP logs, the passing connection attempt shows that all traffic is encapsulated into ESP. The failing test result shows that if the IPsec server is not reachable, no traffic is generated. The test also demonstrates that if the remote end sends non-encrypted traffic that should have been transmitted encrypted, the TOE discards the remote traffic.

The testing conducted for Test 1 covers all different use cases defined for the Always-On-VPN:

- Passing traffic encrypted through the tunnel,
- Preventing traffic generated from TOE to a remote entity if the VPN is not set up,
- Dropping traffic generated from a remote entity that is not encrypted,
- Enabling and disabling all defined exceptions for traffic to bypass the VPN tunnel Testing failing authentication during the VPN handshake.

## FCS_IPSEC_EXT.1.2

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.2-ASE-01

*The evaluator shall check the TSS to ensure it states that the VPN can be established to operate in tunnel mode and/or transport mode (as selected).*

#### Summary

Section 8.9.4 *IPsec Characteristics* in the TSS of [ST] states that the TOE operates IPsec in tunnel mode only, which is consistent with the selection in FCS_IPSEC_EXT.1.2. The evaluator to examine operational guidance for instructions on how to configure the IPsec connection in tunnel mode.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.2-AGD-01

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

*If both transport mode and tunnel mode are implemented, the evaluator shall review the operational guidance to determine how the use of a given mode is specified.*

#### Summary

Per [ST] , the TOE only supports tunnel mode. Related guidance for configuring tunnel mode is provided in [CCGUIDE] section 5.3.5 *VPN Configuration* and section 5.3.6 *Keys for Configuration Network Protocol* . The configuration involves using a Configuration Profile with the VPN Payload.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.2-ATE-01

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

- ***Test 1:*** *[conditional]: If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in tunnel mode and also configures a VPN gateway to operate in tunnel mode. The evaluator configures the TOE and the VPN gateway to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the client to connect to the VPN GW peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.*
- ***Test 2:*** *[conditional]: If transport mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in transport mode and also configures an IPsec peer to accept IPsec connections using transport mode. The evaluator configures the TOE and the endpoint device to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the remote endpoint. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.*
- ***Test 3:*** *[conditional]: If both tunnel mode and transport mode are selected, the evaluator shall perform both Test 1 and Test 2 above, demonstrating that the TOE can be configured to support both modes.*
- ***Test 4:*** *[conditional]: If both tunnel mode and transport mode are selected, the evaluator shall modify the testing for FCS_IPSEC_EXT.1 to include the supported mode for SPD PROTECT entries to show that they only apply to traffic that is transmitted or received using the indicated mode.*

#### Summary

Test 1: The test suite developed covers multiple test aspects. Please see FCS_IPSEC_EXT.1.4 Test 1, which demonstrates that the TOE connects through the VPN with all supported symmetric key cryptographic algorithms and key sizes. Please see FCS_IPSEC_EXT.1.11 Test 2, which demonstrates that the TOE connect through the VPN with all supported asymmetric key cryptographic algorithms and EC curves.

Test 2-4 are not applicable because [ST] specifies tunnel mode only.

## FCS_IPSEC_EXT.1.3

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.3-ASE-01

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

**Summary**

Section 8.9.4.2 *IPsec General* in the TSS of [ST] states the following:

The TOE enforces an "always on" configuration meaning that all traffic entering and leaving the TOE platform interfaces is protected via an IPsec VPN connection. The TOE allows a limited number of services to be configured to either not allow (DISCARD) or be sent plaintext (BYPASS). These services include applications that make use of Captive Networking Identifiers, Voice Mail, Cellular Services and AirPrint. All other communications are always sent through the IPsec tunnel (PROTECT within the Security Policy Database (SPD)). In order to set a service to match a PROTECT rule in the SPD, select "Allow traffic via tunnel." "Drop Traffic" will cause that traffic to match a DISCARD rule. "Allow traffic outside tunnel" will create a BYPASS rule for that service.

Any other plaintext data that is received is ignored (discarded). Discarding happens automatically without the need to configure an explicit discard.

**Guidance Assurance Activities**

**Assurance Activity AA-FCS_IPSEC_EXT.1.3-AGD-01**

*The evaluator shall check that the operational guidance provides instructions on how to construct or acquire the SPD and uses the guidance to configure the TOE for the following test.*

**Summary**

[CCGUIDE] section 5.3.3 *IPsec Configuration* provides related guidance for IPsec. It states that the SPD is created and configured by defining exceptions for IP traffic routing in a Configuration Profile using the VPN Payload and related keys and key values described in table 12 "Essential Keys for the VPN Payload" of [CCGUIDE] . Additionally, the evaluator would be using the provided guidance to perform independent testing.

**Test Assurance Activities**

**Assurance Activity AA-FCS_IPSEC_EXT.1.3-ATE-01**

*The evaluator shall perform the following test:*

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

**Summary**

Test 1: The evaluator established a VPN in tunnel mode to a remote VPN peer via WiFi or LTE and installed a Captive Portal bypass setting. Using the packet sniffer logs, the evaluator was able to clearly identify the captive portal bypass for both WiFi and LTE.

## FCS_IPSEC_EXT.1.4

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.4-ASE-01

*The evaluator shall examine the TSS to verify that the algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the ST author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA- based HMAC algorithm conforms to the algorithms specified in the relevant iteration of FCS_COP.1 from the Base-PP that applies to keyed-hash message authentication.*

### Summary

Section 8.9.4.3 *IPsec Characteristics* in the TSS of [ST] describes the IPsec characteristics enforced by the TOE in the the evaluated configuration. It specifies the following:

- Symmetric algorithms for IKE and ESP encryption: AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256.
- Integrity mechanisms: HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512.

The evaluator found the above information to be consistent with FCS_IPSEC_EXT.1.4 as well as FCS_COP.1(4).

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.4-AGD-01

*The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.*

### Summary

Per [CCGUIDE] section 5.3.5 *VPN Configuration* , the administrator may use the VPN Payload in the Configuration Profile to configure a traditional system-wide VPN based on IPsec. Table 12 "Essential Keys for the VPN Payload" of [CCGUIDE] describes the related keys and key values for the VPN Payload.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.4-ATE-01

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

### Summary

Test 1: The evaluator established a VPN in tunnel mode to a remote VPN peer via WiFi and LTE. The connection is established with AES-128-CBC, AES-256-CBC, AES-128-GCM, and AES-256-GCM.

## FCS_IPSEC_EXT.1.5

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.5-ASE-01

*The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. If IKEv1 is implemented, the evaluator shall verify that the TSS indicates whether or not XAUTH is supported, and that aggressive mode is not used for IKEv1 Phase 1 exchanges (i.e. only main mode is used). It may be that these are configurable options.*

### Summary

Section 8.9.4.3 *IPsec Characteristics* in the TSS of [ST] specifies that the TOE implements IKEv2 only which is consistent with FCS_IPSEC_EXT.1.5.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.5-AGD-01

*The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the test below. If XAUTH is implemented, the evaluator shall verify that the operational guidance provides instructions on how it is enabled or disabled.*

*If the TOE supports IKEv1, the evaluator shall verify that the operational guidance either asserts that only main mode is used for Phase 1 exchanges, or provides instructions for disabling aggressive mode.*

### Summary

Per [ST] , the TOE only supports IKEv2 in the evaluated configuration. Also, the TOE does not support XAUTH.

[CCGUIDE] section 5.3.3 *IPsec Configuration* contains the following statement:

> *When the VPN is configured as Always-On, the mobile device uses IKEv2 for security association (SA) establishment. Since the mobile device must be configured with Always-On VPN in order to be in the evaluated configuration, the use of IKEv2 does not need to be configured separately.*

In other words, IKEv2 is set when configuring Always-On as per instructions in section 5.3.5 *VPN Configuration* of [CCGUIDE] . This involves using the VPN Payload in a Configuration Profile and setting the *VPNType key* to 'AlwaysOn' and *ProtocolType* key to 'IKEv2'.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.5-ATE-01

- ***Test 1:*** *The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 7296, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed. If the TOE supports IKEv1 with or without XAUTH, the evaluator shall verify that this test can be successfully repeated with XAUTH enabled and disabled in the manner specified by the operational guidance. If the TOE only supports IKEv1 with XAUTH, the evaluator shall verify that connections not using XAUTH are unsuccessful. If the TOE only supports IKEv1 without XAUTH, the evaluator shall verify that connections using XAUTH are unsuccessful.*
- ***Test 2:*** *[conditional]: If the TOE supports IKEv1, the evaluator shall perform any applicable operational guidance steps to disable the use of aggressive mode and then attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator shall show that the TOE will reject a VPN gateway from initiating an IKEv1 Phase 1 connection in aggressive mode. The evaluator should then show that main mode exchanges are supported.*

**Summary**

Test 1: The evaluator established a VPN in tunnel mode to a remote VPN peer via WiFi and LTE using NAT.

Test 2 is not applicable because [ST] only specifies IKEv2.

## FCS_IPSEC_EXT.1.6

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.6-ASE-01

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

**Summary**

Section 8.9.4.3 *IPsec Characteristics* in the TSS of [ST] specifies that the algorithms used for encrypting IKE payload are AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256, which is found to be consistent with FCS_IPSEC_EXT.1.6.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.6-AGD-01

*The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.*

**Summary**

Per [ST] , the TOE supports AES-CBC-128, AES-CBC-256, AES-GCM-128, and AES-GCM-256 for payload encryption.

[CCGUIDE] Table 12 "Essential Keys for the VPN Payload" specifies the algorithms for the *EncryptionAlgorithm* key to be either AES-128, AES-256, AES-128-GCM, or AES-256-GCM. This section also contains a note explaining that AES-128 and AES-256 use the CBC mode.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.6-ATE-01

*The evaluator shall use the operational guidance to configure the TOE (or to configure the Operational Environment to have the TOE receive configuration) to perform the following test for each ciphersuite selected:*

- ***Test 1:*** *The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation. The evaluator will confirm that the connection is successful by confirming that data can be passed through the connection once it is established. For example, the evaluator may connect to a webpage on the remote network and verify that it can be reached.*

**Summary**

Test 1: The evaluator established a VPN in tunnel mode to a remote VPN peer via WiFi and LTE. The connection is established with AES-128-CBC, AES-256-CBC, AES-128-GCM, and AES-256-GCM. The list of ciphers covers all ciphers selected in [ST].

## FCS_IPSEC_EXT.1.7

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.7-ASE-01

*There are no TSS EAs for this requirement.*

### Summary

There are no TSS EAs for this requirement.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.7-AGD-01

*The evaluator shall check the operational guidance to ensure it provides instructions on how the TOE configures the values for SA lifetimes. In addition, the evaluator shall check that the guidance has the option for either the Administrator or VPN Gateway to configure Phase 1 SAs if time-based limits are supported. Currently there are no values mandated for the number of packets or number of bytes, the evaluator shall simply check the operational guidance to ensure that this can be configured if selected in the requirement.*

### Summary

Per [ST] , the TOE ensures that IKEv2 SA lifetimes can be configured by an administrator based on length of time. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.

[CCGUIDE] Table 12 "Essential Keys for the VPN Payload" describes the relevant keys and key values to set in the VPN Payload in a Configuration Profile.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.7-ATE-01

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

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

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

- ***Test 1:*** *[conditional]: The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.*

- ***Test 2:*** *[conditional]: The evaluator shall construct a test where a Phase 1 SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.*
- ***Test 3:*** *[conditional]: The evaluator shall perform a test similar to Test 2 for Phase 2 SAs, except that the lifetime will be 8 hours or less instead of 24 hours or less.*
- ***Test 4:*** *[conditional]: If a fixed limit for IKEv1 SAs is supported, the evaluator shall establish an SA and observe that the connection is closed after the fixed traffic and/or time value is reached.*

**Summary**

The rekey time is not configurable, all devices rekey in slightly over 6 hours.

Test 1 is not applicable because [ST] specifies time-based maximum lifetime only.

Test 2: The test suite establishes a VPN communication channel with a lifetime longer than 24 hours. The VPN is established via WiFi and LTE. The evaluator found the connection is rekeyed before 7 hours passed.

Test 3: The test suite establishes a VPN communication channel with a lifetime longer than 8 hours. The VPN is established via WiFi and LTE. The evaluator found the connection is rekeyed before 7 hours passed.

Test 4 is not applicable because [ST] specifies IKEv2 only.

## FCS_IPSEC_EXT.1.8

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.8-ASE-01

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

**Summary**

Section 8.9.4.3 *IPsec Characteristics* in the TSS of [ST] lists the following DH groups supported by the TOE:

- 14 (2048-bit MODP),
- 15 (3072-bit MODP),
- 19 (256-bit Random ECP), and
- 20 (384-bit Random ECP).

Section 8.9.4.2 *IPsec General* states that configuration of VPN connection setting, such as, authentication method and algorithm selection, is performed by the IPsec VPN client administrator. This implies that DH groups are negotiated based on the configuration chosen by such administrator.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.8-AGD-01

*There are no AGD EAs for this requirement.*

### Summary

There are no AGD EAs for this requirement.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.8-ATE-01

*The evaluator shall perform the following test:*

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

### Summary

Test 1: The evaluator established a VPN in tunnel mode to a remote VPN peer via WiFi. The connection is established with the Diffie-Hellman groups of 14, 15, 19, 20 compliant with [ST].

## FCS_IPSEC_EXT.1.9

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.9-ASE-01

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

### Summary

Section 8.9.4.5 *IKE* in the TSS of [ST] states that TOE generates the secret value ‘x’ and nonces used in the IKEv2 Diffie-Hellman key exchanges using the TOE platform CAVP validated DRBG specified (as specified in FCS_RBG_EXT.1). T he possible lengths of ‘x’ and the nonces are 224, 256, or 384 bits.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.9-AGD-01

*There are no AGD EAs for this requirement.*

### Summary

There are no AGD EAs for this requirement.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.9-ATE-01

*There are no test EAs for this requirement.*

### Summary

There are no test EAs for this requirement.

## FCS_IPSEC_EXT.1.11

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.11-ASE-01

*The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication.*

*If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished depending on whether the TSF can generate a pre-shared key, accept a pre-shared key, or both.*

*The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN) and, if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate.*

#### Summary

Section 8.9.4.4 *Peer authentication* in the TSS of [ST] describes how the TOE performs peer authentication for an IPsec connection. It states the following:

- The TOE supports the use of RSA or ECDSA X.509 certificates for peer authentication.
- During the authentication process, a comparison is made of the Subject Alternative Name (SAN) contained within the peer certificate to the Fully Qualified Domain Name (FQDN) of the requested server. The Common Name (CN) is ignored. If the FQDN in the certificate does not match the expected SAN for the peer, then the application cannot establish the connection.
- If the SAN in the peer certificate does not match that of the peer's identifier or a SAN does not exist in the peer certificate, the authentication process fails. If the SAN matches the peer's identifier, the authentication process is successful.

The evaluator noted that the ST does not select "pre-shared keys" in FCS_IPSEC_EXT.1.11.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.11-AGD-01

*The evaluator shall check that the operational guidance describes how pre-shared keys are to be generated and established.*

*The evaluator ensures the operational guidance describes how to set up the TOE to use the cryptographic algorithms RSA and/or ECDSA.*

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

*The evaluator shall also ensure that the operational guidance includes the configuration of the reference identifier(s) for the peer.*

#### Summary

[CCGUIDE] section 5.3.5 *VPN Configuration* provides related guidance on VPN configuration. The configuration is done (by an administrator) via the Configuration Profile using the VPN Payload. table 12 "Essential Keys for the VPN Payload" of [CCGUIDE] describes the keys that must be specified for the payload. Per table 12, the *CertificateType* key must be set to RSA (default), ECDSA

P-256, or ECDSA P-384. Also, the certificate to be used for authentication is specified via the *PayloadCertificateUUID* key. Additionally, reference identifier(s) must be set via the *LocalIdentifier* and *RemoteIdentifier* keys.

### Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.11-ATE-01

*For efficiency's sake, the testing that is performed here has been combined with the testing for **FIA_X509_EXT.2** and **FIA_X509_EXT.3** (for IPsec connections and depending on the Base-PP), **FCS_IPSEC_EXT.1.12**, and **FCS_IPSEC_EXT.1.13**. The following tests shall be repeated for each peer authentication protocol selected in the **FCS_IPSEC_EXT.1.11** selection above:*

- ***Test 1:** The evaluator shall have the TOE generate a public-private key pair, and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE and the peer VPN used to establish a connection) for its signature. The values for the DN (Common Name, Organization, Organizational Unit, and Country) will also be passed in the request. Alternatively, the evaluator may import to the TOE a previously generated private key and corresponding certificate.*
- ***Test 2:** The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.*
- ***Test 3:** : The evaluator shall test that the TOE can properly handle revoked certificates – conditional on whether CRL or OCSP is selected; if both are selected, and then a test is performed for each method. For this current version of the PP-Module, the evaluator has to only test one up in the trust chain (future drafts may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the SA is established. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the TOE will not establish an SA*
- ***Test 4:** [conditional]: The evaluator shall generate a pre-shared key and use it, as indicated in the operational guidance, to establish an IPsec connection with the VPN GW peer. If the generation of the pre-shared key is supported, the evaluator shall ensure that establishment of the key is carried out for an instance of the TOE generating the key as well as an instance of the TOE merely taking in and using the key.*

*For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:*

- ***Test 5:** For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds.*
- ***Test 6:** For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKE authentication fails.*

*The following tests are conditional:*

- ***Test 7:** [conditional]: If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented certificate but to match the Common Name in the peer's presented certificate, and verify that the IKE authentication fails.*
- ***Test 8:** [conditional]: If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKE authentication fails. **To demonstrate a comparison of DN values, the evaluator shall change any one of the four DN values and verify that the IKE authentication fails.***
- ***Test 9:** [conditional]: If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator must repeat test 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKE authentication fails.*
- ***Test 10:** [conditional]: If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKE peer.*

**Summary**

Test 1: The TOE only supports the import of previously generated private key and corresponding certificates for VPN (although it is capable of generating a public-private key pair, it does not provide interfaces to generate a CSR and forward it to a CA). Thus, all previously conducted successful VPN connections require the import of a public certificate and the associated private key into the TOE when configuring a VPN. In particular, the test steps outlined for FDP_IFC_EXT.1 {MDF} explain how to import a certificate and its key into the TOE using a configuration profile along with the VPN configuration details. All VPN configurations used for all tests import a configuration profile containing the certificate and key pair to establish a successful authentication and thus a successful VPN connection. To support that statement, all configuration profiles used for the VPN testing are provided which contain the certificate/key data as PKCS #12 data blob embedded in these configuration profiles.

Test 2: The evaluator established a VPN in tunnel mode to a remote VPN peer via WiFi and LTE. The connection is established with the certificate types of RSA and ECDSA compliant with specified in [ST]. The authentication with all certificate types was successful using the remote ASA VPN server which demonstrates the correct verification of the DN.

Test 3: The evaluator “successfully” failed to establish a VPN in tunnel mode to a remote VPN peer via WiFi and LTE. The connection attempt is performed with the certificate types of RSA and ECDSA compliant with specified in [ST]. The authentication failure is triggered with CRLs. The testing is repeated with the new certificates of the same types that are not revoked including certificates that are subjected to OCSP.

Test 4 is not applicable because it is removed by TD0379.

Test 5 is not applicable because [ST] specifies certificate-based authentication only.

Test 6: As outlined in [ST], the CN/DN must match the remote peer’s FQDN. All certificates used in the preceding tests covering successful VPN connections utilize a FQDN.

Test 7: The evaluator “successfully” failed to establish a VPN in tunnel mode to a remote VPN peer via WiFi and LTE. The connection attempt is performed with the certificate types of RSA and ECDSA compliant with specified in [ST]. The authentication with all certificate types fails due to an incorrect DN.

Test 8 is not applicable because the TOE no longer supports CN.

Test 9 is not applicable because [ST] specifies support for FQDN as part of the CN/DN only.

Test 10 is not applicable because IP address identifier types are not supported by the TOE.

Test 11 is not applicable because the TOE does not support ID payload.

## FCS_IPSEC_EXT.1.14

### TSS Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.14-ASE-01

*The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.*

### Summary

Section 8.9.4.5 *IKE* in the TSS of [ST] states the following.

The strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the IKEv2/IKE_SA connection and the strength of the symmetric algorithm negotiated to protect the IKEv2 CHILD_SA connection is configured using .xml configuration files. The administrator must explicitly choose the cryptographic algorithms (including key strength) used for each SA. Key strength must be one of 128 or 256 bits as specified in the IKEv2 Dictionary Keys, EncryptionAlgorithm Key.

The possible lengths of 'x' and the nonces are 224, 256, or 384 bits. In the evaluated configuration, during negotiation the TOE will only negotiate the configured algorithms which must include an IKEv2/IKE_SA at least that of IKEv2 CHILD_SA. This configuration is specified in the [CCGUIDE]

## Guidance Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.14-AGD-01

*There are no AGD EAs for this requirement.*

### Summary

There are no AGD EAs for this requirement.

## Test Assurance Activities

### Assurance Activity AA-FCS_IPSEC_EXT.1.14-ATE-01

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

- ***Test 1:*** *This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.*
- ***Test 2:*** *[conditional]: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.*
- ***Test 3:*** *This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.*
- ***Test 4:*** *This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.*

### Summary

Tests 1, 3, 4: The evaluator established a VPN in tunnel mode to a remote VPN peer via WiFi and LTE. The connection is successfully established by using all of the following different cipher types:

- Symmetric ciphers: AES-128-CBC, AES-256-CBC, AES-128-GCM, AES-256-GCM
- Hashes: SHA-1, SHA-256, SHA-384, SHA-512
- IKEv2
- All mentioned cipher combinations apply equally to the ISAKMP SA and the IPSEC SA.

Test 2 is not applicable because the TOE only allows specifications of ciphers once for a given VPN configuration.

## 2.2.2.22 Initialization Vector Generation (FCS_IV_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_IV_EXT.1-ASE-01

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

#### Summary

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes key management. Per this section, 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 SEP (i.e., FCS_RBG_EXT.1(SEP)). Files are encrypted using this key with AES in Xor-Encrypt-Xor-based tweaked-codebook mode with ciphertext stealing (XTS) where the initialization vector (IV) is calculated with the block offset into the file, encrypted with the SHA-1 hash of the per-file key, and follows [SP800-38E]. On Apple A14 and later devices and M1 and later devices, the encryption uses AES-256 in XTS mode in the SEP. On Apple A9 through A13 devices, the encryption uses AES-128 in XTS mode in the SEP where the 256-bit per file key is split to provide a 128-bit tweak and a 128-bit cipher key.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.23 Random Bit Generation (FCS_RBG_EXT.1)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

#### Assurance Activity AA-FCS_RBG_EXT.1-AGD-01

*The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.*

#### Summary

[CCGUIDE] Table 4 "SFR Configuration Requirements" identifies which SFR can be or needs to be configured in the evaluated configuration. Table entry FCS_RBG_EXT.1 {MDF} states that no configuration needed as generation of random numbers is hard coded. It also references section 5.2.4 for more details which states the following:

> " *For the evaluated configuration, no configuration is required from the mobile device user.* "

and

> " *For the evaluated configuration, no configuration is required from the mobile device administrator.* "

### Test Assurance Activities

#### Assurance Activity AA-FCS_RBG_EXT.1-ATE-01

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

*The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.*

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

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

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

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

#### Summary

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.24 Cryptographic Algorithm Services (FCS_SRV_EXT.1)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_SRV_EXT.1-ATE-01

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

**Summary**

The evaluator confirmed that CAVS testing of all cryptographic operations was performed in accordance with the CAVP test procedures. The results have been validated by the CAVP and the certificate information is provided in section 2.1, "CAVP Summary," in the Assurance Activity Report (AAR).

## 2.2.2.25 Cryptographic Key Storage (FCS_STG_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_STG_EXT.1-ASE-01

*The evaluator shall review the TSS to determine that the TOE implements the required secure key storage. The evaluator shall ensure that the TSS contains a description of the key storage mechanism that justifies the selection of "mutable hardware" or "software-based".*

**Summary**

The ST selects "software-based" in FCS_STG_EXT.1 as a key storage mechanism for asymmetric keys, symmetric keys, and persistent secrets.

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes how the TOE peforms key managemnet. This section describes how keys and data are stored and encrypted (wrapped) on the TOE OS, and describes the different key classes. The evaluator reviewed section 8.3.1 and also figure 5 "Key Hierarchy in the TOE OS" in section 8.3.2 and determined that the TOE implements the required secure key storage.

### Guidance Assurance Activities

#### Assurance Activity AA-FCS_STG_EXT.1-AGD-01

*The evaluator shall review the AGD guidance to determine that it describes the steps needed to import or destroy keys/secrets.*

**Summary**

[CCGUIDE] section 5.2.5 *Keys/Secrets Import/Destruction* provides related guidance for managing keys and secrets. It states that cryptographic keys are stored in keychains and an application only has access to its own key chain items, so access restrictions are automatically enforced. No configuration is required by the TOE users or administrators.

### Test Assurance Activities

#### Assurance Activity AA-FCS_STG_EXT.1-ATE-01

*The evaluator shall test the functionality of each security function:*

- ***Test 1:*** *The evaluator shall import keys/secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that generates a key/secret of each supported type and calls the import functions. The evaluator shall verify that no errors occur during import.*
- ***Test 2:*** *The evaluator shall write, or the developer shall provide access to, an application that uses an imported key/secret:*
  - *For RSA, the secret shall be used to sign data.*
  - *For ECDSA, the secret shall be used to sign data*

  *In the future additional types will be required to be tested:*
  - *For symmetric algorithms, the secret shall be used to encrypt data.*
  - *For persistent secrets, the secret shall be compared to the imported secret.*

  *The evaluator shall repeat this test with the application-imported keys/secrets and a different application's imported keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to use the key/secret imported by the user or by a different application:*
  - *The evaluator shall deny the approvals to verify that the application is not able to use the key/secret as described.*
  - *The evaluator shall repeat the test, allowing the approvals to verify that the application is able to use the key/secret as described.*

  *If the ST author has selected "common application developer", this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.*
- ***Test 3:*** *The evaluator shall destroy keys/secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that destroys an imported key/secret. The evaluator shall repeat this test with the application-imported keys/secrets and a different application's imported keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to destroy the key/secret imported by the administrator or by a different application:*
  - *The evaluator shall deny the approvals and verify that the application is still able to use the key/secret as described.*
  - *The evaluator shall repeat the test, allowing the approvals and verifying that the application is no longer able to use the key/secret as described.*

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

**Summary**

Test 1: The evaluator generated RSA and ECDSA keys using OpenSSL (external to the TOE), loaded them into the TOE's keychain, and verified that their use succeeded. The evaluator also used the CryptoTest application to locally generate keys, store them and fetch in/from the TOE.

Test 2: The evaluator installed .p12 key bundles on the TOE, started the test WAP, and connected the TOE to the WLAN network. The evaluator then accessed the test web server and selected the RSA certificate when prompted by the TOE. The evaluator verified that the connection was successful. The evaluator then cleared the browser cache and data and re-performed the operation with the ECDSA certificate. The evaluator verified that once gain the connection was successful. The evaluator re-performed the operations and verified that if the evaluator declines selecting one of the certificates, the connection fails.

Test 3: The evaluator deleted the configuration profile containing the keys and verified that the keys are only deleted when removal of the profile is authorized by the user of the TOE.

## 2.2.2.26 Encrypted Cryptographic Key Storage (FCS_STG_EXT.2)

### FCS_STG_EXT.2.1

#### TSS Assurance Activities

#### Assurance Activity AA-FCS_STG_EXT.2.1-ASE-01

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

*The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.*

**Summary**

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes key management. Additionally, figure 5 "Key Hierarchy in the TOE OS" in section 8.3.2 *Storage of Persistent Secrets and Private Keys by the Agent* describes the key hierarchy and key materials used by the TOE. This description specifies the keys as 256-bit AES keys and explains how a new per-file key is generated, as well as how keys are wrapped and protected by the TOE OS. The IV is determined on the memory location of the file (the IV is calculated with the block offset into the file).

The evaluator determined that the description is sufficiently detailed, the key management hiearchy both in textual description as well as illustrated in figure 5 is clear and meets the requirements to ensure the keys are adequately protected. Also, the description expands in multiple paragraphs and includes a flow-chart diagram in figure 5 which clearly depicts the key hierarchy in the TOE OS.

**Guidance Assurance Activities**

No assurance activities defined.

**Test Assurance Activities**

No assurance activities defined.

## FCS_STG_EXT.2.2

**TSS Assurance Activities**

**Assurance Activity AA-FCS_STG_EXT.2.2-ASE-01**

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

**Summary**

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes key management. Per this section, DEK is encrypted using AES key wrapping in accordance to RFC 3394 (AES Key Wrap) as defined in FCS_STG_EXT.2.2.

**Guidance Assurance Activities**

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.27 Integrity of Encrypted Key Storage (FCS_STG_EXT.3)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_STG_EXT.3-ASE-01

> *The evaluator shall examine the key hierarchy description in the TSS section to verify that each encrypted key is integrity protected according to one of the options in FCS_STG_EXT.3.*
>
> *The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.*

#### Summary

Per FCS_STG_EXT.3, the following ecrypted keys are integrity protected: DEKs, KEKs, and long-term trusted channel key material . Per section 8.3.1 *Overview of Key Management* in the TSS of [ST] the integrity mechanism used for each encrypted key is the one provided by the Key Wrap cipher mode defined in RFC 3394 and NIST SP 800-38F. This mode includes authentication as described in section 6.2 of SP 800-38F.

The evaluator determined that the description is sufficiently detailed, the key management hiearchy both in text description as well as illustrated in figure 5 is clear and meets the requirements to ensure the keys are adequately protected. Also, the description expands in multiple paragraphs and includes a flow-chart diagram in figure 5 which clearly depicts the key hierarchy in the TOE OS.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.28 Cryptographic Key Storage (FCS_STG_EXT.4)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_STG_EXT.4-MDMA-ASE-01

> *The evaluator will verify that the TSS lists each persistent secret (credential, secret key) and private key needed to meet the requirements in the ST. For each of these items, the evaluator will confirm that the TSS lists for what purpose it is used, and, for each platform listed as supported in the ST, how it is stored. The evaluator shall verify that the Agent calls a platform-provided API to store persistent secrets and private keys.*

#### Summary

Section 8.3.2 *Storage of Persistent Secrets and Private Keys by the MDM Agent* in the TSS of [ST] provides table 9 "Summary of keys and persistent secrets used by the MDM Agent" outlining the keys and persistent secrets stored for the agent. The table is duplicated below:

**Table 6: Summary of keys and persistent secrets used by the MDM Agent**

| 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 must be unique for each installation of your product | Stored in wrapped form in persistent storage |

This section also states that the MD Agent calls the TOE OS API on the device in order to store keys and persistent secrets in the Keychain.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.29 TLS Protocol (FCS_TLS_EXT.1)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

#### Assurance Activity AA-FCS_TLS_EXT.1-AGD-01

*The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.*

#### Summary

For FCS_TLS_EXT.1, the ST selects "TLS as a client". The ST includes FCS_TLSC_EXT.1, FCS_TLSC_EXT.2, FCS_TLSC_EXT.4, and FCS_TLSC_EXT.5 as dependent components. This is consistent with the selection in FCS_TLS_EXT.1 as these SFRs support a TLS client.

### Test Assurance Activities

No assurance activities defined.

## 2.2.2.30 TLS Client Protocol (FCS_TLSC_EXT.1)

### FCS_TLSC_EXT.1.1

#### TSS Assurance Activities

#### Assurance Activity AA-FCS_TLSC_EXT.1.1-ASE-01

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

#### Summary

Section 8.9.1 *EAP-TLS and TLS* in the TSS of [ST] describes the EAP-TLS and TLS protocols and ciphersuites supported by the TOE. This section describes how these protocols can be used for WLAN.

The TOE supports EAP-TLS with TLS v1.0, v1.1, and v1.2 with the following ciphersuite:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC5246
- 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

The TOE supports TLS v1.2 with the following 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
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
- 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

The evaluator verified that the TSS information above is consistent with the definition of FCS_TLSC_EXT.1.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1.1-AGD-01

*The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the product so that TLS conforms to the description in the TSS.*

### Summary

[CCGUIDE] section 5.3.2 *TLS Configuration* provides related guidance for configuring TLS. It states the supported cipher suites below are automatically selected by the TOE (i.e., the TOE does not support the individual selection of TLS cipher suites.) The TLS cipher suites available are defined by the TLS server where all cipher suites listed in the ST are always available. Thus, no additional configuration is required by the administrator. It also states that TLS is provided by the APIs of TOE OS Security Framework, which uses the Apple corecrypto Module [Apple ARM, User, Software, SL1] which implements TLS 1.2 (along with TLS 1.0 and TLS 1.1 for EAP-TLS) supporting the cipher suites listed in Table 10 (and Table 9 for EAP-TLS) of [CCGUIDE] which are duplicated below.

For TLS 1.2, the following cipher suites are supported by the TOE in the evaluated configuration:

- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

For TLS 1.0, 1.1, and 1.2, the TOE supports the following TLS cipher suites in the evaluated configuration:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256

Other configurations for TLS can be done via a Configuration Profile as described in section 5.3.2 including setting the TLS version to 1.2 (for the evaluated configuration). Additionally, the administrator must configure the TOE to reject untrusted HTTPS certificates by setting the *allowUntrustedTLSPrompt* key to ‘false' in the Restrictions Payload; configure which server certificate common names and certificates will be accept by the TOE using the *TLSTrustedServerNames* and *PayloadCertificateAnchorUUID* keys in the Wi-Fi Payload; set the reference identifier as described

in [CKTSREF] (chapter *Policies* , section "Obtain policies for establishing trust"); and if needed, add additional CAs using the *EAPClientConfiguration* , *PayloadCertificateArchorUUID* , and *TLSTrustedServerNames* dictionary keys in the Wi-Fi Payload.

Additionally, section 5.3.2 of [CCGUIDE] also references *Appendix: Configuration Profiles* for a sample configuration profile.

### Test Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1.1-ATE-01

*The evaluator shall also perform the following tests:*

- ***Test 1:*** *The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).*
- ***Test 2:*** *The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.*
  *The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.*
- ***Test 3:*** *The evaluator shall send a server certificate in the TLS connection that does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.*
- ***Test 4:*** *The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.*
- ***Test 5:*** *The evaluator shall perform the following modifications to the traffic:*
  - ***Test 5.1:*** *Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.*
  - ***Test 5.2:*** *Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.*
  - ***Test 5.3:*** *[conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.*
  - ***Test 5.4:*** *Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.*
  - ***Test 5.5:*** *[conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.*
  - ***Test 5.6:*** *Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.*
  - ***Test 5.7:*** *Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.*

### Summary

Test 1: The evaluator configured the WAP to use each cipher suite defined in [ST] and connected the TOE to the WAP, while monitoring the traffic using Wireshark.

Test 2: The evaluator created a certificate that did not have the "Server Authentication" in the extendedKeyusage field. The evaluator then attempted to connect the TOE to the WAP and verified that the connection was unsuccessful.

Test 3: The evaluator configured the WAP in such a way that the selected ciphersuite does not match the ciphersuite listed in the certificate. The evaluator attempted to connect the TOE to the WAP and verified that it failed.

Test 4: The evaluator configured the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite by using a modified version of OpenSSL and verified that the client (the TOE) denied the connection.

For all tests in Test 5, the evaluator built a web server using various modified versions of OpenSSL.

Test 5.1: The evaluator changed the version of TLS to 1.3 using a modified version of OpenSSL and verified that the TOE denies the connection.

Test 5.2: The evaluator changed the version of TLS to 1.3 using a modified version of OpenSSL and verified that the TOE denies the connection.

Test 5.3: The evaluator changed a byte in the Server Hello message using a modified version of OpenSSL and verified that the TOE denies the connection.

Test 5.4: The evaluator changed the ciphersuite in the Server Hello message using a modified version of OpenSSL and verified that the TOE denies the connection.

Test 5.5: The evaluator changed the server's Key Exchange message using a modified version of OpenSSL and verified that the TOE denies the connection.

Test 5.6: The evaluator changed Server Finished message using a modified version of OpenSSL and verified that the TOE denies the connection.

Test 5.7: The evaluator sent a message of random bytes after the Change Cipher Spec message using a modified version of OpenSSL and verified that the TOE denies the connection.

## FCS_TLSC_EXT.1.2

### TSS Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1.2-ASE-01

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

### Summary

Section 8.9.1 *EAP-TLS and TLS* in the TSS of [ST] describes the EAP-TLS and TLS protocols and ciphersuites supported by the TOE.

The evaluator reviewed section 8.9.1 and determined that the TOE will compare the Subject Alternative Name (SAN) contained within the peer certificate (specifically the SAN fields, IP Address, and Wildcard certificate if applicable) to the FQDN of the requested server. The Common Name (CN) is ignored. If the FQDN in the certificate does not match the expected SAN for the peer, then the application cannot establish the connection.

Section 8.9.1 also states that certificate pinning is supported by the TOE. The user of the TLS framework can use certificate pinning. Note that TLS clients in the TOE (such as Safari) do not support certificate pinning. Additionally, WLAN also does not support certificate pinning.

### Guidance Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1.2-AGD-01

*The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.*

### Summary

[CCGUIDE] section 5.3.2 *TLS Configuration* provides related guidance for configuring the reference identifier. It states the following:

- Mobile device administrators can use the functions of the Certificate, Key, and Trust Services [CKTSREF] API to manage and manipulate certificates.
- The iOS/iPadOS device implements a set of X.509 policy checks that cannot be altered. If an application wants to enforce additional checks, it can use the API detailed in [CKTSREF] .
- When interpreting the term "reference identifier" as the name of the remote peer whose certificate should be validated, the TOE TLS and IKE stacks set the FQDN of the remote peer with the X.509 protocol checker. This operation is hard-coded and cannot be influenced by the user via any API when using TLS or IKE.
- Guidance documentation for setting additional constraints in validating an X.509 certificate can be specified with the rule definitions in the “Security Policy Keys” section of the “Policies” chapter in [CKTSREF] .

### Test Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1.2-ATE-01

*[TD0499] The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7.*

- ***Test 1:*** *The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.*
  *Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.*
- ***Test 2:*** *The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.*
- ***Test 3:*** *[conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.*
- ***Test 4:*** *The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.*
- ***Test 5:*** *The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.*
  - ***Test 1:*** *[conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.*
  - ***Test 2:*** *[conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that*

*the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.*

- ***Test 3:** [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.*
- ***Test 4:** [conditional]: If wildcards are not supported, the evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection fails.*

- ***Test 6:** [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.*
- ***Test 7:** [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.*

**Summary**

Test 1: The evaluator generated keys and certificates for the TLS server that did not contain the right Common Name (CN) or Subject Alternative Name (SAN). The evaluator connected the TOE to https://server and verified that either Safari asks for confirmation or that the connection is rejected.

Test 2: The evaluator generated keys and certificates for the server side with a CN that matches the reference identifier, contains the SAN extension, but does not contain a SAN that matches the reference identifier. The evaluator connected the TOE to the TLS test server and verified that the connection failed. The evaluator performed this test for both Domain Name Server (DNS) and Uniform Resource Identifier (URI) SANs.

Test 3 is not applicable because the TOE mandates the presence of a SAN and does not process the CN.

Test 4: The evaluator generated keys and certificates for the server that contained a CN that did not match the reference identifier but contained a SAN extension that did match. The evaluator connected the TOE to the TLS server and verified that the connection was successful.

Test 5.1 - 5.3 are not applicable because the TOE does not support wildcard certificates. Test 5.4: The evaluator generated keys and certificates for the server which contained *.domain.com as a wildcard in the SAN. The evaluator attempted to connect the TOE to https://subdomain.domain.com and verified that the connection was not established.

Test 6: The evaluator tested certificates using correct and incorrect URI and DNS names and got the expected results.

Test 7 is not applicable because the TOE does not support CN.

## FCS_TLSC_EXT.1.3

### TSS Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1.3-ASE-01

*If the selection for authorizing override of invalid certificates is made, then the evaluator shall ensure that the TSS includes a description of how and when user or administrator authorization is obtained. The evaluator shall also ensure that the TSS describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.*

**Summary**

Per FCS_TLSC_EXT.1.3 {TLS}, the TOE shall not establish a trusted channel if the server certificate is invalid with no exceptions. In other words, no overriding of invalid certificates is allowed. Therefore, no TSS description is necessary.

**Guidance Assurance Activities**

No assurance activities defined.

**Test Assurance Activities**

**Assurance Activity AA-FCS_TLSC_EXT.1.3-ATE-01**

*[TD-0513] The evaluator shall demonstrate that using an invalid certificate (unless excepted) results in the function failing as follows, unless excepted:*

- ***Test 1a:*** *The evaluator shall demonstrate that a server using a certifcate with a valid certification path successfully connects.*
- ***Test 1b:*** *The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.*
- ***Test 1c [conditional]:*** *If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.*
- ***Test 2:*** *The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.*
- ***Test 3:*** *The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.*
- ***Test 4:*** *The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.*

**Summary**

Test 1a: The evaluator established a successful connection using a certificate with a valid certification path.

Test 1b: The evaluator modified the certificate chain so it was no longer valid and verified that the connection was not established.

Test 1c is not applicable because CA certificates distributed via a trusted MDM are always trusted by the TOE and the user cannot alter the trust settings.

Test 2 is performed in conjunction with FIA_X509_EXT.1 Test 4.

Test 3 is performed in conjunction with FIA_X509_EXT.1 Test 2.

Test 4 is performed in conjunction with FCS_TLSC_EXT.1.2 Test 1.

# 2.2.2.31 Extensible Authentication Protocol-Transport Layer Security (FCS_TLSC_EXT.1/WLAN)

## TSS Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1-WLAN-ASE-01

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

**Summary**

Section 8.9.1 *EAP-TLS and TLS* in the TSS of [ST] describes the EAP-TLS and TLS protocols. For EAP-TLS, this section states that the TOE supports TLS versions 1.0, 1.1. and 1.2 with the following ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC5246
- 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

The evaluator verified that these ciphersuites are consistent with those specified in FCS_TLSC_EXT.1/WLAN.

The evaluator checked the administrative guidance [CCGUIDE] and could verify that there is no configuration required for the TOE. They are automatically selected by the TOE.

## Guidance Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1-WLAN-AGD-01

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

*The evaluator shall check that the OPE guidance contains instructions for the administrator to configure the list of Certificate Authorities that are allowed to sign certificates used by the authentication server that will be accepted by the TOE in the EAP-TLS exchange, and instructions on how to specify the algorithm suites that will be proposed and accepted by the TOE during the EAP-TLS exchange.*

**Summary**

[CCGUIDE] section 5.3.1 *EAP-TLS Configuration* provides guidance related to EAP-TLS. It states that the supported ciphersuites listed in the Table 9 "EAP-TLS Ciphersuites" (reproduced below) are automatically selected by the TOE (i.e., the TOE does not support the individual selection of EAP-TLS cipher suites) when Wi-Fi Protected Access (WPA)-EAP is configured via (by an administrator) a Configuration Profile as follows:

- *EncryptionType* key must be set to ‘WPA2’.
- *AcceptEAPTypes* key must be set to '13' (as the value representing EAP-TLS)

Additional detail on these keys are provided in [DEV_MAN] .

| Ciphersuite Name |
|---|
| TLS_RSA_WITH_AES_128_CBC_SHA |

| Ciphersuite Name |
|---|
| TLS_RSA_WITH_AES_256_CBC_SHA |
| TLS_RSA_WITH_AES_128_CBC_SHA256 |
| TLS_RSA_WITH_AES_256_CBC_SHA256 |

**Table 7: EAP-TLS Ciphersuites**

[CCGUIDE] section 3.3.2.3 *Mobile device administrators* provides guidance for configuring CAs by stating that additional CAs (beside those preinstalled with the iOS/iPadOS) can be added using a Configuration Profile with the *EAPClientConfiguration* , *PayloadCertificateArchorUUID* , and *TLSTrustedServerNames* dictionary keys in the Wi-Fi Payload which are described in detail in [DEV_MAN] .

Additionally, a sample Configuration Profile for configuring the WLAN is provided in chapter Appendix A *Configuration Profiles* of [CCGUIDE] .

## Test Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.1-WLAN-ATE-01

*The evaluator shall write, or the ST author shall provide, an application for the purposes of testing TLS.*

*The evaluator shall also perform the following tests:*

- *Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128- bit AES and not 256-bit AES).*
- *Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.*
- *Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.*
- *Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.*
- *Test 5: The evaluator shall perform the following modifications to the traffic:*
  - *Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.*
  - *Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.*
  - *Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.*
  - *[TD0492] [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the Server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.*
  - *Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.*

- *Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.*

**Summary**

Test 1: The evaluator configured the AP to use each cipher suite defined in [ST] and connected the TOE to the AP while monitoring the traffic using Wireshark.

Test 2: The evaluator created a certificate that did not have the "Server Authentication" in the extendedKeyusage field, attempted to connect the TOE to the AP, and verified that the connection was unsuccessful.

Test 3: The evaluator configured the AP in such a way that the selected ciphersuite does not match the ciphersuite listed in the certificate, attempted to connect the TOE to the AP, and verified that it failed.

Test 4: Using a modified version of OpenSSL, the evaluator configured the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verified that the the TOE (client) denied the connection.

Test 5.1: Using a modified version of OpenSSL, the evaluator changed the version of TLS used and verified that the TOE denies the connection.

Test 5.2: Using a modified version of OpenSSL, the evaluator modified the first byte of the random number sent in the server hello and verified that the TOE denies the connection.

Test 5.3: The evaluator set the first byte of the cipher spec field in the server hello message to 0xff using a modified version of OpenSSL and verified that the TOE denies the connection.

Test 5.4: Using a modified version of OpenSSL, the evaluator added one to the first byte of the RSA signature of the server key exchange message and verified that the TOE denies the connection.

Test 5.5: Using a modified version of OpenSSL, the evaluator added one to the first byte of the PRF digest server finished message and verified that the TOE denies the connection.

Test 5.6: Using a modified version of OpenSSL, the evaluator sent a garbled message after the change cipher spec and verified that the TOE denies the connection.

## 2.2.2.32 TLS Client Support for Mutual Authentication (FCS_TLSC_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_TLSC_EXT.2-ASE-01

*The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.*

**Summary**

Section 8.5.2 *X.509v3 Certificates* describes X.509 certificates used by the TOE and how the TOE uses certificates for different services including for TLS and EAP-TLS authentication. This section also describes how client-side certificates can be installed and how they are used for TLS mutual authentication.

Section 8.9.1.1 *TLS mutual authentication* describes how the TOE uses client-side cerificate for TLS mutual authentication:

- Each application can have more than one client certificate.

- For X.509 certificates, the TOE allows applications to store client certificates in either a P12 file or in a keychain. A P12 file only holds one client certificate, but a keychain can hold multiple client certificates.
- For applications that use a P12 file, there is only one client certificate choice when performing TLS mutual authentication. The application passes this choice to the TLS API.
- For applications that use a keychain, there may be multiple client certificate choices. An application can select the appropriate client certificate from the keychain and pass the selected certificate to the TLS API or it can pass an array of client certificates to the TLS API. The TLS API uses the first certificate in the array and ignores the other certificates in the array when establishing a connection.
- There are no other factors beyond configuration necessary to engage in mutual authentication.

## Guidance Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.2-AGD-01

*The evaluator shall ensure that the AGD guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.*

### Summary

[CCGUIDE] sections 5.3.2 *TLS Configuration* provides related guidance for configuring client-side X.509 certificates which involves installing a client certificate and its keys on the device using the Certificate Payload in a Configuration Profile as described [DEV_MAN] . Also, examples of Configuration Profiles can be found in Appendix A: Configuration Profiles of [CCGUIDE].

## Test Assurance Activities

### Assurance Activity AA-FCS_TLSC_EXT.2-ATE-01

*The evaluator shall also perform the following tests:*

- ***Test 1:*** *The evaluator shall establish a connection to a server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.*
- ***Test 2:*** *The evaluator shall establish a connection to a server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.*

### Summary

Test 1: The evaluator generate keys and certificates for the test server and connected the TOE to it. The evaluator verified that the server was not configured to require client authentication and analyzed the packet traffic using the tcpdump tool. The evaluator also verified the information on the web page displayed by the server.

Test 2: The evaluator re-performed Test 1, where the server is this time configured to require mutual authentication. The evaluator verified in the packet capture that the server requested the certificate from the client and that the client sent it to the server.

## 2.2.2.33 TLS Client Support for Renegotiation (FCS_TLSC_EXT.4)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_TLSC_EXT.4-ATE-01

*The evaluator shall perform the following tests:*

- ***Test 1:*** *The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the "renegotiation_info" field or the SCSV cipher suite is included in the ClientHello message during the initial handshake.*
- ***Test 2:*** *The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the "renegotiation_info" extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.*
- ***Test 3:*** *The evaluator shall verify that ServerHello messages received during secure renegotiation contain the "renegotiation_info" extension. The evaluator shall modify either the "client_verify_data" or "server_verify_data" value and verify that the client terminates the connection.*

#### Summary

Test 1: The evaluator examined the pcap data obtained while performing the test for FCS_TLSC_EXT.2 Test 1 and found the renegotiation_info field to be present in the Client Hello.

Test 2: The evaluator used a modified version of OpenSSL to cause the Server Hello message to include a renegotiation_info field with a length of one and verified this caused the TOE client to generate a fatal alert. A successful connection was shown in the data examined for Test 1.

Test 3: The evaluator used a modified version of OpenSSL to modify the client_verify_data and server_verify_data values and verified the TOE terminated the connection.

## 2.2.2.34 TLS Client Support for Supported Groups Extension (FCS_TLSC_EXT.5)

### TSS Assurance Activities

#### Assurance Activity AA-FCS_TLSC_EXT.5-ASE-01

*The evaluator shall verify that TSS describes the Supported Groups Extension.*

#### Summary

Section 8.9.1 *EAP-TLS and TLS* states that the elliptic curve cipher suites claimed in the ST may utilize the following supported elliptic curve extenstions by default:

- secp256r1 (P-256)
- secp384r1 (P-384)
- secp521r1 (P-521) (SigGen/SigVer only)

In addition, the TOE supports elliptic curve extension Curve25519 by default.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FCS_TLSC_EXT.5-ATE-01

*The evaluator shall also perform the following test:*

- ***Test 1:*** *The evaluator shall configure a server to perform key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.*

#### Summary

Test 1: The evaluator configured the TLS server to only accept one curve and connected the TOE to the server. The evaluator performed this task for each curve supported by the TOE.

## 2.2.3 User data protection (FDP)

## 2.2.3.1 Access Control for System Services (FDP_ACF_EXT.1)

### FDP_ACF_EXT.1.1

#### TSS Assurance Activities

#### Assurance Activity AA-FDP_ACF_EXT.1.1-ASE-01

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

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

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

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

*For any services for which the user may grant access, the evaluator shall ensure that the TSS identifies whether the user is prompted for authorization when the application is installed, or during runtime.*

#### Summary

Section 8.4 *User Data Protection (FDP)* in the TSS of [ST] describes how the TOE protects user and TSF data.

This section describes what an application is allowed to access on the system and which files it can access. Each application is sandboxed which means that it can only access the files that belong to itself. Regarding the capabilities of the application to access system services, every application must declare device-specific capabilities they need to run. These capabilities are defined in an array or dictionary that contains keys identifying features that the application requires. Applications

owned by the same developer account can share content if configured to be part of an application group on the device. Creating such an appropriate group is up to the developer. Moreover, the TOE allows a user to restrict the applications to access certain system services such as the following:

- Location Services
- Tracking
- Contacts
- Calendar
- Reminders
- Photos
- Bluetooth
- Local Network
- Nearby Interactions
- Microphone
- Speech Recognition
- Cameras
- Health
- Research Sensor & Usage Data
- HomeKit
- Media & Apple Music
- Files and Folders
- Motion & Fitness
- Focus
- Analytics & Improvements
- Apple Advertising
- App Privacy Report
- Record App Activity

Futhermore, this section states that

> " *Applications prompt the mobile device user to grant permission for the application to use system services when they are installed.* "

**Guidance Assurance Activities**

**Assurance Activity AA-FDP_ACF_EXT.1.1-AGD-01**

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

**Summary**

[CCGUIDE] section 5.4.2 *Restrict Application Access to System Services* states the following:

- Access control to system services in the Core Services layer is hardcoded and thus not configurable by the mobile device user or administrator.
- Access control for applications to system services can be restricted on a per-app basis.
- A list of system services can be obtained from the mobile device Settings » Privacy.

- For TOE normal users, for each system service, the applications which have permissions to use that service can be inspected and changed.
- For TOE administrators, they cannot specify access control for applications to system services.

### Test Assurance Activities

#### Assurance Activity AA-FDP_ACF_EXT.1.1-ATE-01

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

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

- ***Test 1:*** *For each system service to which no applications are allowed access, the evaluator shall attempt to access the system service with a test application and verify that the application is not able to access that system service.*
- ***Test 2:*** *For each system service to which only privileged applications are allowed access, the evaluator shall attempt to access the system service with an unprivileged application and verify that the application is not able to access that system service. The evaluator shall attempt to access the system service with a privileged application and verify that the application can access the service.*
- ***Test 3:*** *For each system service to which the user may grant access, the evaluator shall attempt to access the system service with a test application. The evaluator shall ensure that either the system blocks such accesses or prompts for user authorization. The prompt for user authorization may occur at runtime or at installation time, and should be consistent with the behavior described in the TSS.*
- ***Test 4:*** *For each system service listed in the TSS that is accessible by all applications, the evaluator shall test that an application can access that system service.*

#### Summary

Test 1 is not applicable because [ST] specifies that no services are accessible to any applications.

Test 2 is not applicable because [ST] specifies that no services grant special access to privileged applications.

Test 3: The evaluator opened the Edhita editor and created a directory and a file with some content. The evaluator then opened iEditor and attempted to locate the directory or file and was unsuccessful. The evaluator reversed the process, created a directory or file in iEditor and verified that it could not be seen nor accessed by Edhita. The evaluator then installed configuration profiles on the TOE to supply it with certificates and WiFi credentials, connected the TOE to the WLAN network, and accessed https://server using Chrome. The evaluator verified that Chrome does not accept the web certificate as trusted (even though the CA was imported by the profile and the CN applies), showing that the certificates imported for Safari are not usable).

Test 4: The evaluator opened the Edhita editor and created a directory and a file with some content. The evaluator then opened iEditor and attempted to locate the directory or file and was unsuccessful. The evaluator reversed the process, created a directory or file in iEditor and verified that it could not be seen nor accessed by Edhita. The evaluator then installed configuration profiles on the TOE to supply it with certificates and WiFi credentials, connected the TOE to the WLAN network, and accessed https://server using Safari. The evaluator selected the RSA certificate and verified that the cryptographic details were displayed on the web page, indicating a successful connection.

## FDP_ACF_EXT.1.2

### TSS Assurance Activities

#### Assurance Activity AA-FDP_ACF_EXT.1.2-ASE-01

*The evaluator shall examine the TSS to verify that it describes which data sharing is permitted between applications, which data sharing is not permitted, and how disallowed sharing is prevented. It is possible to select both "applications" and "groups of applications", in which case the TSS is expected to describe the data sharing policies that would be applied in each case.*

**Summary**

Section 8.4 *User Data Protecction (FDP)* in the TSS of [ST] describes how the TOE protects user and TSF data.

Subsections 8.4.1 and 8.4.2 state that applications are sandboxed and can only access data within their own directory in the file tree structure, unless they are part of an app group set up by the application developer as described in subsection 8.4.4. During the installation of the application, the installer creates app containers limited to the directories inside the application's sandbox which enforces which files the application can access.

**Guidance Assurance Activities**

No assurance activities defined.

**Test Assurance Activities**

**Assurance Activity AA-FDP_ACF_EXT.1.2-ATE-01**

- ***Test 1:*** *The evaluator shall write, or the developer shall provide, two applications, one that saves data containing a unique string and the other, which attempts to access that data. If "groups of applications" is selected, the applications shall be placed into different groups. If "application" is selected, the evaluator shall install the two applications. If "private data" is selected, the application shall not write to a designated shared storage area. The evaluator shall verify that the second application is unable to access the stored unique string. If "the user" is selected, the evaluator shall grant access as the user and verify that the second application is able to access the stored unique string. If "the administrator" is selected, the evaluator shall grant access as the administrator and verify that the second application is able to access the stored unique string. If "a common application developer" is selected, the evaluator shall grant access to an, application with a common application developer to the first, and verify that the application is able to access the stored unique string.*

**Summary**

The evaluator verified that the iCalendar app exports an interface to the iMail app via the UIDocumentInteractionController to allow invitations received by the Mail system to the Calendar. The evaluator registered an account with iCloud and ensured that Mail is active. He then created a calendar invite from a different system and invite the iCloud account. The evaluator waited for the email invite to appear with iMail, opened it, opened the attached ical file, and verified that it displayed the event. The evaluator then imported it and verified in iCalendar that the event is listed. The evaluator then logged out of iCloud.

## 2.2.3.2 Protected Data Encryption (FDP_DAR_EXT.1)

**TSS Assurance Activities**

**Assurance Activity AA-FDP_DAR_EXT.1-ASE-01**

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

**Summary**

Section 8.4 *User Data Protecction (FDP)* in the TSS of [ST] describes how the TOE implements data at rest (DAR).

Protection of files The TOE implements the following basic classes and policies:

- **Complete Protection** (a.k.a. class A): Files in this class can only be accessed when the device is unlocked.
- **Protected Unless Open** (a.k.a. class B): Files in this class may need to be written while the device is locked.
- **Protected Until First User Authentication** (a.k.a. class C): Files in this class are protected until the user has successfully authenticated.
- **No Protection** (a.k.a. class D): Files in this class can always be accessed. However, the files themselves are encrypted and decrypted

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

Protection of keychain data

Section 8.4.6 describes how the keychain data is separated into four different categories (similar to the class structure of the file data protection above) in table 10 "Keychain to File-system Mapping". The securityd daemon determines which keychain items each process or application can access.

Table 9 "Summary of keys and persistent secrets used by the MDM Agent" describes the keys and persistent secrets the TOE stores for the agent.

Further details on how the TOE protects data is provided in section 8.3.1 *Overview of Key Management* . In particular, the description after figure 5 "Key Hierarchy in the TOE OS" clearly indicates that all data including all file system items, metadata, files and keychains are encrypted.

## Guidance Assurance Activities

### Assurance Activity AA-FDP_DAR_EXT.1-AGD-01

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

#### Summary

[CCGUIDE] section 5.4.1 *Data-At-Rest (DAR) Protection Configuration* provides related guidance on DAR. It states that data is always encrypted for protection which requires the use of a passcode on the device. Administrators must ensure that TOE users set a passcode by using the *forcePin* key in the Passcode Policy Payload via a Configuration Profile. Additionally, a sample Configuration Profile for configuring passcode restrictions is provided in *Appendix: Configuration Profiles* of [CCGUIDE] .

Users can check that data protection is enabled on their device in the Settings at *Touch ID and Passcode* on devices without Face ID functionality (models with a Home button) and Face ID and Passcode on devices with Face ID functionality (models without a Home button). A passcode is required to access the device and this enables data protection on the device. No further configuration is required.

In addition, this sections states that the TOE only supports external storage encryption with storage devices formatted in the APFS format, other formats with encryption or encrypted volumes are not supported by the TOE. In the evaluated configuration, external storage devices must be formatted in the APFS file format and volumes must be encrypted. All other storage formats are not allowed

in the evaluated configuration. Mobile device administrators must ensure through organizational policies that mobile device users only use external storage devices formatted in the APFS format with encrypted volumes.

## Test Assurance Activities

### Assurance Activity AA-FDP_DAR_EXT.1-ATE-01

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

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

#### Summary

Test 1: Currently no tool is available that is able to bypass the encryption/decryption functionality for data. The encryption/decryption is performed directly by the storage controller as follows:

- XNU provides the wrapped file meta data to SEP.
- SEP unwraps the meta data to access the plaintext encryption key.
- SEP re-wraps the meta data with a new key that was established during boot with the storage controller.
- SEP sends the wrapped meta data back to XNU.
- XNU sends the wrapped meta data to the storage controller together with the file access request.
- The storage controller unwraps the meta data and performs the file encryption/decryption with the unwrapped key.
- Storage controller receives/sends plaintext file contents from/to XNU.

Therefore, checking that data is stored encrypted is not possible as discussed and agreed with NIAP for previous evaluations. However, the tester performs the following inspection:

- check the SEP code to verify the steps 2 through 4 in the SEP code path.
- verify that the unwrapped file key is immediately zeroized after the re-wrapping operation.

In addition, the developer explained that the following test is performed when new hardware is released:

- Test vectors using the cryptographic implementation of CoreCrypto are generated:
  - for devices starting with A9:
    - Encryption: AES XTS 128 with key, tweak key, IV, and plaintext
    - Decryption: AES XTS 128 with key, tweak key, IV, and ciphertext
  - for devices up to and including A14:
    - Encryption: AES XTS 256 with key, IV, and plaintext
    - Decryption: AES XTS 256 with key, IV, and ciphertext
  - for all test vectors, the expected ciphertext (encryption) or plaintext (decryption) is calculated
- The generated test vectors are sent to the storage controller to perform the respective cryptographic operation.

- The resulting data (ciphertext for encryption, plaintext for decryption) is compared to the expected data generated by CoreCrypto for the test vectors. If both match, the test is considered to be successful. Otherwise, the test fails and indicates a broken hardware implementation causing a rework of the hardware.

## 2.2.3.3 Sensitive Data Encryption (FDP_DAR_EXT.2)

### FDP_DAR_EXT.2.1

#### TSS Assurance Activities

#### Assurance Activity AA-FDP_DAR_EXT.2.1-ASE-01

*The evaluator shall verify that the TSS includes a description of which data stored by the TSF (such as by native applications) is treated as sensitive. This data may include all or some user or enterprise data and must be specific regarding the level of protection of email, contacts, calendar appointments, messages, and documents.*

*The evaluator shall examine the TSS to determine that it describes the mechanism that is provided for applications to use to mark data and keys as sensitive. This description shall also contain information reflecting how data and keys marked in this manner are distinguished from data and keys that are not (for instance, tagging, segregation in a "special" area of memory or container, etc.).*

#### Summary

Section 8.4.6 *Keychain Data Protection* in the TSS of [ST] states the following:

- The TOE OS Keychain provides a secure way to store sensitive bits of data such as keys, login tokens, digital certificates for VPN connections, and certificates and private keys installed by the Configuration Profile.
- The Keychain is implemented as a SQLite database and the securityd daemon determines which Keychain items each process or app can access.
- Keychain items can only be shared between apps from the same developer.
- Keychain data is protected using a class structure similar to the one used in file Data Protection but with distinct keys.
- Keychains can use ALCs to set policies for accessbility and authentication requirements

Section 8.4.4 *App Groups* states that apps and extensions owned by a given developer account can share content when configured to be part of an App Group. Once configured, apps can have a shared container for stoarge, shared preferences, shared Keychain items.

#### Guidance Assurance Activities

No assurance activities defined.

#### Test Assurance Activities

#### Assurance Activity AA-FDP_DAR_EXT.2.1-ATE-01

- ***Test 1:*** *The evaluator shall enable encryption of sensitive data and require user authentication according to the AGD guidance. The evaluator shall try to access and create sensitive data (as defined in the ST and either by creating a file or using an application to generate sensitive data) in order to verify that no other user interaction is required.*

#### Summary

For Test 1, The storage encryption outlined for FDP_DAR_EXT.1 is the same that provides the extended storage encryption support by using the class keys. Thus, the testing for FDP_DAR_EXT.1 covers the data storage part of this test. The key life cycle description provided in FCS_CKM_EXT.4 demonstrates the key management part of the test. Therefore, both analyses fully cover the test requirement for this SFR.

## FDP_DAR_EXT.2.2

### TSS Assurance Activities

### Assurance Activity AA-FDP_DAR_EXT.2.2-ASE-01

*The evaluator shall review the TSS section of the ST to determine that the TSS includes a description of the process of receiving sensitive data while the device is in a locked state. The evaluator shall also verify that the description indicates if sensitive data that may be received in the locked state is treated differently than sensitive data that cannot be received in the locked state. The description shall include the key scheme for encrypting and storing the received data, which must involve an asymmetric key and must prevent the sensitive data-at-rest from being decrypted by wiping all key material used to derive or encrypt the data (as described in the application note). The introduction to this section provides two different schemes that meet the requirements, but other solutions may address this requirement.*

### Summary

Section 8.4.6 *Keychain Data Protection* in the TSS of [ST] describes how the TOE protects sensitive data. The TOE OS Keychain provides a secure way to store this data. It states that the Keychain data is protected using a class similar to the one used in file Data Protection. This section provides table 11 "Keychain to File-system Mapping" showing the Keychain classes and their equivalent file system classes. Per this table, the Keychain data protection class "when unlocked" is equivalent to the file data protection class NSFileProtectionComplete. The Keychain data protection class "while locked" is equivalent to the file data protection class NSFIleProtectionCompleteUnlessOpen. The Keychain data protection class "after first unlock" is equivalent to the file data protection NSFileProtectionCompleteUntilFirstUserAuthentication. The Keychain protection class "Always" is equivalent to NSFileProtectionNone. Description of the equivalent file data protection classes are described in detailed in section 8.3.1 *Overview of Key Management* .

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

### Assurance Activity AA-FDP_DAR_EXT.2.2-ATE-01

*The evaluator shall perform the tests in FCS_CKM_EXT.4 for all key material no longer needed while in the locked state and shall ensure that keys for the asymmetric scheme are addressed in the tests performed when transitioning to the locked state.*

### Summary

The key lifecycle testing performed as part of FCS_CKM_EXT.4 covers the destruction of all class keys, including the private and public class B keys referenced in this SFR. The evaluator notes that the asymmetric key handling with its Diffie-Hellman operation as described in [PP_MDF_V3.2] Figure 4 is implemented by the class B key pair.

## FDP_DAR_EXT.2.3

### TSS Assurance Activities

### Assurance Activity AA-FDP_DAR_EXT.2.3-ASE-01

*The evaluator shall verify that the key hierarchy section of the TSS required for FCS_STG_EXT.2.1 includes the symmetric encryption keys (DEKs) used to encrypt sensitive data. The evaluator shall ensure that these DEKs are encrypted by a key encrypted with (or chain to a KEK encrypted with) the REK and password-derived or biometric-unlocked KEK.*

*The evaluator shall verify that the TSS section of the ST that describes the asymmetric key scheme includes the protection of any private keys of the asymmetric pairs. The evaluator shall ensure that any private keys that are not wiped and are stored by the TSF are stored encrypted by a key encrypted with (or chain to a KEK encrypted with) the REK and password-derived or biometric-unlocked KEK.*

*The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.*

### Summary

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] provides detailed description on how the TOE performs key management. In particular figure 5 illustrates how the key hierarchy is implemented within the TOE OS. The evaluator verified that the DEKs are encrypted by a key that is chained to the REK. Some keys, like the passcode key, are actually password derived (using the REK as well). Regarding asymmetric cryptography keys, the private keys are wrapped whenever the device is locked. These keys will be unwrapped when the user provides the correct passcode for the device.

The evaluator determined that the description is sufficiently detailed, the key management hiearchy both in text description as well as illustrated in figure 5 is clear and meets the requirements to ensure the keys are adequately protected. Also, the description expands in multiple paragraphs and includes a flow-chart diagram in figure 5 which clearly depicts the key hierarchy in the TOE OS.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## FDP_DAR_EXT.2.4

### TSS Assurance Activities

### Assurance Activity AA-FDP_DAR_EXT.2.4-ASE-01

*The evaluator shall verify that the TSS section of the ST that describes the asymmetric key scheme includes a description of the actions taken by the TSF for the purposes of DAR upon transitioning to the unlocked state. These actions shall minimally include decrypting all received data using the asymmetric key scheme and re-encrypting with the symmetric key scheme used to store data while the device is unlocked.*

### Summary

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] provides detailed description on how the TOE performs key management. Asymmetric cryptography is used for verifying signatures of the software, protocols (key establishment), and when the device receives data while it is locked. The private keys are wrapped by asymmetric cryptography whenever the device is locked. The asymmetric keys used to wrap these keys are destroyed and can only be re-generated whenever the user provides the correct passcode. When the device is unlocked, the keys protecting data received during the locked state are being unwrapped and re-wrapped with a different key.

#### Guidance Assurance Activities

No assurance activities defined.

#### Test Assurance Activities

No assurance activities defined.

## 2.2.3.4 Subset Information Flow Control (FDP_IFC_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FDP_IFC_EXT.1-ASE-01

*[TD0596] The evaluator shall verify that the TSS section of the ST describes the routing of IP traffic through processes on the TSF when a VPN client is enabled. The evaluator shall ensure that the description indicates which traffic does not go through the VPN and which traffic does. The evaluator shall verify that a configuration exists for each baseband protocol in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) or needed for the correct functioning of the TOE is not encapsulated by the VPN protocol (IPsec). The evaluator shall verify that the TSS section describes any differences in the routing of IP traffic when using any supported baseband protocols (e.g. Wi-Fi or, LTE).*

#### Summary

Section 8.9.4 *VPN* in the TSS of [ST] describes the VPN functionality which states the following:

- IPsec is implemented in the TOE natively, as part of the TOE OS, hence the packets are processed by the TOE. P ackets are processed in little-endian order. There is no separate “client” application; the VPN tunnels are configured and controlled by Network Extension Framework, which is a part of the host operating system’s Core OS Layer.
- The TOE implements the IPsec protocol as specified in RFC 4301. Configuration of VPN connection setting, such as, authentication method and algorithm selection, is performed by the IPsec VPN client administrator.
- The TOE enforces an “always on” (AlwaysOn) configuration by default meaning that all traffic entering and leaving the TOE platform interfaces is protected via an IPsec VPN connection.
- The TOE allows limited services such as VoiceMail, Cellular Services, and AirPrint to be configured to either not allow (DISCARD) or be sent plaintext (BYPASS).
- The TOE supports separate configurations for cellular and Wi-Fi.

### Guidance Assurance Activities

#### Assurance Activity AA-FDP_IFC_EXT.1-AGD-01

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

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

**Summary**

[CCGUIDE] section 5.3.5 *VPN Configuration* provides related guidance on VPN configuration. It states that the VPN must be in its Always-On configuration in the evaluated configuration. This configuration enables the organization to have full control over supervised device traffic by tunneling all IP traffic back to the organization. For the TOE users, no configuration is required. The administrator can perform this configuration using the VPN Payload in a Configuration Profile. Also, section 5.3.6 *Keys for Configuring Network Protocols* provides table 12 "Essential Keys for the VPN Payload" outlining the keys and their values that the administrator must specify in the VPN Payload. Additionally, a sample Configuration Profile for configuring the VPN is provided in *Appendix: Configuration Profiles* of [CCGUIDE] .

Further description of the VPN Payload is provided in [DEV_MAN] .

## Test Assurance Activities

### Assurance Activity AA-FDP_IFC_EXT.1-ATE-01

- ***Test 1:*** *[TD0596] If the ST author identifies any differences in the routing between Wi-Fi and cellular protocols, the evaluator shall repeat this test with a base station implementing one of the identified cellular protocols.*
  - *Step 1: The evaluator shall enable a Wi-Fi configuration as described in the AGD guidance (as required by FTP_ITC_EXT.1). The evaluator shall use a packet sniffing tool between the wireless access point and an Internet-connected network. The evaluator shall turn on the sniffing tool and perform actions with the device such as navigating to websites, using provided applications, and accessing other Internet resources. The evaluator shall verify that the sniffing tool captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.*
  - *Step 2: The evaluator shall configure an IPsec VPN client that supports the routing specified in this requirement, and if necessary, configure the device to perform the routing specified as described in the AGD guidance. The evaluator shall ensure the test network is capable of sending any traffic identified as exceptions. The evaluator shall turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step, as well as ensuring that all exception traffic is generated. The evaluator shall verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.*
  - *Step 3: The evaluator shall examine the traffic from both step one and step two to verify that all Data Plane traffic is encapsulated by IPsec, modulo the exceptions identified in the SFR (if applicable). For each exception listed in the SFR, the evaluator shall verify that that traffic is allowed outside of the VPN tunnel. The evaluator shall examine the Security Parameter Index (SPI) value present in the encapsulated packets captured in Step two from the TOE to the Gateway and shall verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from the TOE to the Gateway. The evaluator shall be aware that IP traffic on the cellular baseband outside of the IPsec tunnel may be emanating from the baseband processor and shall verify with the manufacturer that any identified traffic is not emanating from the application processor.*
  - *Step 4: (Conditional: If ICMP is not listed as part of the IP traffic needed for the correct functioning of the TOE) The evaluator shall perform an ICMP echo from the TOE to the IP address of another device on the local wireless network and shall verify that no packets are sent using the sniffing tool. The evaluator shall attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local wireless network, and shall verify that the TOE discards them.*

**Summary**

Step 1: The evaluator connected the TOE to the WiFi WAP on the test laptop and accessed https://server using Safari on the TOE. The evaluator captured that traffic and disconnected from the WLAN.

Step 2: The evaluator loaded the IPsec configuration profile onto the TOE using the Apple Configurator 2, started Strongswan on the test laptop, started the WiFi WAP, and verified that the VPN connection is automatically established. The evaluator then accessed https://server. The evaluator disconnected from the WLAN and re-connected and verified that the VPN connection is automatically re-established.

Step 3: The evaluator recorded all the traffic from Step 2, and reviewed it. The evaluator verified that HTTP traffic is encapsulated into ESP and verified the SPI for both directions. According to the developer, the always-on VPN implies that all traffic is always routed to the VPN channel. If two channels exist (one on WiFi and one on Cellular), the VPN channel on WiFi is preferred. If no VPN channel is found, no traffic other than the captive portal traffic and DNS traffic prior to establishing the VPN link is allowed.

Step 4: The evaluator connected the TOE to the WAP, where Strongswan is not started, and verified that the VPN connection is not established. The evaluator pinged the TOE from the test laptop (to the IP address assigned to the TOE by the VPN connection) and verified that there was no reply from the TOE.

## 2.2.3.5 Subset Information Flow Control (FDP_IFC_EXT.1/VPN)

### TSS Assurance Activities

#### Assurance Activity AA-FDP_IFC_EXT.1-VPN-ASE-01

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

#### Summary

Section 8.9.4.1 *AlwaysOn VPN* and section 8.9.4.2 *IPsec General* describes VPN functionality and IPsec implementation. These sections state the following:

- For managed and supervised devices, the TOE must 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 and/or Wi-Fi) for which the VPN is 'AlwaysOn' (default is: for both cellular and Wi-Fi), the specification of exceptions from this service (only VoiceMail, AirPrint, and CellularServices can be listed as exceptions), and the definition of exceptions for Captive networking (if any).
- The TOE enforces an “always on” configuration meaning that all traffic entering and leaving the TOE platform interfaces is protected via an IPsec VPN connection. The TOE allows a limited number of services to be configured to either not allow (DISCARD) or be sent plaintext (BYPASS). These services include applications that make use of Captive Networking Identifiers, Voice Mail, Cellular Services and AirPrint. All other communications are always sent through the IPsec tunnel (PROTECT within the Security Policy Database (SPD)). In order to set a service to match a PROTECT rule in the SPD, select “Allow traffic via tunnel.” “Drop Traffic” will cause that traffic to match a DISCARD rule. “Allow traffic outside tunnel” will create a BYPASS rule for that service.

- The TOE supports separate configurations for cellular and Wi-Fi.
- There are no differences in the routing of IP traffic when using any of the supported baseband protocols.

## Guidance Assurance Activities

### Assurance Activity AA-FDP_IFC_EXT.1-VPN-AGD-01

*The evaluator shall verify that the following is addressed by the documentation:*

- *The description above indicates that if a VPN client is enabled, all configurations route all IP traffic (other than IP traffic required to establish the VPN connection) through the VPN client.*
- *The AGD guidance describes how the user and/or administrator can configure the TSF to meet this requirement.*

#### Summary

[CCGUIDE] section 5.3.5 *VPN Configuration* provides related guidance on VPN configuration. It states the following:

- In the evaluated configuration, the VPN must be in its Always-On configuration. The Always-On VPN configuration enables the organization to have full control over supervised device traffic by tunneling all IP traffic back to the organization.
- Always-On is enabled by setting the *VPNType key* to 'AlwaysOn' in the Configuration Profile.
- For the user, there is no configuration required.
- For the administrator, they may use the VPN Payload to configure systemwide VPN based on IPsec. Further instructions for the administrators to configure the VPN Payload is provided in section 5.3.6 *Keys for Configuring Network Protocols* of [CCGUIDE] .

## Test Assurance Activities

### Assurance Activity AA-FDP_IFC_EXT.1-VPN-ATE-01

*The evaluator shall perform the following test:*

*Step 1 - The evaluator shall use the platform to enable a network connection without using IPsec. The evaluator shall use a packet sniffing tool between the platform and an Internet-connected network. The evaluator shall turn on the sniffing tool and perform actions with the device such as navigating to websites, using provided applications, accessing other Internet resources (Use Case 1), accessing another VPN client (Use Case 2), or accessing an IPsec-capable network device (Use Case 3). The evaluator shall verify that the sniffing tool captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.*

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

*Step 3 - The evaluator shall examine the traffic from both step one and step two to verify that all IP traffic, aside from and after traffic necessary for establishing the VPN (such as IKE, DNS, and possibly HTTPS), is encapsulated by IPsec.*

*Step 4 - The evaluator shall attempt to send packets to the TOE outside the VPN connection and shall verify that the TOE discards them.*

#### Summary

This test is identical to the test performed for FDP_IFC_EXT.1.

## 2.2.3.6 Storage of Critical Biometric Parameters (FDP_PBA_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FDP_PBA_EXT.1-ASE-01

*The evaluator shall determine that the TSS contains a description of the activities that happen during biometric authentication.*

*The evaluator shall ensure that the authentication template is protected either using a PIN or by other secure means, as specified by the vendor.*

**Summary**

Section 8.5 *Identification and Authentication (FIA)* describes user authentication including biometric authentication mechanisms supported by the TOE. It states the following:

- Biometric authentication inputs do not produce feedback to the user unless an input is rejected.
- For Touch ID, 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 fingerprint samples are presented, then 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 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 between consecutive attempts.
- For Touch ID, the TOE enforces a five 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.
- Successful authentication attempts are achieved exclusively by a successful key derivation that decrypts the keybag in the SEP with the respective class keys.

Section 8.3.1.2 *No plaintext key transmission and export* states that biometric keying material, enrollment and authentication templates, the features an agorithm uses to perform biometric authentication for enrollment or verification, threshold values, intermediate calculations, and final match scores never leave the SEP. Section 8.6.3 further explains that an authentication template is created from the biometric data and stored inside the SEP. The authentication template cannot be retrieved from the SEP. Instead, the SEP can be asked to compare biometric data to the stored authentication template and will return a success or failure result.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.3.7 Full Residual Information Protection (FDP_RIP.2)

### TSS Assurance Activities

#### Assurance Activity AA-FDP_RIP.2-ASE-01

***Requirement met by the platform***

*The evaluator shall examine the TSS to verify that it describes (for each supported platform) the extent to which the client processes network packets and addresses the FDP_RIP.2 requirement.*

***Requirement met by the TOE***

*"Resources" in the context of this requirement are network packets being sent through (as opposed to "to", as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs.*

#### Summary

Section 8.9.4.2 *IPsec General* in the TSS of [ST] states the IPsec is implemented in the TOE natively, as part of the TOE OS, hence the packets are proccessed by the TOE. There is no separate client application that process packrets specifically for the VPN.

Section 8.9.4.6 *Residual information protection and packet processing* of the TSS states that when a network packet is received, the TCP/IP stack allocates a new buffer in memory of the same size as the incoming packet, then copies the packet into the new buffer, thereby, overwriting the entire allocated buffer. The packet is only referred to by reference/address, not copied. The VPN encrypts and decrypts the packets in-place because the size of the data does not change when using symmetric ciphers.

### Guidance Assurance Activities

#### Assurance Activity AA-FDP_RIP.2-AGD-01

*There are no AGD EAs for this requirement.*

#### Summary

There are no AGD EAs for this requirement.

### Test Assurance Activities

#### Assurance Activity AA-FDP_RIP.2-ATE-01

*There are no test EAs for this requirement.*

#### Summary

No test EAs specified by [MOD_VPNC_V2.3].

## 2.2.3.8 User Data Storage (FDP_STG_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FDP_STG_EXT.1-ASE-01

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

#### Summary

Section 8.5.2 *X.509v3 Certificates* in the TSS of [ST] describes how X.509 certificates are used and managed in the TOE. It states the following:

- The Apple certificate which is used for trusted update is installed in ROM during manufacturing. Other certificates used for trusted channels can be imported by a user (if allowed by the policy) or installed using Configuration Profiles.
- Every certificate has a certificate type which defines what the certificate is used for. This ensures that only certificates defined for a specific application can be used for this application, and nothing else.
- Code signing certificates need to be assigned by Apple and can be imported into a TOE device.
- 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 either when reviewed and accepted by the user, using a Configuration Profile, or download from a web site.
- The database containing trust anchors for all certificates is protected via integrity check and write protection.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.3.9 Inter-TSF User Data Transfer Protection (Applications) (FDP_UPC_EXT.1/APPS)

### TSS Assurance Activities

#### Assurance Activity AA-FDP_UPC_EXT.1-APPS-ASE-01

*The evaluator shall examine the TSS to determine that it describes that all protocols listed in the TSS are specified and included in the requirements in the ST.*

#### Summary

Section 8.9 *Trusted Path/Channels (FTP)* in the TSS of [ST] is divided into several section where each section describes the protocol supported by the TOE includes EAP-TLS and TLS, HTTPS, Bluetooth, Wireless LAN, and VPN (IPsec). The evaluator verified that these protocols are included in the requirements in the ST (that is the ST does include FCS_HTTPS_EXT.1, FCS_IPSEC_EXT.1, FCS_TLSC_EXT.1, and FIA_BLT_EXT.*).

### Guidance Assurance Activities

#### Assurance Activity AA-FDP_UPC_EXT.1-APPS-AGD-01

*The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocol(s) selected for use by the applications.*

#### Summary

[CCGUIDE] provides guidance for the following:

- TLS/EAL-TLS: [CCGUIDE] sections 5.3.1, 5.3.2, 5.3.6
- HTTPS: section 5.3.2
- IPsec: [CCGUIDE] sections 5.3.3, 5.3.5, and 5.3.6

These protocols can be configured via a Configuration Profile described in [DEV_MAN] .

While performing guidance assurance activities pertaining to these protocols, the evaluator also verified that sufficient guidance is provided. This was also supported by the related test assurance activities.

### Test Assurance Activities

#### Assurance Activity AA-FDP_UPC_EXT.1-APPS-ATE-01

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

*The evaluator shall write, or the developer shall provide access to, an application that requests protected channel services by the TSF. The evaluator shall verify that the results from the protected channel match the expected results according to the API documentation. This application may be used to assist in verifying the protected channel Evaluation Activities for the protocol requirements. The evaluator shall also perform the following tests:*

- ***Test 1:*** *The evaluators shall ensure that the application is able to initiate communications with an external IT entity using each protocol specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 2:*** *The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext.*

#### Summary

For both tests, TLS/HTTPS is tested implicitly with all EAP-TLS and HTTPS tests.

## 2.2.3.10 Inter-TSF User Data Transfer Protection (Bluetooth) (FDP_UPC_EXT.1/BLUETOOTH)

### TSS Assurance Activities

#### Assurance Activity AA-FDP_UPC_EXT.1-BLUETOOTH-ASE-01

*The evaluator shall examine the TSS to determine that it describes that all protocols listed in the TSS are specified and included in the requirements in the ST.*

**Summary**

Section 8.9 *Trusted Path/Channels (FTP)* in the TSS of [ST] is divided into several section where each section describes the protocol supported by the TOE which are EAP-TLS and TLS, HTTPS, Bluetooth, Wireless LAN, and VPN (IPsec). The evaluator verified that these protocols are included in the requirements in the ST (that is the ST does include FCS_HTTPS_EXT.1, FCS_IPSEC_EXT.1, FCS_TLSC_EXT.1, and FDP_UPC_EXT.1/BLUETHOOTH, FIA_BLT_EXT.*, and FIA_BMG_EXT.*).

## Guidance Assurance Activities

### Assurance Activity AA-FDP_UPC_EXT.1-BLUETOOTH-AGD-01

*The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocol(s) selected for use by the applications.*

**Summary**

[CCGUIDE] provides guidance for the following:

- Bluetooth: [CCGUIDE] sections 5.3.4

While performing guidance assurance activities pertaining to this protocol, the evaluator also verified that sufficient guidance is provided. This was also supported by the related test assurance activities.

## Test Assurance Activities

### Assurance Activity AA-FDP_UPC_EXT.1-BLUETOOTH-ATE-01

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

*The evaluator shall write, or the developer shall provide access to, an application that requests protected channel services by the TSF. The evaluator shall verify that the results from the protected channel match the expected results according to the API documentation. This application may be used to assist in verifying the protected channel Evaluation Activities for the protocol requirements. The evaluator shall also perform the following tests:*

- ***Test 1:*** *The evaluators shall ensure that the application is able to initiate communications with an external IT entity using each protocol specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 2:*** *The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext.*

**Summary**

For Test 1, the evaluator used the BlueZ software to pair the test laptop with the TOE.

For Test 2, the evaluator ensured that the TOE was disconnected and then

- invoked hcidump to capture the traffic,
- initiated the Bluetooth connection,
- opened the pcap file and inspected the traffic.

The evaluator verified that Link Level Encryption was ON and the connection was encrypted.

# 2.2.4 Identification and authentication (FIA)

## 2.2.4.1 Authentication Failure Handling (FIA_AFL_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_AFL_EXT.1-ASE-01

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

*If the TOE supports multiple authentication mechanisms, the evaluator shall ensure that this description also includes how the unsuccessful authentication attempts for each mechanism selected in FIA_UAU.5.1 is handled. The evaluator shall verify that the TSS describes if each authentication mechanism utilizes its own counter or if multiple authentication mechanisms utilize a shared counter. If multiple authentication mechanisms utilize a shared counter, the evaluator shall verify that the TSS describes this interaction.*

*The evaluator shall confirm that the TSS describes how the process used to determine if the authentication attempt was successful. The evaluator shall ensure that the counter would be updated even if power to the device is cut immediately following notifying the TOE user if the authentication attempt was successful or not.*

**Summary**

Section 8.5 *Identification and Authentication (FIA)* in the TSS of [ST] describes the different authentication mechanisms supported by the TOE including a passcode, fingerprint (Touch ID), or facial authentication (Face ID) of which the latter two are biometric mechanisms. Also, this section states the following:

- TOE devices that support Touch ID do not support Face ID and vice versa.
- The passcode and the device's biometric authentication method cannot be combined for two-factor authentication.
- For passcode authentication, the maximum number of consecutive failed attempts is between 2 and 11, where 11 is the default.
- For biometric authentication, if three invalid biometric samples are presented the TOE will offer passcode entry. After five invalid biometric samples are presented passcode authentication is required.
- The number of failed passcode 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.
- Touch ID and Face ID use a separate failed attempt counter from the passcode counter that is not maintained over a power loss or reboot (i.e., the Touch ID and Face ID failed attempt counter is reset after a power loss or reboot).

For Touch ID, 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 fingerprint samples are presented, then 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

Devices that support Touch ID do not support Face ID and vice versa. The passcode and the device's biometric authentication method cannot be combined for two-factor authentication.

The number of failed passcode 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.

Touch ID and Face ID use a separate failed attempt counter from the passcode counter that is not maintained over a power loss or reboot (i.e., the Touch ID and Face ID failed attempt counter is reset after a power loss or reboot).

Entering the same incorrect passcode multiple times consecutively causes the passcode failed login counter to increment only once for those multiple attempts even though these are all passcode failed login attempts. Different passcodes must be entered in order for the passcode failed login counter to increment. (This description only applies to the passcode failed login counter and does not apply to biometric authentication.)

In addition, the following behavior applies to biometric authentication methods. A passcode must be supplied for additional security validation under any of the following conditions.

- 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/Medical ID.

## Guidance Assurance Activities

### Assurance Activity AA-FIA_AFL_EXT.1-AGD-01

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

### Summary

[CCGUIDE] section 5.5.4 *Authentication Attempt Configuration* provides guiidance for setting the authentication attempt. It states that in the evaluated configuration the TOE users are not allowed configure the maximum number of failed authentication attempts. The TOE administrator is responsible for performing this function.

The administrator performs this configuration via a Configuration Profile by specifying in the Passcode Payload the *maxFailedAttempts* key with the value between 2 and 11. Additional detail of the Passcode Policy Payload is provided in [CCGUIDE] section 5.5.1 *Passcode Authentication Configuration* . Also, a sample Configuration Profile for configuring passcode restrictions is provided in *Appendix: Configuration Profiles* of [CCGUIDE] .

### Test Assurance Activities

#### Assurance Activity AA-FIA_AFL_EXT.1-ATE-01

- ***Test 1:*** *The evaluator shall configure the device with all authentication mechanisms selected in FIA_UAU.5.1. The evaluator shall perform the following tests for each available authentication interface:*
  - *Test 1a: The evaluator shall configure the TOE, according to the AGD guidance, with a maximum number of unsuccessful authentication attempts. The evaluator shall enter the locked state and enter incorrect passwords until the wipe occurs. The evaluator shall verify that the number of password entries corresponds to the configured maximum and that the wipe is implemented.*
  - *Test 1b: [conditional] If the TOE supports multiple authentication mechanisms the previous test shall be repeated using a combination of authentication mechanisms confirming that the critical authentication mechanisms will cause the device to wipe and that when the maximum number of unsuccessful authentication attempts for a non-critical authentication mechanism is exceeded, the device limits authentication attempts to other available authentication mechanisms. If multiple authentication mechanisms utilize a shared counter, then the evaluator shall verify that the maximum number of unsuccessful authentication attempts can be reached by using each individual authentication mechanism and a combination of all authentication mechanisms that share the counter.*
- ***Test 2:*** *The evaluator shall repeat test one, but shall power off (by removing the battery, if possible) the TOE between unsuccessful authentication attempts. The evaluator shall verify that the total number of unsuccessful authentication attempts for each authentication mechanism corresponds to the configured maximum and that the critical authentication mechanisms cause the device to wipe. Alternatively, if the number of authentication failures is not maintained for the interface under test, the evaluator shall verify that upon booting the TOE between unsuccessful authentication attempts another authentication factor interface is presented before the interface under test.*

#### Summary

Test 1: The evaluator set the maximum number of failed authentication attempts to 3 and provided three different incorrect passcode. The tester unlocked the device and verified that the data was wiped.

Test 2: The evaluator tried to unlock the device with the wrong password twice, rebooted the device, and tried to unlock the device with a third wrong password. The evaluator then unlocked the device and verified that the data was wiped.

## 2.2.4.2 Bluetooth User Authorization (FIA_BLT_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.1-ASE-01

*The evaluator shall examine the TSS to ensure that it contains a description of when user permission is required for Bluetooth pairing; and that this description mandates explicit user authorization via manual input for all Bluetooth pairing; including application use of the Bluetooth trusted channel and situations where temporary (non-bonded) connections are formed.*

#### Summary

Section 8.9.2 *Bluetooth* in the TSS of [ST] describes the Bluetooth protocol.

The evaluator verified that this section specifically addresses how a user authorizes Bluetooth pairing through the Bluetooth interface in the settings of the TOE OS. This section also states that during the pairing time, another device (or the TOE OS) 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.

## Guidance Assurance Activities

### Assurance Activity AA-FIA_BLT_EXT.1-AGD-01

*The evaluator shall examine the API documentation provided as a means of satisfying the requirements for the ADV assurance class (see section 5.2.2 in the MDF PP and GPOS PP) and verify that this API documentation does not include any API for programmatic entering of pairing information (e.g. PINs; numeric codes; or "yes/no" responses) intended to bypass manual user input during pairing.*

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

#### Summary

The evaluator examined the provided API documentation for Bluetooth [COREBT] (Core Bluetooth framework) to ensure that there is no means for bypassing manual user input during pairing.

[CCGUIDE] section 5.3.4 *Bluetooth Configuration* provides related guidance for configuring Bluetooth which references the related guidance from the iPhone user guide [iPhone_UG] for how to turn Bluetooth on and off and how to pair and un-pair a Bluetooth device. This involves turning on the Bluetooth feature on the TOE device (Settings>> Bluetooth), choose the Bluetooth accessory, and then enter a passkey or PIN if prompted. When pairing is complete, user can use the Bluetooth accessory with their device. Bluetooth can also be disassociated via the Control Center.

Additionally, [CCGUIDE] section 5.3.4 states that prior to the pairing the mobile device must first be discovered. Two conditions must be met for the mobile device to become discoverable: Bluetooth must be enabled, and the Bluetooth configuration panel must be both active and in the foreground. If the Bluetooth configuration panel is not the active panel, or if Bluetooth is disabled, the mobile device is not discoverable. There is no other method to make the mobile device discoverable or not discoverable.

## Test Assurance Activities

### Assurance Activity AA-FIA_BLT_EXT.1-ATE-01

*The evaluator shall perform the following steps:*

*Step 1: Initiate pairing with the TOE from a remote Bluetooth device that requests no man-in-the-middle protection; no bonding; and claims to have NoInput/NoOutput (IO) capability. Such a device will attempt to evoke behavior from the TOE that represents the minimal level of user interaction that the TOE supports during pairing.*

*Step 2: Verify that the TOE does not permit any Bluetooth pairing without explicit authorization from the user (e.g. the user must have to minimally answer "yes" or "allow" in a prompt).*

#### Summary

Step 1: The evaluator used the BlueZ software to pair the TOE and the test laptop. The evaluator verified that during the process, a PIN was displayed and the user is required to approved the PIN. The evaluator verified that the same PIN is shown no the remote Bluetooth test laptop.

Step 2: The evaluator ensured that the TOE was disconnected from the test laptop, invoked hcidump to capture the Bluetooth traffic, and initiated a Bluetooth connection between the TOE and the test laptop. The evaluator then opened the pcap file in Wireshark and search for string: "Encryption Change" and verified that HCI event showed that Link Level Encryption was ON (0x01) and that the command "hcitool con" showed the connection as being encrypted.

## 2.2.4.3 Bluetooth Mutual Authentication (FIA_BLT_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.2-ASE-01

*The evaluator shall ensure that the TSS describes how data transfer of any type is prevented before the Bluetooth pairing is completed. The TSS shall specifically call out any supported RFCOMM and L2CAP data transfer mechanisms. The evaluator shall ensure that the data transfers are only completed after the Bluetooth devices are paired and mutually authenticated.*

#### Summary

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- The TOE 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 Bluetooth Specifications [BT] for details. The TOE generates a new ephemeral ECDH key pair for every new connection attempt. No data can be transferred via Bluetooth until pairing has been completed. The TOE terminates the connection if the remote device stops encryption while connected to the TOE.
- The TOE supports the Logical Link Control and Adaptation Layer Protocol (L2CAP) through an API in the IOBluetoothDevice class.
- An RFCOMM channel object can be obtained by opening an RFCOMM channel in a device, or by requesting a notification when a channel is created (this is commonly used to provide services). See the IOBluetooth RFCOMMChannel class.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.2-ATE-01

*The evaluator shall use a Bluetooth tool to attempt to access TOE files using the OBEX Object Push service (OBEX Push) and verify that pairing and mutual authentication are required by the TOE before allowing access. If the OBEX Object Push service is unsupported on the TOE; a different service that transfers data over Bluetooth L2CAP and/or RFCOMM may be used in this test.*

#### Summary

Test 1: The test performed for FIA_BLT_EXT.1 demonstrates this functionality as well: data can only be communicated once a connection is established. Such connection can only be established when devices are either manually or automatically paired. Automatic pairing can only be enabled on the TOE after an initial manual pairing was successful. The evaluator started the Bluetooth configuration on the TOE to make the device visible, then made another device visible to the TOE. The evaluator started the pairing process without confirming the PIN and verified that the TOE did not share files (Photos). The evaluator repeated the process but confirmed the PIN and verified that the device was in a "Connected" state.

## 2.2.4.4 Rejection of Duplicate Bluetooth Connections (FIA_BLT_EXT.3)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.3-ASE-01

*The evaluator shall ensure that the TSS describes how Bluetooth sessions are maintained such that at least two devices with the same Bluetooth device address are not simultaneously connected and such that the initial session is not superseded by any following session initialization attempts.*

**Summary**

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- Connections via BR/EDR and LE are secured using 128-bit AES Counter with CBC-MAC (AES-CCM-128) mode. No other key sizes are supported; thus, smaller key sizes cannot be negotiated. 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 (i.e., pairing and session initialization attempts) from the same BD_ADDR for an established connection will be discarded.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.3-ATE-01

*The evaluator shall perform the following steps:*

*Step 1: Pair the TOE with a remote Bluetooth device (DEV1) with a known address BD_ADDR. Establish an active session between the TOE and DEV1 with the known address BD_ADDR.*

*Step 2: Attempt to pair a second remote Bluetooth device (DEV2) claiming to have a Bluetooth device address matching DEV1 BD_ADDR to the TOE. Using a Bluetooth protocol analyzer, verify that the pairing attempt by DEV2 is not completed by the TOE and that the active session to DEV1 is unaffected.*

*Step 3: Attempt to initialize a session to the TOE from DEV2 containing address DEV1 BD_ADDR. Using a Bluetooth protocol analyzer, verify that the session initialization attempt by DEV2 is ignored by the TOE and that the initial session to DEV1 is unaffected.*

**Summary**

Step 1: The evaluator paired the TOE with another device and obtained the Bluetooth ID of the remote device.

Step 2: The evaluator set the Bluetooth dongle ID to the ID of the remote device, scanned the TOE, started the Bluetooth sniffer on the Linux test laptop, tried to connect to the TOE with the Bluetooth dongle and verified that the connection fails. The evaluator also verified that the Bluetooth connection remains unaltered.

Step 3: The evaluator verified with Wireshark that the connection failed.

## 2.2.4.5 Secure Simple Pairing (FIA_BLT_EXT.4)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.4-ASE-01

*The evaluator shall verify that the TSS describes the secure simple pairing process.*

#### Summary

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- The TOE 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.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.4-ATE-01

*The evaluator shall perform the following steps:*

*Step 1: Initiate pairing with the TOE from a remote Bluetooth device that supports Secure Simple Pairing.*

*Step 2: During the pairing process; observe the packets in a Bluetooth protocol analyzer and verify that the TOE claims support for both "Secure Simple Pairing (Host Support)" and "Secure Simple Pairing (Controller Support)" during the LMP Features Exchange.*

*Step 3: Verify that Secure Simple Pairing is used during the pairing process.*

#### Summary

Step 1: On the TOE and test laptop, the evaluator removed the paired devices. The evaluator then started the Bluetooth sniffer, initiated and completed the pairing of the TOE with the test laptop.

Step 2: The evaluator opened the pcap file obtained in Step 1 with Wireshark and searched for the packet "Read Remote Supported Features" and verified that the option for "Secure Simple Pairing" is set to True. The evaluator then opened the pcap file with Wireshark and searched for the packet "Read Remote Extended Features Complete" and verified that the option for "Secure Simple Pairing Host" is set to True.

Step 3: The evaluator searched in the pcap file for the packet "Encryption Change" which marks the successful Bluetooth connection.

## 2.2.4.6 Trusted Bluetooth Device User Authorization (FIA_BLT_EXT.6)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.6-ASE-01

*The evaluator shall verify that the TSS describes all Bluetooth profiles and associated services for which explicit user authorization is required before a remote device can gain access. The evaluator shall also verify that the TSS describes any difference in behavior based on whether or not the device has a trusted relationship with the TOE for that service*

*(i.e. whether there are any services that require explicit user authorization for untrusted devices that do not require such authorization for trusted devices). The evaluator shall also verify that the TSS describes the method by which a device can become 'trusted'.*

**Summary**

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- The TOE automatically authorizes the remote Bluetooth device during pairing for all Bluetooth profiles the remote device announces to support during the pairing operation. This approach avoids user confusion between a paired device to which the TOE is connected to but not yet authorized device with which the TOE cannot yet communicate. To de-authorize a device, the user would unpair the device. The TOE establishes a “trusted relationship” with an authorized device at the time of pairing. The only difference in behavior between a trusted device and an untrusted device is that the untrusted device must first be manual authorized.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.6-ATE-01

*The evaluator shall perform the following tests:*

- ***Test 1:*** *While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a "protected" Bluetooth service (as specified in the assignment in FIA_BLT_EXT.6.1) from a "trusted" remote device. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.*
- ***Test 2:*** *The evaluator shall repeat Test 1, this time allowing the authorization and verifying that the remote device successfully accesses the service.*

**Summary**

Test 1: The evaluator made a trusted Bluetooth device visible to the TOE and started the pairing procedure but did not provide the proper PIN and observed the pairing did not complete and the device could not access any TOE data.

Test 2: The evaluator repeated Test 1 but then provided the proper PIN and observed the pairing did complete and the device could access the shared TOE data.

## 2.2.4.7 Untrusted Bluetooth Device User Authorization (FIA_BLT_EXT.7)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.7-ASE-01

*The TSS evaluation activities for this component are addressed by FIA_BLT_EXT.6.*

**Summary**

This evaluation activity was performed in conjunctionw with AA-FIA_BLT_EXT.6-ASE-01 .

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_BLT_EXT.7-ATE-01

*The evaluator shall perform the following tests if the TSF differentiates between "trusted" and "untrusted" devices for the purpose of granting access to services. If it does not, then the test evaluation activities for FIA_BLT_EXT.6 are sufficient to satisfy this component.*

- ***Test 1:*** *While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a "protected" Bluetooth service (as specified in the assignment in FIA_BLT_EXT.7.1) from an "untrusted" remote device. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.*
- ***Test 2:*** *The evaluator shall repeat Test 1, this time allowing the authorization and verifying that the remote device successfully accesses the service.*
- ***Test 3:*** *(conditional): If there exist any services that require explicit user authorization for access by untrusted devices but not by trusted devices (i.e. a service that is listed in FIA_BLT_EXT.7.1 but not FIA_BLT_EXT.6.1), the evaluator shall repeat Test 1 for these services and observe that the results are identical. That is, the evaluator shall use these results to verify that explicit user approval is required for an untrusted device to access these services, and failure to grant this approval will result in the device being unable to access them.*
- ***Test 4:*** *(conditional): If test 3 applies, the evaluator shall repeat Test 2 using any services chosen in Test 3 and observe that the results are identical. That is, the evaluator shall use these results to verify that explicit user approval is required for an untrusted device to access these services, and granting this approval will result in the device being able to access them.*
- ***Test 5:*** *(conditional): If test 3 applies, the evaluator shall repeat Test 3 except this time designating the device as "trusted" prior to attempting to access the service. The evaluator shall verify that access to the service is granted without explicit user authorization (because the device is now trusted and therefore FIA_BLT_EXT.7.1 no longer applies to it). That is, the evaluator shall use these results to demonstrate that the TSF will grant a device access to different services depending on whether or not the device is trusted.*

#### Summary

Test 1: The evaluator made an untrusted Bluetooth device visible to the TOE and started the pairing procedure but did not provide the proper PIN and observed the pairing did not complete and the device could not access any TOE data.

Test 2: The evaluator repeated Test 1 but then provided the proper PIN and observed the pairing did complete and the device could access the shared TOE data.

Tests 3, 4, and 5 are not applicable since authorization is required for all services on both trusted and untrusted Bluetooth devices.

## 2.2.4.8 Accuracy of Biometric Authentication (FIA_BMG_EXT.1)

### FIA_BMG_EXT.1.1

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BMG_EXT.1.1-ASE-01

*The evaluator shall verify that the TSS contains evidence supporting the testing and calculations completed to determine the FAR and FRR. Section G Biometric Derivation and Examplesin* [MDFPPv3.2] *provides guidance to how this testing could be completed and to what error bars are expected when the Rule of 3 is applied. The evaluator shall consult Section G Biometric Derivation and Examplesin* [MDFPPv3.2] *as a reference, but should not treat it as a mandate. The*

*evaluator shall verify that the TSS contains evidence of whether online or offline testing was used. If offline testing was completed, evidence describing the differences between the biometric system used for testing and the TOE in the evaluated configuration, if any must be included.*

*The following documentation is not required to be part of the TSS - it may be submitted as a separate proprietary document. The evaluator shall verify the evidence includes how many imposters were used for testing and that the testing describes how imposters are compared to enrolled users, for example, if multiple devices for online testing or full cross-comparison for offline testing was used. Adequate documentation is required to demonstrate that testing was completed to support the claimed FAR and FRR.*

**Summary**

The evaluator reviewed section 8.5.1 *Biometric Authentication* of [ST] which describes biometric authentication. Specifically, sections 8.5.1.1 and 8.5.1.3 explain the accuracy of biometric authentication of the TOE. The evaluator could find in these sections evidence of the calculation used to determine the FAR and FRR for the TOE. The evaluator verified that: the testing was performed offline; that the system used for testing is emulated on a different platform in a cloud computation infrastructure for efficiency reasons; a full cross-comparison for offline testing is used to compare imposters for Gen.1 sensors, and a partial full-cross comparison is used for Gen.2 and Gen.3 sensors.

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 datasets. The datasets 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).

**Guidance Assurance Activities**

No assurance activities defined.

**Test Assurance Activities**

No assurance activities defined.

## FIA_BMG_EXT.1.2

**TSS Assurance Activities**

**Assurance Activity AA-FIA_BMG_EXT.1.2-ASE-01**

*The evaluator shall verify that the TSS indicates which SAFAR the TOE is targeting and contains evidence supporting the calculations, per Section G.3 SAFAR Calculation Equations in* [MDFPPv3.2] *, completed to determine the SAFAR. The evaluator shall verify that the TSS contains evidence of how the authentication factors interact, per FIA_UAU.5.2 and FIA_AFL_EXT.1. The evaluator shall verify that the TSS, contains the combination(s) of authentication factors needed to meet the SAFAR, and the number of attempts for each authentication factor the TOE is configured to allow. Adequate documentation is required to demonstrate the calculations completed to support the claimed SAFAR.*

**Summary**

The evaluator reviewed section 8.5.1 *Biometric Authentication* of [ST] which describes biometric authentication of the TOE. The TOE supports fingerprint (Touch ID) or facial recognition authentication (Face ID) Specifically, sections 8.5.1.1 and 8.5.1.3 explain the accuracy of biometric authentication of the TOE. The SAFAR that the TOE is targeting is described in section 8.5.1.3.

Section 8.5 describes how the Face ID and Touch ID interact with the passcode set on the TOE device. A TOE device that supports Face ID does not support Touch ID and vice-versa. A passcode must be supplied in the following cases for additional security.

- 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.
- There have been five unsuccessful attempts to match.
- After initiating power off/Emergency SOS/Medical ID.

Section 8.5.1.3 describes that there can be at most only 5 attempts for the fingerprint and 5 attempts for the facial recognition. Past these attempts, at most 10 attempts for the 6-digit passcode are provided.

#### Guidance Assurance Activities

No assurance activities defined.

#### Test Assurance Activities

No assurance activities defined.

## 2.2.4.9 Biometric Enrollment (FIA_BMG_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BMG_EXT.2-ASE-01

> *The evaluator shall verify that the TSS describes how the quality of samples used to create the authentication template at enrollment are verified. As well as the quality standard that the validation method uses to perform the assessment.*

#### Summary

Section 8.5.1.4 *Biometric Sample Quality* of the TSS describes the biometric sample quality used on the TOE for deriving the authentication template enrolled. This includes 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

Moreover, the testing mechanisms are repeated for every major TOE OS release. Section 8.5.1.1 describe the tests subjects chosen for the samples: various people were used to collect facial expressions with various eye wear, indoor/outdoor settings, age, ethnicity and gender. Family members were also chosen, such as siblings.

The method used to perform this assessment is an off-line test. Fingerprint data (Touch ID) were collected separately for each sensor. The data was passed to an emulator for efficiency reasons, and not on the production hardware. Note that a special testing step is performed in order to

guarantee that the results on the emulator match the results on the production hardware. For Gen.1 sensors, a full-cross comparison scheme is used. For Gen.2 and Gen.3 sensors, a partial cross-comparison is used.

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

If the sample quality passes verification during enrollment, the TOE saves it as an enrollment template. When a user authenticates, an authentication template is generated. If a properly formatted template contains unusual data properties, incorrect syntax, low quality, or unrealistic modality, the TOE rejects the template.

The validation of the discussed mechanism is performed regularly for each major TOE OS 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.

## Guidance Assurance Activities

### Assurance Activity AA-FIA_BMG_EXT.2-AGD-01

| |
|---|
| *The evaluator shall verify that the AGD guidance describes how to enroll a user for each biometric modality supported.* |

### Summary

[CCGUIDE] section 5.5.3 *Biometric Authentication Factors* describes how to enroll biometric authentication factors which is by using the device UI as follows:

- Face ID Device: Settings » Face ID & Passcode
- Touch ID Device: Settings » Touch ID & Passcode

This section also states that in the evaluated configuration, users cannot enable Touch ID or Face ID on their devices. This restriction must be performed by the administrator using the Restriction Payload in a Configuration Profile. If enabled, the user can enroll Touch ID or Face ID as follows:

- Enrollment for Touch ID is typically accomplished during initial device configuration can also be performed using the Settings » Touch ID & Passcode menu. Multiple fingerprints may be enrolled, named, and deleted from this menu.
- Enrollment for Face ID is typically accomplished during initial device configuration but can also be performed using the Settings » Face ID & Passcode menu by tapping the "Set up Face ID" option. Users can enroll an alternative appearance for Face ID, for a total of two enrollments per device. Face ID credential is established by providing biometric samples for enrollment.

### Test Assurance Activities

#### Assurance Activity AA-FIA_BMG_EXT.2-ATE-01

*The evaluator shall input biometric samples for enrollment. Upon inputting biometric samples a fixed number of times as specified in the prompts, one or more authentication templates will be generated. The evaluator shall verify that the device only accepts samples of sufficient quality or requests additional samples if the authentication template is not of sufficient quality. For all quality metrics, the evaluator shall ensure that biometric samples achieving a worse quality score than the prescribed threshold are rejected.*

**Summary**

The evaluator enrolled a fingerprint and a face into the device and attempted to unlock the device with another finger and with the nose and mouth covered and verified that both attempts failed. The evaluator then unlocked the device with the correct finger or full face and verified that it succeeds.

## 2.2.4.10 Biometric Verification (FIA_BMG_EXT.3)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BMG_EXT.3-ASE-01

*The evaluator shall verify that the TSS describes how the quality of samples used to verify authentication are verified. As well as the quality standard that the validation method uses to perform the assessment. The evaluator shall enroll a user for each biometric modality supported. The evaluator will then input biometric samples for verification and ensure that the device only accepts samples of sufficient quality. The evaluator shall ensure that biometric samples achieving a worse quality score than the prescribed threshold are rejected.*

**Summary**

Section 8.5.1.4 *Biometric Sample Quality* of the TSS describes the biometric sample quality used on the TOE for deriving the authentication template enrolled. This includes 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

Moreover, the TSS describe that 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.

Section 8.5.1.4 *Biometric Sample Quality* of the TSS describes the biometric sample quality used on the TOE for deriving the authentication template enrolled. This includes:

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

Moreover, the TSS describe that 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.

As part of independent testing, the evaluator will enroll a user for each biometric modality supported. The evaluator will then input biometric samples for verification and ensure that the device only accepts samples of sufficient quality. The evaluator will ensure that biometric samples achieving a worse quality score than the prescribed threshold are rejected.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.4.11 Handling Unusual Biometric Templates (FIA_BMG_EXT.5)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_BMG_EXT.5-ASE-01

*The evaluator shall verify that the TSS how the matching algorithm addresses properly formatted templates with unusual data properties, incorrect syntax, or low quality. The evaluator shall ensure that these claims are sound through appropriate testing based on test programs provided by the vendor.*

**Summary**

Section 8.5.1.4 *Biometric Sample Quality* of the TSS describes the biometric sample quality used on the TOE for deriving the authentication template enrolled. It states that if the sample quality passes verification during enrollment, the TOE saves it as an enrollment template. When a user authenticates, an authentication template is generated. If a properly formatted template contains unusual data properties, incorrect syntax, low quality, or unrealistic modality, the TOE rejects the template.

As part of independent testing, the evaluator will ensure that these claims are sound through appropriate testing based on test programs provided by the vendor.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.4.12 Agent Enrollment of Mobile Device into Management (FIA_ENR_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_ENR_EXT.2-MDMA-ASE-01

*The evaluator shall examine the TSS to verify that it describes which types of reference identifiers are acceptable and how the identifier is specified (e.g. preconfigured in the MDM Agent, by the user, by the MDM server, in a policy).*

**Summary**

Section 8.5.3 *MDM Server Reference ID* in the TSS of [ST] provides table 12 "MDM Server Reference Identifiers" describes the reference identifiers used by the MDM server. They can be the following:

- serve_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

The evaluator verified which type of reference identifiers are acceptable to the TOE, and they are configured by the MDM enrollment service through JavaScript Object Notation (JSON) tags on the mobile device.

### Guidance Assurance Activities

#### Assurance Activity AA-FIA_ENR_EXT.2-MDMA-AGD-01

> *The evaluator shall examine the operational guidance to verify that it describes how to configure reference identifier of the MDM Server's certificate and, if different than the reference identifier, the Domain Name or IP address (for connectivity) of the MDM Server.*

**Summary**

[CCGUIDE] section 4.3.3 *Configure MDM Agent and MDM Communications* describes how to configure MDM Agent and MDM Server communications as follows:

- MDM Agent-Server communication is achieved securely using the MDM protocol which is built on top of HTTP, transport layer security (TLS), and push notifications that use HTTP PUT over TLS (secure sockets layer (SSL)). A managed mobile device uses an identity to authenticate itself to the MDM server over TLS (SSL). This identity can be included in the profile as a Certificate Payload or can be generated by enrolling the mobile device with Simple Certificate Enrollment Protocol (SCEP).
- The MDM Agent communications uses the TOE OS Security Framework as described in section 5.3.2 *TLS Configuration* . Configuring the device's TLS protocol automatically configures the MDM Agent communications. If an additional CA certificate needs to be added to support the MDM Server, see subsection 5.3.2.3.

The evaluator examined these sections and determined that they provided sufficient guidance.

### Test Assurance Activities

#### Assurance Activity AA-FIA_ENR_EXT.2-MDMA-ATE-01

> *The evaluator shall follow the operational guidance to establish the reference identifier of the MDM server on the MDM Agent and in conjunction with other evaluation activities verify that the MDM Agent can connect to the MDM Server and validate the MDM Server's certificate.*

**Summary**

The evaluator went on the TOE's
`Settings -> General -> Device Management -> Remote Management -> More Details`
and verified that the Device Identity Certificate contains the issuers certificate. In Device Management, the evaluator verified that the URL indeed points to the OSX server.

## 2.2.4.13 Port Access Entity Authentication (FIA_PAE_EXT.1)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_PAE_EXT.1-WLAN-ATE-01

*The evaluator shall perform the following tests:*

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

**Summary**

Test 1: The evaluator removed the profile from the TOE containing the WiFi information and verified that the device could not connect to the network. The evaluator then restored the profile to the TOE and verified that the device could connect to the network by witnessing the EAPOL 4-way handshake using Wireshark.

Test 2: The evaluator removed the profile from the TOE containing the WiFi information and verified that the device could not connect to the network. The evaluator then created a similar profile but with a wrong certificate, loaded it on the TOE, and verified that the TOE could not connect to the network.

Test 3: The evaluator removed the profile from the TOE containing the WiFi information and verified that the device could not connect to the network. The evaluator then created an incorrect certificate for the server and verified that the TOE could not connect to the network.

## 2.2.4.14 Password Management (FIA_PMG_EXT.1)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

#### Assurance Activity AA-FIA_PMG_EXT.1-AGD-01

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

**Summary**

[CCGUIDE] section 5.5.1 *Passcode Authentication Configuration* provides guidance to administrators on the composition of strong passcode which consists of the following:

- Passcode composing of any combination of upper and lower case letters, numbers, and special characters: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)”;
- Passcode length must be between 1 and 16 characters

Passcode policy is defined by administrator using the Passcode Payload in a Configuration Profile described in [DEV_MAN] . The Passcode Payload presents the administrator with an alphanumeric passcode entry mechanism, which allows for the entry of arbitrarily long and complex passcodes including the selection of special characters. The administrator must set the configuration keys *allowSimple* to 'false'; *requireAlphanumberic* to 'true' and *minLength* to a value defined by the organization's policy (i.e., greater than zero). Also, a sample Configuration Profile for configuring passcode restrictions is provided in *Appendix: Configuration Profiles* of [CCGUIDE] .

### Test Assurance Activities

#### Assurance Activity AA-FIA_PMG_EXT.1-ATE-01

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

#### Summary

The evaluator configured the password policy rule using the Apple Configurator 2 to require at least 4 characters and at least 1 complex character. The evaluator loaded the profile onto the TOE, and attempted to change the passcode (or waited until the change password request appeared). The evaluator set the passcode to "1234" and verified that it is rejected. The evaluator set the password to a more complex value (e.g. "zaq1@") and verified that it is accepted. The evaluator then locked the TOE and unlocked it with the new passcode.

## 2.2.4.15 Authentication Throttling (FIA_TRT_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_TRT_EXT.1-ASE-01

*The evaluator shall verify that the TSS describes the method by which authentication attempts are not able to be automated. The evaluator shall ensure that the TSS describes either how the TSF disables authentication via external interfaces (other than the ordinary user interface) or how authentication attempts are delayed in order to slow automated entry and shall ensure that this delay totals at least 500 milliseconds over 10 attempts for all authentication mechanisms selected in FIA_UAU.5.1.*

#### Summary

Section 8.5 *Identification and Authentication (FIA)* in the TSS of the [ST] describes the user identification and authentication. Per this section all authentication mechanisms require authentication data (passcode or biometric samples) to be entered by the TOE user.

Also, this section states that the time between consecutive authentication attempts, including biometric authentication factors, is at least the time it takes the PBKDF2 function to execute which is calibrated to be at least 80 milliseconds between consecutive attempts. In addition, for Touch ID, the TOE enforces a delay of 5 seconds between repeated failed authentication.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.4.16 Multiple Authentication Mechanisms (FIA_UAU.5)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_UAU.5-ASE-01

*The evaluator shall ensure that the TSS describes each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication.*

*Specifically, for all authentication mechanisms specified in FIA_UAU.5.1, the evaluator shall ensure that the TSS describes the rules as to how each authentication mechanism is used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or biometric sample was entered), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used). If multiple BAFs are supported per FIA_UAU.5.1, the interaction between the BAFs must be described. For example, whether the multiple BAFs can be enabled at the same time.*

**Summary**

Section 8.5 *Identification and Authentication (FIA)* in the TSS of [ST] describes the three authentication mechanisms supported by the TOE including a passcode, fingerprint (Touch ID), or facial authentication (Face ID) of which the latter two are biometric mechanisms.

Passcode

All passcode entries are obscured by a dot symbol for each character as the user input occurs.

The TOE can enforce the following passcode policy 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 11, the default is 11)
- 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)

For biometric authentication, a passcode must be supplied for additional security validation under any of the following conditions:

- 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/Medical ID.

When a user exceeds the number of consecutive failed passcode login attempts, the user's partition is erased (by erasing the encryption key). The TOE OS partition is mounted READONLY upon boot and is never modified during the use of the TOE except during a software update or restore.

Touch ID

Devices that support Touch ID do not support Face ID.

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 fingerprint samples are presented, then the device will offer passcode entry. After five invalid biometric samples are presented, passcode authentication is required.

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.

In addition, the passcode and the biometric authentication methods cannot be combined for two-factor authentication.

## Guidance Assurance Activities

### Assurance Activity AA-FIA_UAU.5-AGD-01

*The evaluator shall verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.*

#### Summary

Per the definition of FIA_UAU.5 from [ST] , the TOE provides password (passcode), fingerprint (Touch ID), and face (Face ID) for user authentication.

[CCGUIDE] provides related guidance for each of the authentication mechanism as follows:

- Passcode: section 5.5.1 *Passcode Authentication Configuration*
- Touch ID: section 5.5.3 *Biometric Authentication Factors*
- Face ID: section 5.5.3 *Biometric Authentication Factors*

In previous assurance activities, the evaluator already determined that sufficient guidance is provided for each of the authentication mechanism noted above.

## Test Assurance Activities

### Assurance Activity AA-FIA_UAU.5-ATE-01

- ***Test 1:*** *For each authentication mechanism selected in FIA_UAU.5.1, the evaluator shall enable that mechanism and verify that it can be used to authenticate the user at the specified authentication factor interfaces.*
- ***Test 2:*** *For each authentication mechanism rule, the evaluator shall ensure that the authentication mechanism(s) behave accordingly.*

#### Summary

For both Test 1 & Test 2, the password mechanism is test during testing performed for FIA_AFL_EXT.1, FIA_PMG_EXT.1 {MDF}, FIA_UAU.6.2 {MDF}, FMT_SMF_EXT.1 Test 1, and FIA_UAU.7 as these tests all require locking and unlocking the device and verifying the success and failure.

For both biometric mechanisms, Touch ID and Face ID, the evaluator added his appropriate biometric data, locked the device, and confirmed the biometric mechanism could be used to unlock the device.

## 2.2.4.17 Re-Authentication (FIA_UAU.6)

### FIA_UAU.6.1

#### TSS Assurance Activities

No assurance activities defined.

#### Guidance Assurance Activities

No assurance activities defined.

#### Test Assurance Activities

#### Assurance Activity AA-FIA_UAU.6.1-ATE-01

- ***Test 1:*** *The evaluator shall configure the TSF to use the Password Authentication Factor according to the AGD guidance. The evaluator shall change Password Authentication Factor according to the AGD guidance and verify that the TSF requires the entry of the Password Authentication Factor before allowing the factor to be changed.*
- ***Test 2:*** *[conditional] For each BAF selected in FIA_UAU.5.1, the evaluator shall configure the TSF to use the BAF, which includes configuring the Password Authentication Factor, according to the AGD guidance. The evaluator shall change the BAF according to the AGD guidance and verify that the TSF requires the entry of the Password Authentication Factor before allowing the BAF to be changed.*
- ***Test 3:*** *[conditional] If "hybrid" is selected in FIA_UAU.5.1, the evaluator shall configure the TSF to use the BAF and PIN or password, which includes configuring the Password Authentication Factor, according to the AGD guidance. The evaluator shall change the BAF and PIN according to the AGD guidance and verify that the TSF requires the entry of the Password Authentication Factor before allowing the factor to be changed.*

#### Summary

Test 1: The evaluator changed the passcode on the TOE by going to
`Settings -> Touch ID & Passcode -> Change Passcode`
and verified that the old password is required.

Test 2: The evaluator tried to change both Touch ID using
`Settings -> Touch ID & Passcode`
and Face ID using
`Settings -> Face ID & Passcode`
and verified that the TOE prompted for a password in both cases.

Test 3 is not applicable because "hybrid" is not selected in FIA_UAU.5.1 in [ST].

### FIA_UAU.6.2

#### TSS Assurance Activities

No assurance activities defined.

#### Guidance Assurance Activities

No assurance activities defined.

#### Test Assurance Activities

#### Assurance Activity AA-FIA_UAU.6.2-ATE-01

- ***Test 1:*** *The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF_EXT.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.*
- ***Test 2:*** *[conditional] For each BAF selected in FIA_UAU.5.1, the evaluator shall repeat Test 1 verifying that the TSF requires the entry of the BAF before transitioning to the unlocked state.*
- ***Test 3:*** *[conditional] If "hybrid" is selected in FIA_UAU.5.1, the evaluator shall repeat Test 1 verifying that the TSF requires the entry of the BAF and PIN/password before transitioning to the unlocked state.*
- ***Test 4:*** *The evaluator shall configure user-initiated locking according to the AGD guidance. The evaluator shall lock the TSF and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.*
- ***Test 5:*** *[conditional] For each BAF selected in FIA_UAU.5.1, the evaluator shall repeat Test 4 verifying that the TSF requires the entry of the BAF before transitioning to the unlocked state.*
- ***Test 6:*** *[conditional] If "hybrid" is selected in FIA_UAU.5.1, the evaluator shall repeat Test 4 verifying that the TSF requires the entry of the BAF and PIN/password before transitioning to the unlocked state.*

**Summary**

Test 1: The evaluator made sure that the lock out period is not set to "never" and waited for the device to auto-lock. The evaluator then authenticated to the device.

Test 2: The evaluator re-performed Test 1 with Touch ID (fingerprint) and then Face ID (face).

Test 3 is not applicable because "hybrid" is not selected in FIA_UAU.5.1 in [ST].

Test 4: The evaluator locked the device manually (power button) and authenticated to the TOE with the password as described in Test 1.

Test 5: The evaluator performed Test 4 but unlocked the device with fingerprint (Touch ID) or Face ID.

Test 6 is not applicable because "hybrid" is not selected in FIA_UAU.5.1 in [ST].

## 2.2.4.18 Protected Authentication Feedback (FIA_UAU.7)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_UAU.7-ASE-01

*The evaluator shall ensure that the TSS describes the means of obscuring the authentication entry, for all authentication methods specified in FIA_UAU.5.1.*

**Summary**

Section 8.5 *Identification and Authentication (FIA)* in the TSS of [ST] describes the three authentication mechanisms supported by the TOE including a passcode, fingerprint (Touch ID), or facial authentication (Face ID) of which the latter two are biometric mechanisms.

This section specifically states the following:

- 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.

### Guidance Assurance Activities

#### Assurance Activity AA-FIA_UAU.7-AGD-01

*The evaluator shall verify that any configuration of this requirement is addressed in the AGD guidance and that the password is obscured by default.*

**Summary**

[CCGUIDE] section 5.5.2 *Protected Authentication Feedback* provides related guidance on protected authentication feedback. It states the following:

- All passcode entries are obscured by iOS/iPadOS. This is done by displaying a dot symbol in place of each character as the passcode entry user input occurs. No configuration of this feature is required from the mobile device administrator.
- Biometric authentication inputs do not provide feedback to the user unless the input is rejected. Additionally, biometric authentication inputs do not relay authentication entry information and are inherently obscured. When an invalid fingerprint sample is given or a fingerprint sample cannot be authenticated, a simple error message is returned which prompts the user to try again. When an invalid facial sample is given or a facial sample cannot be authenticated, the mobile device will vibrate. If three invalid biometric samples are presented the mobile device will offer passcode entry. After five invalid biometric samples are presented passcode authentication is required.

### Test Assurance Activities

#### Assurance Activity AA-FIA_UAU.7-ATE-01

- ***Test 1:*** *The evaluator shall enter passwords on the device, including at least the Password Authentication Factor at lockscreen, and verify that the password is not displayed on the device.*
- ***Test 2:*** *[conditional] For each BAF selected in FIA_UAU.5.1, the evaluator shall authenticate by producing a biometric sample at lockscreen. As the biometric algorithms are performed, the evaluator shall verify that sensitive images, audio, or other information identifying the user are kept secret and are not revealed to the user. Additionally, the evaluator shall produce a biometric sample that fails to authenticate and verify that the reason(s) for authentication failure (user mismatch, low sample quality, etc.) are not revealed to the user. It is acceptable for the BAF to state that it was unable to physically read the biometric sample, for example, if the sensor is unclean or the biometric sample was removed too quickly. However, specifics regarding why the presented biometric sample failed authentication shall not be revealed to the user.*

**Summary**

Test 1: The evaluator locked the device and verified that when unlocking, password characters are not displayed and dots are displayed instead.

Test 2: The evaluator locked and unlocked the device using both Touch ID and Face ID and verified that no information pertaining to either the fingerprint or the face is displayed.

## 2.2.4.19 Authentication for Cryptographic Operation (FIA_UAU_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_UAU_EXT.1-ASE-01

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

**Summary**

Section 8.3.1 *Overview of Key Management* in the TSS of the [ST] describes key management.

This section states that the user's passcode is used, combined with the UID and the salt (128-bit random salt generated by the SEP described in section 8.2.1 of the TSS), to derive the 256-bit passcode key which is then used to unwrap the user's keybag that holds the class keys for the file system data protection. Only when the unwrapping of the user's keybag is successful is the user considered authenticated. That passcode key is erased whenever the device is locked, and reconstructed whenever the device is unlocked with the right passcode.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_UAU_EXT.1-ATE-01

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

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

- ***Test 1:*** *The evaluator shall enable encryption of protected data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as protected data. The evaluator shall reboot the device, use a tool provided by developer to search for the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by the developer to access the unique string amongst the application data, and verify that the unique string can be found.*
- ***Test 2:*** *[conditional] The evaluator shall require user authentication according to the AGD guidance. The evaluator shall store a key in the software-based secure key storage. The evaluator shall lock the device, use a tool provided by developer to access the key amongst the stored data, and verify that the key cannot be retrieved or accessed. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the key, and verify that the key can be retrieved or accessed.*
- ***Test 3:*** *[conditional] The evaluator shall enable encryption of sensitive data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as sensitive data. The evaluator shall lock the device, use a tool provided by developer to attempt to access the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the unique string amongst the application data, and verify that the unique string can be retrieved.*

#### Summary

Test 1: The suggested testing cannot be conducted on any Apple device due to the data encryption implementation as described in the test for FDP_DAR_EXT.1. The evaluator verified those steps in the source code of the TOE. This approach has been approved by NIAP in previous iOS evaluations.

Test 2 & Test 3: Please see the test for FCS_CKM_EXT.4. This approach has been approved by NIAP in previous iOS evaluations.

## 2.2.4.20 Timing of Authentication (FIA_UAU_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_UAU_EXT.2-ASE-01

*The evaluator shall verify that the TSS describes the actions allowed by unauthorized users in the locked state.*

**Summary**

Section 8.5 *Identification and Authentication (FIA)* in the TSS of the [ST] describes user authentication. It states that making emergency calls, answering calls, accessing Medical ID information, using the cameras (unless their use is generally disallowed), using the flashlights, using the control center, and using the notification center are the functions that do not require user authentication.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_UAU_EXT.2-ATE-01

*The evaluator shall attempt to perform some actions not listed in the selection while the device is in the locked state and verify that those actions do not succeed.*

**Summary**

The evaluator performed this test twice, once waiting for the device to lock automatically and a second time by locking the device manually. In both instances, the evaluator verified that the screen is overwritten by non-sensitive data and that the only functions that can be performed are answering calls, making emergency calls, accessing Medical ID (iPhones only), using the camera (unless that use is generally disallowed), using the flashlight, and using Control Center or Notification Center.

## 2.2.4.21 X.509 Validation of Certificates (FIA_X509_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_X509_EXT.1-ASE-01

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

**Summary**

Section 8.5.2 *X.509v3 Certificates* in the TSS of the [ST] describes X.509v3 certificates.

The evaluator reviewed this section and confirmed that the verification of certificates takes place whenever the TOE tries to establish a connection (via IPsec, TLS, or EAP-TLS) using a remote certificate. The process for certificate validation is as follows:

- When attempting to establish a connection using a peer certificate (i.e., a certificate received from the other endpoint), the peer certificate is first checked to ensure it is valid as per RFC 5280. Certificates are validated against the Subject Alternative Name (SAN). Wildcards are supported. The Common Name (CN) is ignored. If the SAN 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.

- The TOE, excluding WLAN, uses the online certificate status protocol (OCSP) for validating the revocation status of certificates. When a connection cannot be established to the OCSP server to determine the revocation status of a certificate, the TOE considers the certificate as not revoked.
- As part of the certificate chain validation, the validity period of each certificate in the chain is verified. If the certificate is marked as an extended validation certificate, the TOE performs an OCSP lookup to verify the validity (revocation status) of the certificate (except for WLAN certificate validation which does not support OCSP). The basicContraints extension and the CA flag are checked. CA certificates must have the basicContraints extension, the CA flag set to TRUE, and include the caSigning purpose. The extendedKeyUsage (EKU) is validated against the rules defined in FIA_X509_EXT.1 (which is a superset of the rules in FIA_X509_EXT.1/WLAN). Finally, the signature of the issuer of the certificate is verified. Only when all checks succeed, the certificate is considered valid and the next certificate in the certificate chain is checked.
- The certificate chain searches for the certificates in the trust store. The trust store is a combination of the trust store delivered with the TOE and the certificates stored in the keychain and marked as trustworthy. Certificates from the trusted store are validated using the previously described checks at the time that they are used. Certificate path validation terminates with a certificate in the trust store.
- 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 (e.g., HTTPS connection) will display a message to the user that the peer certificate validation failed and allow the user to examine the certificate with the option to allow the connection or not.

## Guidance Assurance Activities

No assurance activities defined.

## Test Assurance Activities

### Assurance Activity AA-FIA_X509_EXT.1-ATE-01

*The tests described must be performed in conjunction with the other Certificate Services evaluation activities, including the use cases in FIA_X509_EXT.2.1 and FIA_X509_EXT.3. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. The evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.*

- ***Test 1:*** *The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:*
  - *by establishing a certificate path in which one of the issuing certificates is not a CA certificate,*
  - *by omitting the basicConstraints field in one of the issuing certificates,*
  - *by setting the basicConstraints field in an issuing certificate to have CA=False,*
  - *by omitting the CA signing bit of the key usage field in an issuing certificate, and*
  - *by setting the path length field of a valid CA field to a value strictly less than the certificate path.*

  *The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.*
- ***Test 2:*** *The evaluator shall demonstrate that validating an expired certificate results in the function failing.*
- ***Test 3:*** *The evaluator shall test that the TOE can properly handle revoked certificates-conditional on whether CRL, OCSP, OSCP stapling, or OCSP multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method: The evaluator shall test revocation of the node certificate. The evaluator shall also test revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). For the test of the WLAN use case, only pre-stored CRLs are used. If OCSP stapling per RFC 6066 is the only supported revocation method, this test is omitted. The evaluator shall ensure that a valid certificate*

*is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.*

- ***Test 4:*** *If any OCSP option is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.*
- ***Test 5:*** *The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate (the certificate will fail to parse correctly).*
- ***Test 6:*** *The evaluator shall modify any bit in the last byte of the signature algorithm of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).*
- ***Test 7:*** *The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).*
- ***Test 8:***
  - ***Test 1:*** *(Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.*
  - ***Test 2:*** *(Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.*

**Summary**

Test 1: The evaluator verified that each of a number of improper settings in an issuing certificate in a certificate path cause a connection attempt to fail.

1. a certificate is not a CA certificate
2. a certificate with no basicConstraints field
3. setting the basicConstraints field to CA=False
4. omitting the CA signing bit of the key usage field
5. setting the path length field of a valid CA field to a value strictly less than the certificate path

The evaluator also confirmed that establishing a valid certificate path consisting of valid CA certificates allows the connection attempt to succeed. The test where an installed and otherwise trusted CA certificate has the trust removed to verify that the remote server is not authenticated is performed in FCS_HTTPS_EXT.1 Test 1 and 3.

Test 2: The evaluator generated a certificate for the server, advanced the date one day so that the certificate is expired, connected the TOE to https://server and verified that the connection is rejected.

Test 3: The evaluator cleared the Safari cache settings, connected the TOE to the network, started Wireshark, and accessed the multiple URLs with Extended Validation (EV) certificates. The evaluator verified that the OCSP requests were found in the pcap files. The revocation test is performed as part of Test 4.

Test 4: The evaluator tested an OCSP server with a revoked certificate and found the expected result in the pcap file. The evaluator tested with an invalid OCSP responder certificate per the test and verified that the certificate was not determined to be revoked because the OCSP response was discarded.

Test 5: The evaluator attempted to deploy a configuration profile containing a modified certificate as described. The TOE returned an error for the invalid certificate.

Test 6: The evaluator attempted to deploy a configuration profile containing a modified certificate as described. The TOE returned an error for the invalid certificate.

Test 7: The evaluator attempted to deploy a configuration profile containing a modified certificate as described. The TOE returned an error for the invalid certificate.

Test 8a: The evaluator created an EC certificate chain to a web server for each supported curve and was able to establish a TLS connection to it from the TOE via the web browser.

Test 8b: The evaluator replaced the intermediate certificate in the chain used in each test in Test 8a with a certificate modified as described and found the TOE showed the certificates were not trusted and the connection was unsuccessful.

## 2.2.4.22 X.509 Certificate Validation (EAP-TLS) (FIA_X509_EXT.1/WLAN)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_X509_EXT.1-WLAN-ASE-01

*[TD0439] The evaluator shall ensure the TSS describes where the check of validity of the EAP-TLS certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.*

#### Summary

Section 8.5.2 *X.509v3 Certificates* of [ST] describes X.509 certificates. It states the following:

- When attempting to establish a connection using a peer certificate (i.e., a certificate received from the other endpoint), the peer certificate is first checked to ensure it is valid as per RFC 5280. Certificates are validated against the Subject Alternative Name (SAN). Wildcards are supported. The Common Name (CN) is ignored. If the SAN 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.
- The TOE, excluding WLAN, uses OCSP for validating the revocation status of certificates. This includes EAP-TLS even though EAP-TLS is not required to support certificate revocation. When a connection cannot be established to the OCSP server to determine the revocation status of a certificate, the TOE considers the certificate as not revoked.
- As part of the certificate chain validation, the validity period of each certificate in the chain is verified. If the certificate is marked as an extended validation certificate, the TOE performs an OCSP lookup to verify the validity (revocation status) of the certificate (except for WLAN certificate validation which does not support OCSP). The basicContraints extension and the CA flag are checked. CA certificates must have the basicContraints extension, the CA flag set to TRUE, and include the caSigning purpose. The extendedKeyUsage (EKU) is validated against the rules defined in FIA_X509_EXT.1 (which is a superset of the rules in FIA_X509_EXT.1/WLAN). Finally, the signature of the issuer of the certificate is verified. Only when all checks succeed, the certificate is considered valid and the next certificate in the certificate chain is checked.

- The certificate chain searches for the certificates in the trust store. The trust store is a combination of the trust store delivered with the TOE and the certificates stored in the key chain and marked as trustworthy. Certificates from the trusted store are validated using the previously described checks at the time that they are used. Certificate path validation terminates with a certificate in the trust store.
- 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 (e.g., HTTPS connection) will display a message to the user that the peer certificate validation failed and allow the user to examine the certificate with the option to allow the connection or not.

## Guidance Assurance Activities

No assurance activities defined.

## Test Assurance Activities

### Assurance Activity AA-FIA_X509_EXT.1-WLAN-ATE-01

*[TD0439] The tests described must be performed in conjunction with the other Certificate Services assurance activities. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. The evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.*

- *Test 1: The evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function (e.g. application validation), and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.*
- *Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.*
- *Test 3: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.*
- *Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.*
- *Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate (the certificate will fail to parse correctly).*
- *Test 6: The evaluator shall modify any bit in the last byte of the signature algorithm of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).*
- *Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).*

### Summary

Test 1: The evaluator created both a valid and invalid certificate and demonstrated the connection attempt was successful using the valid certificate and unsuccessful using the invalid certificate.

Test 2: The evaluator created a certificate expiring "tomorrow" and used it for a successful connection. He then set the TOE's clock forward one day and attempted the connection again, but this time the connection was not successful.

Test 3: The evaluator created a CA certificate without a basicConstraints extension and verified the TOE could not connect to the network.

Test 4: The evaluator created a CA certificate with the basicConstraints extension and the CA flag set to false and verified the TOE could not connect to the network.

Test 5: The evaluator modified a byte in the first 8 bytes of a valid certificate and attempted to upload it to the TOE using Apple Configurator 2 and found the upload failed.

Test 6: The evaluator modified a bit in the last byte of a valid certificate and attempted to upload it to the TOE using Apple Configurator 2 and found the upload failed.

Test 7: The evaluator modified a byte in the public key of a valid certificate and attempted to upload it to the TOE using Apple Configurator 2 and found the upload failed.

## 2.2.4.23 X.509 Certificate Authentication (FIA_X509_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_X509_EXT.2-ASE-01

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

*The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described.*

#### Summary

Section 8.5.2 *X.509v3 Certificates* in the TSS of the [ST] describes X.509v3 certificates. This section states that every certificate has a certificate type respective to the area in which this certificate is going to be used such as AppleX509Basic, AppleIPsec, and AppleCodeSigning, as described in the administrative guidance [CCGUIDE] .

This section also describes certification vallidation as follows:

- When attempting to establish a connection using a peer certificate (i.e., a certificate received from the other endpoint), the peer certificate is first checked to ensure it is valid as per RFC 5280. Certificates are validated against the Subject Alternative Name (SAN). Wildcards are supported. The Common Name (CN) is ignored. If the SAN 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.
- The TOE, excluding WLAN, uses OCSP for validating the revocation status of certificates. When a connection cannot be established to the OCSP server to determine the revocation status of a certificate, the TOE considers the certificate as not revoked.
- As part of the certificate chain validation, the validity period of each certificate in the chain is verified. If the certificate is marked as an extended validation certificate, the TOE performs an OCSP lookup to verify the validity (revocation status) of the certificate (except for WLAN certificate validation which does not support OCSP). The basicContraints extension and the CA flag are checked. CA certificates must have the basicContraints extension, the CA flag set to TRUE, and include the caSigning purpose. The extendedKeyUsage (EKU) is validated against the rules defined in FIA_X509_EXT.1 (which is a superset of the rules in FIA_X509_EXT.1/WLAN). Finally, the signature of the issuer of the certificate is verified. Only when all checks succeed, the certificate is considered valid and the next certificate in the certificate chain is checked.

- The certificate chain searches for the certificates in the trust store. The trust store is a combination of the trust store delivered with the TOE and the certificates stored in the key chain and marked as trustworthy. Certificates from the trusted store are validated using the previously described checks at the time that they are used. Certificate path validation terminates with a certificate in the trust store.
- 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 (e.g., HTTPS connection) will display a message to the user that the peer certificate validation failed and allow the user to examine the certificate with the option to allow the connection or not.

## Guidance Assurance Activities

### Assurance Activity AA-FIA_X509_EXT.2-AGD-01

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

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

### Summary

The TSS in section 8.5.2 *X.509v3 Certificates* of [ST] states the following:

- There are a number of X.509v3 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 have a certificate type that defines their respective application area. This ensures that only certificates defined for a specific application area are used.
- The certificate chain searches for the certificates in the trust store. The trust store is a combination of the trust store delivered with the TOE and the certificates stored in the keychain and marked as trustworthy.

[CCGUIDE] section 5.5.6 provides related guidance which states the following:

- X.509 certificates are configured by an administrator using the keys of the Certificates Payload in a Configuration Profile as described in [DeployRef]
- Certificates have a certificate type that defines their respective application area. This ensures that only certificates defined for a specific application area are used. Certificate types supported are AppleX509Basic, AppleSSL, AppleSMIME, AppleEAP, AppleIPsec, AppleCodeSigning, AppleIDValidation, AppleTimeStamping
- External entities can be authenticated using a digital certificate. Out of the box, the TOE includes a number of preinstalled root certificates.
- Code signing certificates need to be assigned by Apple and can be imported into a device.
- There are also blocked and always-ask certificates in the Trust Anchor Database. Blocked certificates are believed to be compromised and are never trusted. Always-ask certificates prompt the user whether they want to trust the certificate.
- In the evaluated configuration, mobile device administrators are allowed to modify the Trust Anchor Database using a Configuration Profile.

### Test Assurance Activities

#### Assurance Activity AA-FIA_X509_EXT.2-ATE-01

*The evaluator shall perform the following test for each trusted channel:*

- ***Test 1:*** *The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.*

#### Summary

Test 1: The evaluator set up a packet filter to drop packets sent from the TOE to the OCSP server, thus preventing the TOE from completing a request. The evaluator observed the web browser established the connection without asking the user to trust the certificate, as expected.

## 2.2.4.24 X.509 Certificate Authentication (EAP-TLS) (FIA_X509_EXT.2/WLAN)

### TSS Assurance Activities

#### Assurance Activity AA-FIA_X509_EXT.2-WLAN-ASE-01

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

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

#### Summary

Section 8.9.1 *EAP-TLS and TLS* in the TSS of [ST] describes the EAP-TLS and TLS protocols including the ciphersuites supported by the TOE. This section describes that when the TOE is configured to use EAP-TLS, 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. When the TLSAllowTrustExceptions key in the Wi-Fi payload is used, the administrator can enforce that untrusted certificates are not accepted and the authentication fails if such an untrusted certificate is presented. Additional administrative guidance is provided in [CCGUIDE] .

### Guidance Assurance Activities

#### Assurance Activity AA-FIA_X509_EXT.2-WLAN-AGD-01

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

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

**Summary**

[CCGUIDE] section 5.5.6 *X.509 Certificate Configuration* states the following:

- X.509 certificates are configured by an administrator using the keys of the *Certificate Payload* in a Configuration Profile as described in [DEV_MAN] .
- Certificates have a certificate type that defines their respective application area thus ensuring 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:
  - AppleX509Basic
  - AppleSSL
  - AppleSMIME
  - AppleEAP
  - AppleIPsec
  - AppleCodeSigning
  - AppleIDValidation
  - AppleTimeStamping
- Per [TRUST_STORE] , the TOE has a iOS/iPadOS Trust Store that contains trusted root certificates presintalled with the iOS/iPadOS. These preinstalled trusted root certificates cannot be modified, are automatically trusted, and do not need to be included when creating a Configuration Profile. New certificates can be added to the Trust Anchor Database or currently installed certificates can be removed.
- WLAN certificate configuration can be found in [DEV_MAN] and subsection "EAPClientConfiguration Dictionary" of the Wi-Fi section of [DEV_MAN] . In particular, when configuring the TOE devices to utilize EAP-TLS as part of a 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 described in [DEV_MAN] .
- To configure the TOE devices to reject untrusted certificates, the administrator can use the PayloadCertificateAnchorUUI and TLSAllowTrustExceptions dictionary keys in the Wi-Fi Payload EAPClientConfiguration Dictionary of the Configuration Profile which enforces that untrusted certificates are not accepted and the authentication fails if such untrusted certificates are presented.
- To enforce the verification of the server name defined with the X.509 certificate during the WPA-EAP handshake between the TOE device and the remote access point, the policy must contain the server name to be expected in the certificate with the TLSTrustedServerNames dictionary key in the Wi-Fi Payload EAPClientConfiguration Dictionary of the Configuration Profile.

## Test Assurance Activities

### Assurance Activity AA-FIA_X509_EXT.2-WLAN-ATE-01

*The evaluator shall perform the following test for each trusted channel:*

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

**Summary**

Test 1: The evaluator configured the EAP server to only support one cipher. OpenSSL supports all cipher modes. For each configured cipher, the evaluator connected the TOE to the server successfully.

Test 2: The evaluator then created a server certificate without the "Server Authentication" value in the extendedKeyUsage field and verified than the TOE could not connect successfully to the server.

Test 3: The evaluator used an ECDSA certificate and attempted to connect to the server supporting only the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite and verified that the TOE could not connect successfully to the server.

The evaluator notes that the comparison to an action selected in FIA_X509_EXT.2.2 is no longer necessary since that SFR is removed by TD0517.

## 2.2.4.25 Request Validation of Certificates (FIA_X509_EXT.3)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FIA_X509_EXT.3-ATE-01

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

**Summary**

The evaluator used an application provided by the developer, namely Safari, which requests certificate validation by the TSF. Safari uses the certificate validation logic offered by the TSF via the NSURLSession API (https://developer.apple.com/documentation/foundation/nsurlsession).

The hostap EAP-TLS server is configured to accept only one cipher. OpenSSL supports all selected cipher modes. For each configured cipher, an access to the web server must be performed which must be successful. The evaluator generated a full CA hierarchy and connected the TOE using supported ciphersuite. The evaluator verified that the web connection is successful and that the web page displays the TLS information.

# 2.2.5 Security management (FMT)

## 2.2.5.1 Management of Security Functions Behavior (FMT_MOF_EXT.1)

### FMT_MOF_EXT.1.1

#### TSS Assurance Activities

#### Assurance Activity AA-FMT_MOF_EXT.1.1-ASE-01

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

#### Summary

Section 8.6.2 *Configuration Profiles* in the TSS of [ST] describes the TOE management using configuration profiles.

The evaluator identified from table 6 "Management Functions" of [ST] that the functions 4, 13, 18, 20, 21, 23, and 47 are restricted to the users only. The last paragraph of section 8.6.2 explains that configuration profiles can be deployed such that users are unable to override or remove restrictions set by the TOE administrators or MDM administrators. Depending on the behavior defined in the configuration profile, users might be unable to access, perform, or relax management functions defined in table 6 of [ST] .

#### Guidance Assurance Activities

No assurance activities defined.

#### Test Assurance Activities

No assurance activities defined.

### FMT_MOF_EXT.1.2

#### TSS Assurance Activities

#### Assurance Activity AA-FMT_MOF_EXT.1.2-ASE-01

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

#### Summary

Section 8.6 *Specification of Management Functions (FMT)* describes security management of the TOE. It states that administrative configuration can be specified via the Configuration Profiles deployed with either the Apple Configurator 2, an email message, a web site, over-the-air configuration ), or over-the-air configuration via an MDM server. Settings or restrictions defined in these Configuration Profiles would restrict any options the user has. If the Configuration Profiles are protected against removal by the administrator, the user cannot circumvent the restrictions or

settings by removing the respective profile. In other words, depending on the behavior defined in the Configuration Profile, users will be unable to access, perform, or relax management functions defined in table 6 "Management Functions" of [ST] .

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FMT_MOF_EXT.1.2-ATE-01

- ***Test 1:*** *The evaluator shall use the test environment to deploy policies to Mobile Devices.*
- ***Test 2:*** *The evaluator shall create policies which collectively include all management functions which are controlled by the (enterprise) administrator and cannot be overridden/relaxed by the user as defined in FMT_MOF_EXT.1.2. The evaluator shall apply these policies to devices, attempt to override/relax each setting both as the user (if a setting is available) and as an application (if an API is available), and ensure that the TSF does not permit it. Note that the user may still apply a more restrictive policy than that of the administrator.*
- ***Test 3:*** *Additional testing of functions provided to the administrator are performed in conjunction with the testing activities for FMT_SMF_EXT.1.1.*

#### Summary

Test 1: Numerous previous tests have used Apple Configurator 2 to deploy policies.

Test 2: All configuration options are tested for FMT_SMF_EXT.1. For administrator-related configuration options, Apple Configurator 2 is used.

Test 3: Please see the test for FMT_SMF_EXT.1.

## 2.2.5.2 Agent Trusted Policy Update (FMT_POL_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FMT_POL_EXT.2-MDMA-ASE-01

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

#### Summary

Section 8.6.2 *Configuration Profiles* in the TSS of [ST] states that candidate policies (defined in Configuration Profiles) can be received using the Apple Configurator 2 tool, as a file (via an email message, a web page or a files folder), via OTA configuration, or OTA configuration via an MDM server. This section also states that to Configuration Profiles are digitally signed and encrypted. The Configuration Profile can specify if a digital signature of the MDM payload is required. When checked as required possible outcomes are:

- Signature verified - the payload is then deployed.
- Signature failed - the payload is then not deployed.

- No signature present or signature cannot be verified due to other reasons such as missing a CA certificate - the TOE then checks the origin of the payload (i.e., its connection). If the connection to the MDM payload origin (e.g., Apple Configurator 2) is trusted (i.e., the certificates can be validated and its signature checks out), the MDM payload is processed as trusted and deployed.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FMT_POL_EXT.2-MDMA-ATE-01

*This evaluation activity is performed in conjunction with the evaluation activity for FIA_X509_EXT.1 and FIA_X509_EXT.2 as defined in the Base-PPs.*

- ***Test 1:*** *The evaluator shall perform a policy update from an available configuration interface (such as through a test MDM Server). The evaluator shall verify the update is signed and is provided to the MDM Agent. The evaluator shall verify the MDM Agent accepts the digitally signed policy.*
- ***Test 2:*** *The evaluator shall perform a policy update from an available configuration interface (such as through a test MDM Server). The evaluator shall provide an unsigned and an incorrectly signed policy to the MDM Agent. The evaluator shall verify the MDM Agent does not accept the digitally signed policy.*

#### Summary

The assurance activities for this requirement are performed in conjunction with the assurance activities for FIA_X509_EXT.1 and FIA_X509_EXT.2 as defined in the base PP. Test 1 is also executed along with Test 4 for FAU_ALT_EXT.2 {AGENT}.

Test 1: The evaluator deployed a password policy on the TOE through the Apple Profile Manager and could verify that the added policy's signature was successfully verified by the TOE.

Test 2: The evaluator created a policy with one option, generated a certificate using the keychain tool, signed the policy using a certificate that the TOE would not accept. The evaluator then loaded the policy and verified that the TOE prompted the evaluator with the choice to accept the profile or not.

## 2.2.5.3 Specification of Management Functions (VPN) (FMT_SMF.1/VPN)

### TSS Assurance Activities

#### Assurance Activity AA-FMT_SMF.1-VPN-ASE-01

*The evaluator shall check to ensure the TSS describes the client credentials and how they are used by the TOE.*

#### Summary

Section 8.6.7 *VPN Certificate Credentials* in the TSS of [ST] identifies X.509 certificates as client credentials used for IPsec authentication.

Section 8.9.4.4 *Peer authentication* in the TSS of [ST] describes how the TOE performs IPsec authentication using X.509 certificates. During the authentication process, a comparison is made of the Subject Alternative Name (SAN) contained within the peer certificate to the SAN of the requested server. If the SAN in the certificate does not match the expected SAN for the peer, then the session will not be established. The Common Name (CN) is ignored. If the SAN in the peer

certificate does not match that of the peer's identifier or a SAN does not exist in the peer certificate, the authentication process fails. If the SAN matches the peer's identifier, the authentication process is successful.

### Guidance Assurance Activities

#### Assurance Activity AA-FMT_SMF.1-VPN-AGD-01

*The evaluator shall check to make sure that every management function mandated in the ST for this requirement is described in the operational guidance and that the description contains the information required to perform the management duties associated with each management function.*

#### Summary

The evaluator noted that administrator-provided management functions defined in [PP_MDF_V3.2] and [MOD_VPNC_V2.3] are provided throughout [CCGUIDE] . A high-level summary is provided in [CCGUIDE] Table 5 "Required Mobile Device Management Functions" where for each management function, a pointer to the related guidance is provided. Also, these management functions are already assessed in prior work units of this report particularly the work units of AA-FMT_SMF_EXT.1-AGD-01

Additionally, [CCGUIDE] provides Table 4 "SFR Configuration Requirements" outlining for each function (expressed in the form of an SFR and its description) whether configuration is required.

### Test Assurance Activities

#### Assurance Activity AA-FMT_SMF.1-VPN-ATE-01

*The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE according to the operational guidance and testing each management activity listed in the ST.*

*The evaluator shall ensure that all management functions claimed in the ST can be performed by completing activities described in the AGD. Note that this may be performed in the course of completing other testing.*

#### Summary

The evaluator executed all the test required by [MOD_VPNC_V2.3] which covers all the management functions defined in FMT_SMF_EXT.1.

## 2.2.5.4 Specification of Management Functions (FMT_SMF_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FMT_SMF_EXT.1-ASE-01

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

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

***Function 1***

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

***Function 2***

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

***Function 3***

*No activity for this function.*

***Function 4***

*The evaluator shall verify that the TSS includes a description of each radio and an indication of if the radio can be enabled/disabled along with what role can do so. In addition the evaluator shall verify that the frequency ranges at which each radio operates is included in the TSS. The evaluator shall verify that the TSS includes at what point in the boot sequence the radios are powered on and indicates if the radios are used as part of the initialization of the device.*

***Function 5***

*The evaluator shall verify that the TSS includes a description of each collection device and an indication of if it can be enabled/disabled along with what role can do so.*

***Function 6***

*No activity for this function.*

***Function 7***

*No activity for this function.*

***Function 8***

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

***Function 9 & Function 10***

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

***Function 11***

*No activity for this function.*

***Function 12***

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

***Function 13***

*The evaluator shall examine the TSS to ensure that it contains a description of each management function that will be enforced by the enterprise once the device is enrolled.*

***Function 14***

*The evaluator shall verify that the TSS includes an indication of what applications (e.g., user-installed applications, Administrator-installed applications, or Enterprise applications) can be removed along with what role can do so.*

***Function 15***

*No activity for this function.*

***Function 16***

*No activity for this function.*

***Function 17***

*No activity for this function.*

***Function 18***

*No activity for this function.*

***Function 19***

*No activity for this function.*

***Function 20***

*No activity for this function.*

***Function 21***

*No activity for this function.*

***Function 22***

*No activity for this function.*

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

***Function 23 [conditional]***

*No activity for this function.*

***Function 24 [conditional]***

*The evaluator shall verify that the TSS includes a list of each externally accessible hardware port and an indication of if data transfer over that port can be enabled/disabled.*

***Function 25 [conditional]***

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

***Function 26 [conditional]***

*No activity for this function.*

***Function 27 [conditional]***

*No activity for this function.*

***Function 28 [conditional]***

*No activity for this function.*

***Function 29 [conditional]***

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

***Function 30 [conditional]***

*No activity for this function.*

***Function 31 [conditional]***

*The evaluator shall ensure that the TSS describes which cellular protocols can be disabled.*

***Function 32 [conditional]***

*No activity for this function.*

***Function 33 [conditional]***

*No activity for this function.*

***Function 34 [conditional]***

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

***Function 35 [conditional]***

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

***Function 36 [conditional]***

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

***Function 37 [conditional]***

*No activity for this function.*

***Function 38 [conditional]***

*No activity for this function.*

***Function 39 [conditional]***

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

***Function 40 [conditional]***

*The evaluator shall verify that the TSS includes a description of available backup methods that can be enabled/disabled. If "selected applications" or "selected groups of applications" are selected the TSS shall include which applications of groups of applications backup can be enabled/disabled.*

***Function 41 [conditional]***

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

***Function 42 [conditional]***

*No activity for this function.*

***Function 43 [conditional]***

*No activity for this function.*

***Function 44 [conditional]***

*No activity for this function.*

***Function 45 [conditional]***

*The evaluator shall verify that the TSS contains guidance to configure the VPN as Always-On.*

***Function 46 [conditional]***

*The evaluator shall verify that the TSS describes the procedure to revoke a biometric credential stored on the TOE.*

***Function 47***

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

**Summary**

Section 8.6 *Specification of Management Functions (FMT)* describes security management of the TOE.

This section references table 6 "Management Functions" of [ST] , which describes which roles can perform which functions. The last paragraph of section 8.6.2 explains that Configuration Profiles can be deployed such that users are unable to override or remove restrictions set by the TOE administrators or MDM administrators. Depending on the behavior defined in the Configuration Profile, users might be unable to perform, or access, management functions defined in table 6.

**Function 1**

Section 8.5 *Identification and Authentication (FIA)* in the TSS of [ST] describes the passcode policy including the length, complexity and lifetime of the passcode configurable by the user authenticated to the device, or by administrator using a Configuration Profile if the device is under MDM.

**Function 2**

Section 8.5 *Identification and Authentication (FIA)* in the TSS of [ST] describes the passcode policy stating that the maximum number of consecutive failed attempts to enter the right passcode can be set to, between 2 and 11 (with 11 being the default), again, by either the

user or through a configuration profile. This section also describes that the number of minutes for which the device can be idle before it locks itself automatically can be defined by the user or a Configuration Profile.

**Function 4**

Section 8.5 *Identification and Authentication (FIA)* in the TSS of [ST] describes the passcode policy stating that the maximum number of consecutive failed attempts to enter the right passcode can be set to, between 2 and 11 (with 11 being the default), again, by either the user or through a configuration profile. This section also describes that the number of minutes for which the device can be idle before it locks itself automatically can be defined by the user or a Configuration Profile.

**Function 5**

Section 8.6.6 *Audio and Visual collection devices* in the TSS of [ST] which identifies the cameras and microphones as audio and visual collection devices. This section refers to table 5 "Management Functions" and according to this table, the cameras can be enabled/disabled across device or on a per-app basis and the microphone is enabled/disabled on a per-app basis. Also for both the cameras and microphones, either the user or administrator can enable/disable.

**Function 8**

The ST selects "denying installation of applications" in function 8 of FMT_SMF_EXT.1.

Section 8.6.2 *Configuration Profiles* in the TSS [ST] describes that administrator can configure via a Configuration Profile to allow or disable installation of apps by a user.

**Functions 9 & 10**

Section 8.3.2 *Storage of Persistent Secrets and Private Keys by the Agent* in the TSS of [ST] provides table 8 "Summary of keys and persistent secrets used by the Agent" summarizing the keys/secrets that can be imported and stored on the TOE device, their purpose, and how they are stored.

**Function 12**

Section 8.5.2 *X.509v3 Certificates* in the TSS of [ST] describes the different certificates used by the TOE. According to this section X.509 certificates are used for the following purposes:

- The Apple certificate is used for verifying the integrity and authenticity of software updates.
- Other certificates are used for setting up trusted channels including for IPsec, TLS, and EAP-TLS connections.
- Certificates are also used for validating the integrity of Configuration Profiles.
- Certificates are used for code signing.

**Function 13**

Section 8.6.2 *Configuration Profiles* in the TSS of [ST] describes management functions including those that will be enforced by the enterprise once the device is enrolled. These management functions which include password policy, VPN policy, Wi-Fi policy, allow/disallow specific services (e.g., remote backup) are enforced using Configuration Profiles.

**Function 14**

Section 8.6.2 *Configuration Profiles* states that configuration profiles can also be deployed such that users are unable to override or remove restrictions set in place by Administrators or MDM Administrators. This includes installation or removal of application of the TOE.

**Function 18**

The ST selects function 1 "specify minimum level of security for each pairing. (for BR/EDR and LE" for Function 18 in FMT_SMF_EXT.1.

Section 8.9.2 *Bluetooth* in the TSS of [ST] describes the Bluetooth protocol. This section states that the TOE supports Bluetooth 4.2, and 5.0 with the following 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)

Also, this section describes the pairing process which requires the TOE users to manually (via the TOE device's Bluetooth status menu) pair their device with a remote Bluetooth device. During the pairing time, another device (or the TOE itself) can send a pairing request which is commonly a six-digit number displayed on both sides which must be manually matched by the user. If one device does not support this automatic exchange of PIN, a window for entering a manual PIN is presented. Devices that want to pair with the TOE must use Secure Simple Pairing which uses ECDH-based authentication and key exchange. Connections via Bluetooth is secured using AES-CTR-128 with CCM.

**Function 36**

Section 8.8.3 *Lock Screen / Access Banner Display* in the TSS of [ST] states that the banner can be defined using an image that is presented during the lock screen. As this banner is an image there are no character limitations, information is restricted to what can be included in an image appropriate to the TOE device display.

**Function 45**

Section 8.9.4.1 *AlwaysOn VPN* in the TSS of [ST] describes the Always-On VPN which states the following:

For managed and supervised devices, the TOE must 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 and/or Wi-Fi) for which the VPN is 'AlwaysOn' (default is: for both cellular and Wi-Fi), the specification of exceptions from this service (only VoiceMail, AirPrint, and CellularServices can be listed as exceptions), and the definition of exceptions for Captive networking (if any). The TOE supports separate configurations for cellular and Wi-Fi.

For IKEv2, the configuration contains the following (among other items).

- The IP address or hostname of the VPN server
- 
- 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-CBC-128, AES-CBC-256 (default), AES-GCM-128, and AES-GCM-256).
- The integrity algorithm (allows for SHA1-160, SHA2-256 (default), SHA2-384, SHA2-512).

**Function 47**

The ST assignss "Enable/disable microphones on a per-app basis" for Function 47 in FMT_SMF_EXT.1.

Section 8.6.2 *8.6.2 Configuration Profiles* states that the microphones cannot be disabled in general but a user can restrict access to the microphones on a per-app basis.

All other functions not mentioned above are either have no assurance activities or not claimed in the ST.

## Guidance Assurance Activities

### Assurance Activity AA-FMT_SMF_EXT.1-AGD-01

*The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.*

***Function 1***

*No activity for this function.*

***Function 2***

*No activity for this function.*

***Function 3***

*No activity for this function.*

***Function 4***

*The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function for each radio.*

***Function 5***

*The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function.*

***Function 6***

*No activity for this function.*

***Function 7***

*No activity for this function.*

***Function 8***

*No activity for this function.*

***Function 9 & Function 10***

*No activity for this function.*

***Function 11***

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

***Function 12***

*No activity for this function.*

***Function 13***

*The evaluator shall examine the AGD guidance to determine that this same information is present.*

***Function 14***

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

***Function 15***

*No activity for this function.*

***Function 16***

*No activity for this function.*

***Function 17***

*No activity for this function.*

***Function 18***

*The evaluator shall examine the AGD Guidance to determine that it specifies, for at least each category of information selected for Function 18, how to enable and disable display information for that type of information in the locked state.*

***Function 19***

*No activity for this function.*

***Function 20***

*No activity for this function.*

***Function 21***

*No activity for this function.*

***Function 22***

*No activity for this function.*

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

***Function 23 [conditional]***

*No activity for this function.*

***Function 24 [conditional]***

*The AGD guidance will describe how to perform the enable/disable function.*

***Function 25 [conditional]***

*No activity for this function.*

***Function 26 [conditional]***

*No activity for this function.*

***Function 27 [conditional]***

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

***Function 28 [conditional]***

*No activity for this function.*

***Function 29 [conditional]***

*The evaluator shall also verify that the API documentation provided according to Section 5.2.2 Class ADV: Development in* [MDFPPv3.2] *includes any security functions (import, modification, or destruction of the Trust Anchor Database) allowed by applications.*

***Function 30 [conditional]***

*No activity for this function.*

***Function 31 [conditional]***

*The evaluator shall confirm that the AGD guidance describes the procedure for disabling each cellular protocol identified in the TSS.*

***Function 32 [conditional]***

*No activity for this function.*

***Function 33 [conditional]***

*No activity for this function.*

***Function 34 [conditional]***

*No activity for this function.*

***Function 35 [conditional]***

*No activity for this function.*

***Function 36 [conditional]***

*No activity for this function.*

***Function 37 [conditional]***

*No activity for this function.*

***Function 38 [conditional]***

*No activity for this function.*

***Function 39 [conditional]***

*No activity for this function.*

***Function 40 [conditional]***

*No activity for this function.*

***Function 41 [conditional]***

*No activity for this function.*

***Function 42 [conditional]***

*No activity for this function.*

***Function 43 [conditional]***

*No activity for this function.*

***Function 44 [conditional]***

*No activity for this function.*

***Function 45 [conditional]***

*No activity for this function.*

***Function 46 [conditional]***

*No activity for this function.*

***Function 47***

*No activity for this function.*

**Summary**

Guidance to perform each management function is provided throughout [CCGUIDE] , notably in section 3.8 *Security Management Configuration*, which has been assessed by the evaluator while performing other guidance assurance activities. Also, [CCGUIDE] section 3.8 contains table 5 "Required Mobile Device Management Functions" which includes management functions from table 5 "Management Functions" of [ST] and management functions specific to Wi-Fi and VPN. For each of these management functions, table 5 provides pointer to the corresponding section in [CCGUIDE] where guidance can be found. In addition, all these management functions were performed (in accordance to the provided guidance) as part of independent testing.

The following sections provide the management functions and corresponding guidance.

**Function 1**

No assurance activity for this function, however, related guidance is provided in section 5.5.1 of [CCGUIDE] .

**Function 2**

No assurance activity for this function, however, related guidance is provided in section 5.6.3 of [CCGUIDE] .

**Function 3**

No assurance activity for this function, however, related guidance is provided in section 5.3.5 of [CCGUIDE] .

**Function 4**

[CCGUIDE] section 5.6.7 *Enable/Disable Cellular, Wi-Fi, Wi-Fi Hotspot, Bluetooth, NFC, UWB* provides related guidance which states that the TOE devices contain a variety of radios which can be configured by the users or administrators according to the organization's policy.

For the functions with radios, [CCGUIDE] mainly refers to the device user guide [iPhone_UG] for guidance on how to perform the enable/disable functions of the devices. For example, the user is instructed to enable/disable Wi-Fi by following the instructions provided in the [iPhone_UG] : "Safety, handling, and support" >> "View or change cellular data settings on iPhone"

**Function 5**

Section 5.6.6 *Enable/Disable Cameras and Microphones* of [CCGUIDE] provides related guidance. It states the following:

- The cameras and microphones on the iPhone and iPad can be managed across the devices or on a per-app basis.
- Mobile device users can optionally disable the use of the cameras on a per-app basis. This can be done on the iPhone or iPad from Settings >> Privacy >> Camera. If the mobile device administrator has restricted the use of the camera then this functionality will not work.
- Mobile device users can optionally disable the use of the microphones on a per-app basis. This can be done on the iPhone or iPad from Settings >> Privacy >> Microphone.
- Mobile device administrator can optionally disallow camera use across the mobile device by using the key *allowCamera* in the Restrictions Payload.
- Mobile device administrator can optionally disallow camera use on a per app basis using the key Camera in the Privacy Preferences Policy Control Payload.

- Mobile device administrator can optionally disallow microphone use on a per app basis using the key Microphone in the Privacy Preferences Policy Control Payload.

Additional guidance is provided in [DEV_MAN] .

**Function 6**

No assurance activity for this function, however, related guidance is provided in section 5.6.3 of [CCGUIDE] .

**Function 7**

No assurance activity for this function, however, related guidance is provided in section 5.4.3 of [CCGUIDE] .

**Function 8**

No assurance activity for this function, however, related guidance is provided in section 5.6.12 of [CCGUIDE] .

**Function 9**

No assurance activity for this function, however, related guidance is provided in section 5.2.5 of [CCGUIDE] .

**Function 10**

No assurance activity for this function, however, related guidance is provided in section 5.2.5 of [CCGUIDE] .

**Function 11**

Section 5.5.6 *X.509 Certificate Configuration* of [CCGUIDE] for related guidance on certificate management which states the following:

- In the evaluated configuration, users are not allowed to import X.509 certificates into the Trust Anchor Database.
- If allowed (by the administrator) in the evaluated configuration, the TOE users can manually remove certificates installed on their device by choosing Settings >> General >> Profile & Device Management, select a profile, choose More Details, and then choose the appropriate certificate to remove.
- In the evaluated configuration, the user can only remove imported X.509v3 certificates but cannot remove other (pre-installed) X.509v3 certificates in the Trust Anchor Database.
- Administrator configures X.509 certificates using the *Certificate Payload* in a Configuration Profile.
- In the evaluated configuration, the mobile device administrator must disallow the removal of a Certificates Payload by a user in a Configuration Profile.
- Administrators can modify the Trust Anchor Database via a Configuration Profile using either the Certificate Payload or SCEP Payload.
- Administrators can view or remove any installed certificate via the MDM protocol using the *RequestType* key with the content “CertificateList” in a Configuration Profile.

**Function 12**

No assurance activity for this function, however, related guidance is provided in section 5.5.6 *X.509 Certificate Configuration* of [CCGUIDE] .

**Function 13**

Table 6 "Management Functions" of [ST] identifies the management functions that will be enforced by the enterprise once the devices is enrolled. [CCGUIDE] Table 5 "Required Mobile Device Management Functions" provides pointers to the corresponding guidance. While performing related assurance activities the evaluator verified that the information in the TSS is consistent with [CCGUIDE] .

**Function 14**

[CCGUIDE] section 5.6.1 *Install/Remove Apps from the Device* provides related guidance on installing/removing apps. It states the following:

- For enrolled TOE devices, managed apps can be removed by an administrator remotely via the MDM system or when the user removes their own device from the MDM.
- TOE users can install or remove apps from their device in described in [iPhone_UG] . This is of course subject to the organization's policy and the settings of the Restriction Payload in a Configuration Profile.
- Administrator can install apps using an MDM system or Apple Configurator 2 as described in *DeployRef* and [AConfig] , respectively.
- When a managed app is removed from a device, the associated data is removed with it.

**Function 15**

No assurance activity for this function, however, related guidance is provided in section 5.6 *Security Management* of [CCGUIDE] .

**Function 16**

No assurance activity for this function, however, related guidance is provided in section 5.6.1 *Passcode Authentication Configuration* of [CCGUIDE] .

**Function 17**

No assurance activity for this function, however, related guidance is provided in section 5.6.1 *Passcode Authentication Configuration* of [CCGUIDE] .

**Function 18**

Per [ST] , the TOE can be enable/disable display notifications, in the locked state for all notifications.

[CCGUIDE] section 5.6.2 *Configure Access and Notification in Locked State* provides related guidance which states the following:

- Certain display notifications can be set when the mobile device is locked. These notifications can be enable/disable via the device settings and as described in [iPhone_UG]
- Administrator can restrict user from viewing past notifications by disabling Notification history setting the *allowLockScreenNotificationsView* key in the Restrictions Payload in a Configuration Profile.
- Likewise, administrator can disable displaying notifications for applications by setting the *ShowInLockScreen* key in a Notification Payload in a Configuration Profile.
- Once the notification settings have been implemented by the mobile device administrator, the allowNotificationsModification key in the Restrictions Payload must be set to ‘true’ if the settings are not allowed to be modified.

**Function 19**
No assurance activity for this function, however, related guidance is provided in section 5.4.1 *Data-At-Rest (DAR) Protection Configuration* of [CCGUIDE] .

**Function 20**
No assurance activity for this function, however, related guidance is provided in section 5.4.1 *Data-At-Rest (DAR) Protection Configuration* of [CCGUIDE] .

**Function 21**
No assurance activity for this function, however, related guidance is provided in section 5.6.8 *Enable/Disable Location Service* of [CCGUIDE] .

**Function 22**
No assurance activity for this function, however, related guidance is provided in section 5.5.3 *Biometric Authentication Factor* of [CCGUIDE] .

**Function 23 (Conditional)**
No assurance activity for this function, however, related guidance is provided in section 5.5.6 *5.5.6 X.509 Certificate Configuration* of [CCGUIDE] .

**Function 24 (Conditional)**
The ST does not select this function.

**Function 25 (Conditional)**
The ST does not select this function.

**Function 26 (Conditional)**
The ST does not select this function.

**Function 27 (Conditional)**
The ST does not select this function.

**Function 28 (Conditional)**
No assurance activity for this function, however, related guidance is provided in section 5.4.3 *Wiping of Protected Data* of [CCGUIDE] .

**Function 29 (Conditional)**
The ST does not select this function.

**Function 30 (Conditional)**
No assurance activity for this function, however, related guidance is provided in section 5.5.6 *X.509 Certificate Configuration* of [CCGUIDE] .

**Function 31 (Conditional)**
The ST does not select this function.

**Function 32 (Conditional)**
The ST does not select this function.

**Function 33 (Conditional)**
The ST does not select this function.

**Function 34 (Conditional)**
The ST does not select this function.

**Function 35 (Conditional)**
The ST does not select this function.

**Function 36 (Conditional)**
No assurance activity for this function, however, related guidance is provided in section 5.6.5 *Access Banner Configuration* of [CCGUIDE] .

**Function 37 (Conditional)**
No assurance activity for this function, however, related guidance is provided in section 6.3 *Configure the Auditable Items* of [CCGUIDE] .

**Function 38 (Conditional)**
The ST does not select this function.

**Function 39 (Conditional)**
The ST does not select this function.

**Function 40 (Conditional)**
The ST does not select this function.

**Function 41 (Conditional)**
The ST does not select this function.

**Function 42 (Conditional)**
The ST does not select this function.

**Function 43 (Conditional)**
The ST does not select this function.

**Function 44 (Conditional)**
No assurance activity for this function, however, related guidance is provided in section 4.3.4 *Device Unenrollment Prevention* of [CCGUIDE] .

**Function 45 (Conditional)**
The ST does not select this function.

**Function 46 (Conditional)**
The ST does not select this function.

**Function 47 (Conditional)**
The ST does not select this function.

## Test Assurance Activities

### Assurance Activity AA-FMT_SMF_EXT.1-ATE-01

*Test activities specified below shall take place in the test environment described in the evaluation activity for FPT_TUD_EXT.1.*

***Function 1***

- ***Test 1:*** *The evaluator shall exercise the TSF configuration as the administrator and perform positive and negative tests, with at least two values set for each variable setting, for each of the following:*
  - *minimum password length*
  - *minimum password complexity*
  - *maximum password lifetime*

***Function 2***

- ***Test 2:*** *The evaluator shall exercise the TSF configuration as the administrator. The evaluator shall perform positive and negative tests, with at least two values set for each variable setting, for each of the following:*
  - *screen-lock enabled/disabled*
  - *screen lock timeout*
  - *number of authentication failures (may be combined with test for FIA_AFL_EXT.1)*

***Function 3***

- ***Test 3:*** *The evaluator shall perform the following tests:*

a. *The evaluator shall exercise the TSF configuration to enable the VPN protection. These configuration actions must be used for the testing of the FDP_IFC_EXT.1.1 requirement.*

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

c. *[conditional] If "per-groups of application basis" is selected, the evaluator shall create two applications and the applications shall be placed into different groups. Enable one application group to use the VPN and the other to not use the VPN. The evaluator shall exercise each application (attempting to access network resources; for example, by browsing different websites) individually while capturing packets from the TOE. The evaluator shall verify from the packet capture that the traffic from the application in the VPN-enabled group is encapsulated in IPsec and that the traffic from the application in the VPN-disabled group is not encapsulated in IPsec.*

***Function 4***

*The evaluator shall ensure that minimal signal leakage enters the RF shielded enclosure (i.e. Faraday bag, Faraday box, RF shielded room) by performing the following steps:*

- *Step 1: Place the antenna of the spectrum analyzer inside the RF shielded enclosure.*
- *Step 2: Enable "Max Hold" on the spectrum analyzer and perform a spectrum sweep of the frequency range between 300MHz – 6000MHz, in I KHz steps (this range should encompass 802.11, 802.15, GSM, UMTS, and LTE). This range will not address NFC 13.56MHz, another test should be set up with similar constraints to address NFC.*

*If power above -90 dBm is observed, the Faraday box has too great of signal leakage and shall not be used to complete the test for Function 4.*

- ***Test 4:** The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable and disable the state of each radio (e.g. Wi-Fi, cellular, NFC, Bluetooth). Additionally, the evaluator shall repeat the steps below, booting into any auxiliary boot mode supported by the device. For each radio, the evaluator shall:*
  - *Step 1: Place the antenna of the spectrum analyzer inside the RF shielded enclosure. Configure the spectrum analyzer to sweep desired frequency range for the radio to be tested (based on range provided in the TSS)). The ambient noise floor shall be set to -110dBm. Place the TOE into the RF shielded enclosure to isolate them from all other RF traffic.*
  - *Step 2: The evaluator shall create a baseline of the expected behavior of RF signals. The evaluator shall power on the device, ensure the radio in question is enabled, power off the device, enable "Max Hold" on the spectrum analyzer and power on the device. The evaluator shall wait 2 minutes at each Authentication Factor interface prior to entering the necessary password to complete the boot process, waiting 5 minutes after the device is fully booted. The evaluator shall observe that RF spikes are present at the expected uplink channel frequency. The evaluator shall clear the "Max Hold" on the spectrum analyzer.*
  - *Step 3: The evaluator shall verify the absence of RF activity for the uplink channel when the radio in question is disabled. The evaluator shall complete the following test five times. The evaluator shall power on the device, ensure the radio in question is disabled, power off the device, enable "Max Hold" on the spectrum analyzer and power on the device. The evaluator shall wait 2 minutes at each Authentication Factor interface prior to entering the necessary password to complete the boot process, waiting 5 minutes after the device is fully booted. The evaluator shall clear the "Max Hold" on the spectrum analyzer. If the radios are used for device initialization, then a spike of RF activity for the uplink channel can be observed initially at device boot. However, if a spike of RF activity for the uplink channel of the specific radio frequency band is observed after the device is fully booted or at an Authentication Factor interface it is deemed that the radio is enabled.*

***Function 5***

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

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

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

c. *[conditional] If "per-groups of application basis" is selected, the evaluator shall create two applications and the applications shall be placed into different groups. Enable one group to access the A/V device and the other to not access the A/V device. The evaluator shall exercise each application attempting to access the A/V device individually. The evaluator shall verify that the application in the enabled group is able to access the A/V device and the application in the disabled group is not able to access the A/V device.*

***Function 6***

- ***Test 6:*** *The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to transition to a locked state, and verify that the device transitions to the locked state upon command.*

***Function 7***

- ***Test 7:*** *The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to perform a wipe of protected data. The evaluator must ensure that this management setup is used when conducting the Evaluation Activities in FCS_CKM_EXT.5.*

***Function 8***

- ***Test 8:*** *The evaluator shall exercise the TSF configuration as the administrator to restrict particular applications, sources of applications, or application installation according to the AGD guidance. The evaluator shall attempt to install unauthorized applications and ensure that this is not possible. The evaluator shall, in conjunction, perform the following specific tests:*

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

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

***Function 9 & Function 10***

- ***Test 9:*** *The test of these functions is performed in association with FCS_STG_EXT.1.*
- ***Test 10:*** *The test of these functions is performed in association with FCS_STG_EXT.1.*

***Function 11***

- ***Test 11:*** *The evaluator shall import certificates according to the AGD guidance as the user and/or as the administrator, as determined by the administrative guidance. The evaluator shall verify that no errors occur during import. The evaluator should perform an action requiring use of the X.509v3 certificate to provide assurance that installation was completed properly.*

***Function 12***

- ***Test 12:*** *The evaluator shall remove an administrator-imported certificate and any other categories of certificates included in the assignment of function 14 from the Trust Anchor Database according to the AGD guidance as the user and as the administrator.*

***Function 13***

- ***Test 13:*** *The evaluator shall verify that user approval is required to enroll the device into management.*

***Function 14***

- ***Test 14:*** *The evaluator shall attempt to remove applications according to the AGD guidance and verify that the TOE no longer permits users to access those applications or their associated data.*

***Function 15***

- ***Test 15:*** *The evaluator shall attempt to update the TSF system software following the procedures in the AGD guidance and verify that updates correctly install and that the version numbers of the system software increase.*

***Function 16***

- ***Test 16:*** *The evaluator shall attempt to install an application following the procedures in the AGD guidance and verify that the application is installed and available on the TOE.*

***Function 17***

- ***Test 17:*** *The evaluator shall attempt to remove any Enterprise applications from the device by following the administrator guidance. The evaluator shall verify that the TOE no longer permits users to access those applications or their associated data.*

***Function 18***

- ***Test 18:*** *For each category of information listed in the AGD guidance, the evaluator shall verify that when that TSF is configured to limit the information according to the AGD, the information is no longer displayed in the locked state.*

***Function 19***

- ***Test 19:*** *The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable system-wide data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all Evaluation Activities for DAR (FDP_DAR) are conducted with the device in this configuration.*

***Function 20***

- ***Test 20:*** *The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable removable media's data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all Evaluation Activities for DAR (FDP_DAR) are conducted with the device in this configuration.*

***Function 21***

- ***Test 21:*** *The evaluator shall perform the following tests.*
  a. *The evaluator shall enable location services device-wide and shall verify that an application (such as a mapping application) is able to access the TOE's location information. The evaluator shall disable location services device-wide and shall verify that an application (such as a mapping application) is unable to access the TOE's location information.*
  b. *[conditional] If "per-app basis" is selected, the evaluator shall create two applications and enable one to use access the location services and the other to not access the location services. The evaluator shall exercise each application attempting to access location services individually. The evaluator shall verify that the enabled application is able to access the location services and the disabled application is not able to access the location services.*

***Function 22***

- ***Test 22:*** *[The evaluator shall verify that the TSS states if the TOE supports a BAF and/or hybrid authentication.] If the TOE does not include a BAF and/or hybrid authentication this test is implicitly met.*
  a. *[conditional] If a BAF is selected the evaluator shall verify that the TSS describes the procedure to enable/disable the BAF. If the TOE includes multiple BAFs, the evaluator shall verify that the TSS describes how to enable/disable each BAF, specifically if the different modalities can be individually enabled/disabled. The evaluator shall configure the TOE to allow each supported BAF to authenticate and verify that successful authentication can be achieved using the BAF. The evaluator shall configure the TOE to disable the use of each supported BAF for authentication and confirm that the BAF cannot be used to authenticate.*
  b. *[conditional] If "Hybrid" is selected the evaluator shall verify that the TSS describes the procedure to enable/disable the hybrid (biometric credential and PIN/password) authentication. The evaluator shall configure the TOE to allow hybrid authentication to authenticate and confirm that successful authentication can be achieved using the hybrid authentication. The evaluator shall configure the TOE to disable the use of hybrid authentication and confirm that the hybrid authentication cannot be used to authenticate.*

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

***Function 23 [conditional]***

- ***Test 23:*** *The test of this function is performed in conjunction with FIA_X509_EXT.2.2, FCS_TLSC_EXT.1.3 in the Package for Transport Layer Security.*

***Function 24 [conditional]***

- ***Test 24:*** *The evaluator shall exercise the TSF configuration to enable and disable data transfer capabilities over each externally accessible hardware ports (e.g. USB, SD card, HDMI) listed by the ST author. The evaluator shall use test equipment for the particular interface to ensure that no low-level signaling is occurring on all pins used for data transfer when they are disabled. For each disabled data transfer capability, the evaluator shall repeat this test by rebooting the device into the normal operational mode and verifying that the capability is disabled throughout the boot and early execution stage of the device.*

***Function 25 [conditional]***

- ***Test 25:*** *The evaluator shall attempt to disable each listed protocol in the assignment. The evaluator shall verify that remote devices can no longer access the TOE or TOE resources using any disabled protocols.*

***Function 26 [conditional]***

- ***Test 26:*** *The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable and disable any developer mode. The evaluator shall test that developer mode access is not available when its configuration is disabled. The evaluator shall verify the developer mode remains disabled during device reboot.*

***Function 27 [conditional]***

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

***Function 28 [conditional]***

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

***Function 29 [conditional]***

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

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

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

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

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

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

***Function 30 [conditional]***

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

***Function 31 [conditional]***

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

***Function 32 [conditional]***

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

***Function 33 [conditional]***

- ***Test 33:*** *The test of this function is performed in conjunction with FPT_TUD_EXT.5.1.*

***Function 34 [conditional]***

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

***Function 35 [conditional]***

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

***Function 36 [conditional]***

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

***Function 37 [conditional]***

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

***Function 38 [conditional]***

- ***Test 38:*** *The test of this function is performed in conjunction with FPT_NOT_EXT.2.1.*

***Function 39 [conditional]***

- ***Test 39:*** *The evaluator shall perform the following tests based on the selections made in the table:*
  a. *[conditional] The evaluator shall disable USB mass storage mode, attach the device to a computer, and verify that the computer cannot mount the TOE as a drive. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.*
  b. *[conditional] The evaluator shall disable USB data transfer without user authentication, attach the device to a computer, and verify that the TOE requires user authentication before the computer can access TOE data. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.*
  c. *[conditional] The evaluator shall disable USB data transfer without connecting system authentication, attach the device to a computer, and verify that the TOE requires connecting system authentication before the computer can access TOE data. The evaluator shall then connect the TOE to another computer and verify that the computer cannot access TOE data. The evaluator shall then connect the TOE to the original computer and verify that the computer can access TOE data.*

***Function 40 [conditional]***

- ***Test 40:*** *If "all applications" is selected, the evaluator shall disable each selected backup location in turn and verify that the TOE cannot complete a backup. The evaluator shall then enable each selected backup location in turn and verify that the TOE can perform a backup. If "selected applications" is selected, the evaluator shall disable each selected backup location in turn and verify that for the selected application the TOE prevents backup from occurring. The evaluator shall then enable each selected backup location in turn and verify that for the selected application the TOE can perform a backup. If "selected groups of applications" is selected, the evaluator shall disable each selected backup location in turn and verify that for a group of applications the TOE prevents the backup from occurring. The evaluator shall then enable each selected backup location in turn and verify for the group of application the TOE can perform a backup. If "configuration data" is selected, the evaluator shall disable each selected backup location in turn and verify that the TOE prevents the backup of configuration data from occurring. The evaluator shall then enable each selected backup location in turn and verify that the TOE can perform a backup of configuration data.*

***Function 41 [conditional]***

- ***Test 41:*** *The evaluator shall perform the following tests based on the selections in Function 41.*
  a. *[conditional] The evaluator shall enable hotspot functionality with each of the of the support authentication methods. The evaluator shall connect to the hotspot with another device and verify that the hotspot functionality requires the configured authentication method.*
  b. *[conditional] The evaluator shall enable USB tethering functionality with each of the of the support authentication methods. The evaluator shall connect to the TOE over USB with another device and verify that the tethering functionality requires the configured authentication method.*

***Function 42 [conditional]***

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

***Function 43 [conditional]***

- ***Test 43:*** *The evaluator shall set a policy to cause a designated application to be placed into a particular application group. The evaluator shall then install the designated application and verify that it was placed into the correct group.*

***Function 44 [conditional]***

- ***Test 44:*** *The evaluator shall attempt to unenroll the device from management and verify that the steps described in FMT_SMF_EXT.2.1 are performed. This test should be performed in conjunction with the FMT_SMF_EXT.2.1 evaluation activity.*

***Function 45 [conditional]***

- ***Test 45:*** *The evaluator shall verify that the TSS contains guidance to configure the VPN as Always-On. The evaluator shall configure the VPN as Always-On and perform the following test.*
  a. *The evaluator shall verify that when the VPN is connected all traffic is routed through the VPN. This test is performed in conjunction with FDP_IFC_EXT.1.1.*
  b. *The evaluator shall verify that when the VPN is not established, that no traffic leaves the device. The evaluator shall ensure that the TOE has network connectivity and that the VPN is established. The evaluator shall use a packet sniffing tool to capture the traffic leaving the TOE. The evaluator shall disable the VPN connection on the server side. The evaluator shall perform actions with the device such as navigating to websites, using provided applications, and accessing other Internet resources and verify that no traffic leaves the device.*
  c. *The evaluator shall verify that the TOE has network connectivity and that the VPN is established. The evaluator shall disable network connectivity (i.e. Airplane Mode) and verify that the VPN disconnects. The evaluator shall re-establish network connectivity and verify that the VPN automatically reconnects.*

***Function 46 [conditional]***

- ***Test 46:*** *The evaluator shall verify that the TSS describes the procedure to revoke a biometric credential stored on the TOE. The evaluator shall configure the TOE to use BAF and confirm that the biometric can be used to authenticate to the device. The evaluator shall revoke the biometric credential's ability to authenticate to the TOE and confirm that the same BAF cannot be used to authenticate to the device.*

***Function 47***

- ***Test 47:*** *The evaluator shall design and perform tests to demonstrate that the function may be configured and that the intended behavior of the function is enacted by the TOE.*

**Summary**

**Function 1**

Test 1: The evaluator used the Apple Configurator 2 (AC2) to set up the password policy for the TOE. The evaluator loaded the profile on the TOE and attempted to change the password to a password not fulfilling the password policy in both length and complexity. The evaluator verified that the TOE enforced the password policy for each change.

**Function 2**

Test 2: The evaluator used the AC2 to configure auto-lock on the TOE, loaded the profile on the TOE and verified that it was not possible to select a longer auto-lock timeout on the TOE. The evaluator then measured how long it took for the TOE to timeout and verified that it was consistent with the policy.

**Function 3**

Tests 3a & 3b: Please see FDP_IFC_EXT.1{MDF}.

Tests 3c & 3d are not applicable because they are not claimed in [ST].

**Function 4**

Test 4: The evaluator put the TOE in a shielded room, set the radio frequency scanner to sweep a defined frequency, and verified that the power emanating from the TOE had a baseline measurement showing that a transmission spike is about 40dBm or higher (from -90dBm to -40dBm).

**Function 5**

Test 5a (camera): The evaluator disabled the camera app on the TOE and verified that the camera app disappeared from the main page. Also, the evaluator installed a QR scanner app and verified that the camera was not accessible through the QR scanner app.

Test 5a (aux boot): The evaluator powered off the devices and powered them on using a special combination of keys. The evaluator verified that the iTunes logo was displayed.

Test 5b (camera): The evaluator enabled the camera on the TOE, verified that the app is present, opened he QR scanner app and verified that it prompts for camera access permission. The evaluator accepted and verified that the app was authorized by the OS in the settings to use the camera and that the QR scanner app could make use of the camera. The evaluator then removed the camera usage permission from the QR scanner app and verified that the app could not use the camera anymore.

Tests 5c & 5d are not applicable since these functions are not claimed in [ST].

**Function 6**

Test 6: The evaluator pressed the power button and verified that the phone was locked by pressing the home button and confirming that the device requests a password. The evaluator then enrolled the devices into iCloud lost mode and triggered a remote lock of the device. Again, the evaluator confirmed that the device was locked.

**Function 7**

Test 7: The evaluator went to
`Settings -> General -> Reset -> Erase All Content and Settings`
and verified that all the data was wiped from the device.

**Function 8**

Test 8: The evaluator created a profile through AC2 that disabled installation of applications. The evaluator loaded that profile and verified that the App Store was not present anymore.

**Function 9**

Test 9 is performed in association with FCS_STG_EXT.1.

**Function 10**

Test 10 is performed in association with FCS_STG_EXT.1.

**Function 11**

Test 11: Please see tests for FCS_TLSC_EXT.1 {MDFPP} and FCS_TLSC_EXT.1/WLAN {WLAN}.

**Function 12**

Test 12: The evaluator installed a profile containing certificates and tried to delete/remove it. The TOE either refused or prompted for a password/passcode. These steps are performed numerous time during testing of FIA_X509_EXT.1.

**Function 13**

Test 13: The evaluator enrolled the TOE into MDM 3 different ways during testing for FMT_SMF_EXT.4 {MDF} Test 3. He verified during enrollment that the user is required to provide a username and passcode which indicates user consent to enroll the device into management.

**Function 14**

Test 14: The evaluator used the App Store application to install Google Chrome and used the app to browse some website. The evaluator then remove the app from the device and verified that the app was indeed removed and not usable.

**Function 15**

Test 15: The evaluator checked the current version using
`Settings -> General -> About -> Software Version`
and verified that it is an older OS release. The evaluator installed the update for 15.1.0 using
`Settings -> General -> Software Update`
After completion, checking the version again as before, the evaluator confirmed the device now runs 15.1.0.

**Function 16**

Test 16: Performed in conjunction with Test 14.

**Function 17**

Test 17: The evaluator compiled and archived a test application and generated an IPA archive file using Xcode. The evaluator deployed this archive on the TOE using the AC2, verified that the app was installed, and executed properly. The evaluator then used the AC2 to remove the app from the TOE.

**Function 18**

Tests 18a-18e are not applicable since the functions are not claimed in [ST].

Test 18f: The evaluator used AC2 to disable notifications on the lock screen. The evaluator created a calendar event and ensured that no notification was displayed on the lock screen, but that the notification was indeed displayed once the device was unlocked.

**Function 19**

Test 19: Please see FDP_DAR_EXT.2.1 Test 1.

**Function 20**

On a PC or Mac, the tester formats an external storage device using the APFS encrypted format, setting a password. Then create a directory called "Test Directory" and a text file called "Atsec-Tester-File.txt" on the external storage device. Remove the device from the PC or Mac, insert device into the TOE and open Files app on the TOE. Verify that a password must be entered before the storage device is accessible to the TOE and enter the password. Verify that the directory and file are now visible. Create a new directory called <device-name>. Move the text file from Test Directory to the new directory. Remove device from TOE. Insert device into PC or Mac. Verify that the encrypted file was moved to the new directory.

**Function 21**

Test 21a: The evaluator enabled the location services on the TOE, opened the Maps app and verified that the TOE could find its location. The evaluator disabled the location services and verified that the TOE was unable to find its location.

Test 21b: The evaluator enabled the location services, opened Maps, and verified that the app could find the location of the device. The evaluator then disabled the location services specifically for the Maps app, opened Maps, and verified that Maps could not find the location.

**Function 22**

Test 22: The evaluator enabled the biometric configuration for the devices (either Touch ID or Face ID), locked the device, unlocked the device, removed the biometric configuration, and verified that he could not unlock the device anymore using biometric authentication factors.

**Function 23**

Test 23: Please see the test for FIA_X509_EXT.2-WLAN {WLAN}.

**Function 24**

Test 24 is not applicable because Function 24 is not claimed in [ST].

**Function 25**

Test 25 is not applicable because Function 25 is not claimed in [ST].

**Function 26**

Test 26 is not applicable because Function 26 is not claimed in [ST].

**Function 27**

Test 27 is not applicable because Function 27 is not claimed in [ST].

**Function 28**

There is no difference between enterprise applications and their data and regular apps. Thus Test 16 covers this test for application data. For a general wipe, FIA_AFL_EXT.1 Test 7 and FCS_CKM_EXT.4 demonstrate that all data is wiped on the device.

**Function 29**

Test 29 is not applicable because Function 29 is not claimed in [ST].

**Function 30**

Please see the test for FIA_X509_EXT.2.

**Function 31**

Test 31 is not applicable because Function 31 is not claimed in [ST].

**Function 32**

Test 32 is not applicable because Function 32 is not claimed in [ST].

**Function 33**

Test 33 is not applicable because Function 33 is not claimed in [ST].

**Function 34**

Test 34 is not applicable because Function 34 is not claimed in [ST].

**Function 35**

Test 35 is not applicable because Function 35 is not claimed in [ST].

**Function 36**

Test 36: Please see the test for FTA_TAB.1.

**Function 37**

Test 37: Please see the test for FAU_SEL.1.

**Function 38**

Test 38 is not applicable because Function 38 is not claimed in [ST].

**Function 39**

Test 39 is not applicable because Function 39 is not claimed in [ST].

**Function 40**

Test 40 is not applicable because Function 40 is not claimed in [ST].

**Function 41**

Test 41 is not applicable because Function 41 is not claimed in [ST].

**Function 42**

Test 42 is not applicable because Function 42 is not claimed in [ST].

**Function 43**

Test 43 is not applicable because Function 43 is not claimed in [ST].

**Function 44**

Test 44: Please see FTP_ITC_EXT.1(2) Test 2.

**Function 45**

Test 45 a,b,c: Please see the test and results for FDP_IFC_EXT.1.1.

Test 45b: The traffic is generated by pinging the TOE demonstrating that the device will not return an ICMP echo reply, but only attempts to establish an IPsec tunnel (i.e. IKE traffic).

Test 45c: Enabling and disabling of WLAN connectivity is performed, including a review of automated reconnect.

**Function 46**

Test 46 is not applicable because Function 46 is not claimed in [ST].

**Function 47**

Test 47a (microphone): The evaluator disabled the microphone on the TOE, installed Google Chrome, and tried to perform a search with the microphone. The evaluator verified that this operation failed.

Test 47b (microphone): The evaluator enabled the microphone on the TOE and used Chrome to search using the voice. The evaluator then verified that the Chrome app had the microphone access permission in the settings. The evaluator then removed this permission and verified, by using the Chrome app, that the microphone could not be used.

## 2.2.5.5 Specification of Management Functions (FMT_SMF_EXT.1/BT)

### TSS Assurance Activities

#### Assurance Activity AA-FMT_SMF_EXT.1-BT-ASE-01

*The evaluator shall ensure that the TSS includes a description of the Bluetooth profiles and services supported and the Bluetooth security modes and levels supported by the TOE.*

*If function BT-4, "Allow/disallow additional wireless technologies to be used with Bluetooth," is selected, the evaluator shall verify that the TSS describes any additional wireless technologies that may be used with Bluetooth, which may include Wi-Fi with Bluetooth High Speed and/or NFC as an Out of Band pairing mechanism.*

*If function BT-5, "Configure allowable methods of Out of Band pairing (for BR/EDR and LE)," is selected, the evaluator shall verify that the TSS describes when Out of Band pairing methods are allowed and which ones are configurable.*

*If function BT-8, "Disable/enable the Bluetooth services and/or profiles available on the OS (for BR/EDR and LE)," is selected, the evaluator shall verify that all supported Bluetooth services are listed in the TSS as manageable and, if the TOE allows disabling by application rather than by service name, that a list of services for each application is also listed.*

*If function BT-9, "Specify minimum level of security for each pairing (for BR/EDR and LE)," is selected, the evaluator shall verify that the TSS describes the method by which the level of security for pairings are managed, including whether the setting is performed for each pairing or is a global setting.*

**Summary**

The ST selects BT-1 in FMT_SMF_EXT.1.1/BT.

Section 8.9.2 *Bluetooth* states the following:

- Connections via Bluetooth/LE are secured using 128-bit AES-CCM-128 mode.
- A local database is kept of all Bluetooth device addresses for paired devices which is checked prior to any automatic connection attempt.
- Bluetooth devices may not establish more than one connection.
- Multiple connection attempts (i.e., pairing and session initialization attempts) from the same BD_ADDR for an established connection will be discarded
- The TOE 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.
- Users can pair their device with a remote Bluetooth device using the 'Set up Bluetooth Device' option from the Bluetooth status menu. Manual authorization is implicitly configured since pairing can only occur when explicitly authorized through the Settings»Bluetooth interface.

## Guidance Assurance Activities

### Assurance Activity AA-FMT_SMF_EXT.1-BT-AGD-01

*The evaluator shall ensure that the management functions defined in the PP-Module are described in the guidance to the same extent required for the Base-PP management functions.*

**Summary**

The evaluator noted that administrator-provided management functions defined in [PP_MDF_V3.2] and [MOD_BT_V1.0] are provided throughout [CCGUIDE] . A high-level summary is provided in [CCGUIDE] Table 5 "Required Mobile Device Management Functions" where for each management function, a pointer to the related guidance is provided. Also, these management functions are already assessed in prior work units of this report, in particular, the work units of AA-FMT_SMF_EXT.1-AGD.

Additionally, [CCGUIDE] provides Table 4 "SFR Configuration Requirements" outlining for each function (expressed in the form of an SFR and its description) whether configuration is required.

## Test Assurance Activities

### Assurance Activity AA-FMT_SMF_EXT.1-BT-ATE-01

*The evaluator shall use a Bluetooth-specific protocol analyzer to perform the following tests:*

- ***Test 1:** The evaluator shall disable the Discoverable mode and shall verify that other Bluetooth BR/EDR devices cannot detect the TOE. The evaluator shall use the protocol analyzer to verify that the TOE does not respond to inquiries from other devices searching for Bluetooth devices. The evaluator shall enable Discoverable mode and verify that other devices can detect the TOE and that the TOE sends response packets to inquiries from searching devices.*
  *The following tests are conditional on if the corresponding function is included in the ST:*

- ***Test 2:*** *(conditional): The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name, change the Bluetooth device name, and verify that the Bluetooth traffic from the TOE lists the new name. The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name for BR/EDR and LE. The evaluator shall change the Bluetooth device name for LE independently of the device name for BR/EDR. The evaluator shall verify that the Bluetooth traffic from the TOE lists the new name.*
- ***Test 3:*** *(conditional): The evaluator shall disable Bluetooth BR/EDR and enable Bluetooth LE. The evaluator shall examine Bluetooth traffic from the TOE to confirm that only Bluetooth LE traffic is present. The evaluator shall repeat the test with Bluetooth BR/EDR enabled and Bluetooth LE disabled, confirming that only Bluetooth BR/EDR is present.*
- ***Test 4:*** *(conditional): For each additional wireless technology that can be used with Bluetooth as claimed in the ST, the evaluator shall revoke Bluetooth permissions from that technology. If the set of supported wireless technologies includes Wi-Fi, the evaluator shall verify that Bluetooth High Speed is not able to send Bluetooth traffic over Wi-Fi when disabled. If the set of supported wireless technologies includes NFC, the evaluator shall verify that NFC cannot be used for pairing when disabled. For any other supported wireless technology, the evaluator shall verify that it cannot be used with Bluetooth in the specified manner when disabled. The evaluator shall then re-enable all supported wireless technologies and verify that all functionality that was previously unavailable has been restored.*
- ***Test 5:*** *(conditional): The evaluator shall attempt to pair using each of the Out of Band pairing methods, verify that the pairing method works, iteratively disable each pairing method, and verify that the pairing method fails.*
- ***Test 6:*** *(conditional): The evaluator shall enable Advertising for Bluetooth LE, verify that the advertisements are captured by the protocol analyzer, disable Advertising, and verify that no advertisements from the device are captured by the protocol analyzer.*
- ***Test 7:*** *(conditional): The evaluator shall enable Connectable mode and verify that other Bluetooth devices may pair with the TOE and (if the devices were bonded) re-connect after pairing and disconnection. For BR/EDR devices: The evaluator shall use the protocol analyzer to verify that the TOE responds to pages from the other devices and permits pairing and re-connection. The evaluator shall disable Connectable mode and verify that the TOE does not respond to pages from remote Bluetooth devices, thereby not permitting pairing or re-connection. For LE: The evaluator shall use the protocol analyzer to verify that the TOE sends connectable advertising events and responds to connection requests. The evaluator shall disable Connectable mode and verify that the TOE stops sending connectable advertising events and stops responding to connection requests from remote Bluetooth devices.*
- ***Test 8:*** *(conditional): For each supported Bluetooth service and/or profile listed in the TSS, the evaluator shall verify that the service or profile is manageable. If this is configurable by application rather than by service and/or profile name, the evaluator shall verify that a list of services and/or profiles for each application is also listed.*
- ***Test 9:*** *(conditional): The evaluator shall allow low security modes/levels on the TOE and shall initiate pairing with the TOE from a remote device that allows only something other than Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR), or Security Mode 1/Level 3 (for LE). (For example, a remote BR/EDR device may claim Input/Output capability "NoInputNoOutput" and state that man-in-the-middle (MiTM) protection is not required. A remote LE device may not support encryption.) The evaluator shall verify that this pairing attempt succeeds due to the TOE falling back to the low security mode/level. The evaluator shall then remove the pairing of the two devices, prohibit the use of low security modes/levels on the TOE, then attempt the connection again. The evaluator shall verify that the pairing attempt fails. With the low security modes/levels disabled, the evaluator shall initiate pairing from the TOE to a remote device that supports Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR) or Security Mode 1/Level 3 (for LE). The evaluator shall verify that this pairing is successful and uses the high security mode/level.*

**Summary**

Test 1: The evaluator used the hcitool to obtain the device ID of the TOE. The evaluator then disabled Bluetooth on the TOE and used hcidump to watch for connections. The evaluator verified that the TOE was not seen. The evaluator then tried to force a connection to the TOE using its Bluetooth ID and verified that the connection did not succeed and that no Bluetooth packets were recorded by the packet sniffer tool.

Tests 2-9 are not applicable because only function 1 is selected in [ST].

## 2.2.5.6 Specification of Management Functions (Wireless LAN) (FMT_SMF_EXT.1/WLAN)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

#### Assurance Activity AA-FMT_SMF_EXT.1-WLAN-AGD-01

*The evaluator shall check to make sure that every management function mandated by the EP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.*

#### Summary

The evaluator noted that administrator-provided management functions defined in [PP_MDF_V3.2] and [PP_WLAN_CLI_EP_V1.0] are provided throughout [CCGUIDE] . A high-level summary is provided in [CCGUIDE] Table 5 "Required Mobile Device Management Functions" where for each management function, provides a pointer to the related guidance. These management functions are already assessed the related work units of the Base-PP, [PP_MDF_V3.2] particularly the work units of AA-FMT_SMF_EXT.1-AGD-01.

Additionally, [CCGUIDE] provides Table 4 "SFR Configuration Requirements" outlining for each function (expressed in the form of an SFR and its description) whether configuration is required.

### Test Assurance Activities

#### Assurance Activity AA-FMT_SMF_EXT.1-WLAN-ATE-01

*The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE and testing each option listed in the requirement above.*

*Note that the testing here may be accomplished in conjunction with the testing of other requirements, such as FCS_TLSC_EXT and FTA_WSE_EXT.*

#### Summary

Test 1: For each configured cipher, the tester configured OpenSSL to only accept one cipher suite and used the TOE to connect to http://server. The tester verified that the access was possible.

Test 2: The tester verified that accessing http://server is possible and accessing http://10.0.0.1 is not when using a server certificate with the option "Server Authentication".

## 2.2.5.7 Specification of Remediation Actions (FMT_SMF_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FMT_SMF_EXT.2-ASE-01

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

#### Summary

Section 8.6.4 *Unenrollment* in the TSS of [ST] describes the unenrollment process.

This section references the user guide *Configuration Profile Reference* , [DEV_MAN] , which describes the unenrollment options. The optional key PayLoadRemovalDisallowed, if present and set to true, prevents the user from deleting the configuration profile, unless this profile has a password that the user can provide.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FMT_SMF_EXT.2-ATE-01

*The evaluator shall use the test environment to iteratively configure the device to perform each remediation action in the selection. The evaluator shall configure the remediation action per how the TSS states it is provided to the administrator. The test environment could be a MDM agent application, but can also be an application with administrator access.*

**Summary**

The evaluator used Profile Manager to go to
`Device Groups -> testgroup1 -> Settings`
and unselect "Prevent unenrollment". On the device, the evaluator went to
`Settings -> General -> Device Management -> Remote Management`
and removed the profile and verified it was removed.

## 2.2.5.8 Specification of Management Functions (FMT_SMF_EXT.4)

### TSS Assurance Activities

#### Assurance Activity AA-FMT_SMF_EXT.4-MDMA-ASE-01

*The evaluator shall verify that the any assigned functions are described in the TSS and that these functions are documented as supported by the platform. The evaluator shall examine the TSS to verify that any differences between management functions and policies for each supported mobile device are listed.*

*The evaluator shall verify that the TSS describes the methods in which the MDM Agent can be enrolled.*

*The TSS description shall make clear if the MDM Agent supports multiple interfaces for enrollment and configuration (for example, both remote configuration and local configuration).*

**Summary**

The ST specifies the following management functions in FMT_SMF_EXT.4:

- Enroll in management
- configure whether users can unenroll from management

Since all the mobile devices specified in this [ST] use the TOE OS, there are no differences between supported management functions and policies between the different mobile devices. The supported management functions for the TOE OS are described in [DEV_MAN] . The evaluator verified that the functions provided in FMT_SMF_EXT.4.1 are all configurable through the configuration profiles defined by the administrator (or MDM administrator) of the TOE.

Section 8.5.3 *MDM Server Reference ID* in the TSS of [ST] describes how to set up the MDM server for enrollment. Section 8.6.1 *Enrollment* describes how a device enrolls with an MDM server.

Section 8.6.1 describes that a device can enroll in multiple ways with an MDM server as follows.

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

Section 8.5.3 also describes that the TOE devices automatically connect to the MDM Server during setup if the device is enrolled into the Apple Business Manager (ABM) and is assigned to an MDM Server. No other function was defined for the MDM agent than the pre-defined ones in FMT_SMF_EXT.4.2.

## Guidance Assurance Activities

### Assurance Activity AA-FMT_SMF_EXT.4-MDMA-AGD-01

*The evaluator shall verify the AGD guidance includes detailed instructions for configuring each function in this requirement.*

*If the MDM Agent is a component of the MDM system (i.e. MDM Server is the Base-PP), the evaluator shall verify, by consulting documentation for the claimed mobile device platforms, that the configurable functions listed for this Agent are supported by the platforms.*

*If the MDM Agent supports multiple interfaces for configuration (for example, both remote configuration and local configuration), the AGD guidance makes clear whether some functions are restricted to certain interfaces.*

#### Summary

The evaluator noted that administrator-provided management functions in [PP_MDF_V3.2] , [MOD_BT_V1.0], [MOD_MDM_AGENT_V1.0] , [MOD_VPNC_V2.3], [PP_WLAN_CLI_EP_V1.0], , and [PKG_TLS_V1.1] are provided throughout [CCGUIDE] . A high-level summary is provided in Table 5 "Required Mobile Device Management Functions" of [CCGUIDE] where for each management function, a pointer to the relevant guidance is provided. Also, these management functions are already assessed in prior work units of this report, in particular, the work units of AA-FMT_SMF_EXT.1-AGD-01 .

Additionally, [CCGUIDE] provides Table 4 "SFR Configuration Requirements" outlining for each function (expressed in the form of an SFR and its description) whether configuration is needed and referencing the section where related guidance is provided, if any.

## Test Assurance Activities

### Assurance Activity AA-FMT_SMF_EXT.4-MDMA-ATE-01

- ***Test 1:*** *In conjunction with the evaluation activities in the Base-PP, the evaluator shall attempt to configure each administrator-provided management function and shall verify that the mobile device executes the commands and enforces the policies.*
- ***Test 2:*** *[TD0491] The evaluator shall configure the MDM Agent authentication certificate in accordance with the configuration guidance. The evaluator shall verify that the MDM Agent uses this certificate in performing the tests for FPT_ITT.1(2) (MDM as Base-PP) or FTP_ITC_EXT.1(2) (MDF as Base-PP).*
- ***Test 3:*** *In conjunction with other evaluation activities, the evaluator shall attempt to enroll the MDM Agent in management with each interface identified in the TSS, and verify that the MDM Agent can manage the device and communicate with the MDM Server.*
- ***Test 4:*** *[conditional] In conjunction with the evaluation activity for FAU_ALT_EXT.2.1, the evaluator shall configure the periodicity for reachability events for several configured time periods and shall verify that the MDM Server receives alerts on that schedule.*

- ***Test 5:*** *[conditional] The evaluator shall design and perform tests to demonstrate that the assigned function may be configured and that the intended behavior of the function is enacted by the mobile device.*

**Summary**

Test 1: The evaluator re-performed all the tests from FMT_SMF_EXT.1 for all the management function marked as 'Y' in the "Admin" column in Table 6 of [ST].

Test 2: For each device, the evaluator deployed the RemoteManagement device management configuration profile and verified the device started its Remote Management configuration.

Test 3: Manual enrollment: The evaluator accessed a specific preset URL containing an enrollment page and clicked to enroll the device. The evaluator accepted and installed the enrollment profile and verified the profile contained all settings and configurations applicable.

Test 3: Manual enrollment with Apple Configurator 2: The evaluator used the Profile Manager web UI to set allow enrollment and then wiped the device. When the first configuration screen appears on the device, the evaluator used Apple Configurator 2 to apply a profile. On the device he stepped through the Setup Assistant process. Upon completion, the evaluator verified the enrollment profile contains all appropriate settings and configurations applicable.

Test 3:Business Manager enrollment: The evaluator used Profile Manager to enable enforced Business Manager enrollment. The evaluator wiped the device, followed the new setup procedure, and verified that username and password are required and cannot be bypassed. After the setup process, the evaluator verified that the TOE is enrolled.

Tests 4 & 5 are not applicable.

## 2.2.5.9 User Unenrollment Prevention (FMT_UNR_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FMT_UNR_EXT.1-MDMA-ASE-01

*The evaluator shall ensure that the TSS describes the mechanism used to prevent users from unenrolling or the remediation actions applied when unenrolled.*

**Summary**

Section 8.6.4 *Unenrollment* in the TSS of [ST] describes the unenrollment process for the TOE. It states the to prevent unenrollment by users the key "PayloadRemovalDisallowed" must be specified in the Configuration Profile. 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 mobile device administrator to ensure that this key is set appropriately.

### Guidance Assurance Activities

#### Assurance Activity AA-FMT_UNR_EXT.1-MDMA-AGD-01

*The evaluator shall ensure that the administrative guidance instructs administrators in configuring the unenrollment prevention in each available configuration interface. If any configuration allows users to unenroll, the guidance also describes the actions that unenroll the Agent.*

**Summary**

[CCGUIDE] section 4.3.4 *Device Un-enrollment Prevention* provides guidance on how configure unenrollment prevention as follows:

- To allow or disallow a device user from removing the MDM Payload from the device, device administrator can set the PayloadRemovalDisallowed key in as described in [DEV_MAN] .
- The mobile device must Supervised Mode to lock the MDM Payload to the device.
- An MDM Payload can have a removal password associated with it. If the PayloadRemovalDisallowed key is set to prevent unenrollment and the MDM Payload has a removal password associated with it, the mobile device user can unenroll the mobile device only if the mobile device user knows the removal password.
- In the evaluated configuration, the TOE must be configured to disallow (prevent) unenrollment. If the administrator has allowed the ability for a mobile device user to unenroll the device, the user can remove the profile from the device by choosing Settings >> General >> Profile & Device Management, selecting the appropriate profile, and removing the profile.

### Test Assurance Activities

#### Assurance Activity AA-FMT_UNR_EXT.1-MDMA-ATE-01

- ***Test 1:*** *If 'prevent the unenrollment from occurring' is selected: The evaluator shall configure the Agent according to the administrative guidance for each available configuration interface, shall attempt to unenroll the device, and shall verify that the attempt fails.*
- ***Test 2:*** *If 'apply remediation actions' is selected: If any configuration allows the user to unenroll, the evaluator shall configure the Agent to allow user unenrollment, attempt to unenroll, and verify that the remediation actions are applied.*

#### Summary

Test 1: The evaluator used the Profile Manager to select "Prevent unenrollment" and deployed the change to the device. The evaluator then wiped the device and reinstalled as done for Business Manager, tried to remove the enrollment profile, and verified that the device cannot be unenrolled.

Test 2 is not applicable since the functionality is not selected in [ST].

## 2.2.6 Protection of the TSF (FPT)

### 2.2.6.1 Application Address Space Layout Randomization (FPT_AEX_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_AEX_EXT.1-ASE-01

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

#### Summary

Section 8.7.5 *Domain Isolation* in the TSS of [ST] states that the 8 bits of randonmess generated for ALSR (Address space layout randomization) are 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.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_AEX_EXT.1-ATE-01

> ***Evaluation Activity Note:*** *The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*
>
> - ***Test 1:*** *The evaluator must select 3 apps included with the TSF. These must include any web browser or mail client included with the TSF. For each of these apps, the evaluator shall launch the same app on two separate Mobile Devices of the same type and compare all memory mapping locations. The evaluator must ensure that no memory mappings are placed in the same location on both devices. If the rare (at most 1/256) chance occurs that two mappings are the same for a single app and not the same for the other two apps, the evaluator shall repeat the test with that app to verify that in the second test the mappings are different.*

#### Summary

The evaluator performed this test for three TOE OS applications, Safari, Mail, and Messenger. For each test, the evaluator did the following:

- start the app on two identical devices
- send a signal to each application to kill them and have them write a crash dump
- synchronize each device with iTunes
- retrieve the crash logs for the applications containing the memory addresses of the text segments and shared libraries

The evaluator verified the memory locations were different in each case.

## 2.2.6.2 Memory Page Permissions (FPT_AEX_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_AEX_EXT.2-ASE-01

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

#### Summary

Section 8.7.5 *Domain Isolation* in the TSS of [ST] describes memory management unit (MMU) which states that the MMU supports memory address translation using a translation table maintained by the TOE 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.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.6.3 Stack Overflow Protection (FPT_AEX_EXT.3)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_AEX_EXT.3-ASE-01

*The evaluator shall determine that the TSS contains a description of stack-based buffer overflow protections implemented in the TSF software which runs in the non-privileged execution mode of the application processor. The exact implementation of stack-based buffer overflow protection will vary by platform. Example implementations may be activated through compiler options such as "-fstack-protector-all", "-fstack-protector", and "/GS" flags. The evaluator shall ensure that the TSS contains an inventory of TSF binaries and libraries, indicating those that implement stack-based buffer overflow protections as well as those that do not. The TSS must provide a rationale for those binaries and libraries that are not protected in this manner.*

**Summary**

Section 8.7.5 *Domain Isolation* in the TSS of [ST] states that stack-based buffer overflow protection is implemented in every sandbox. The protection includes the following:

- Automatic reference counting (ARC): a memory management system that handles the reference count of objects automatically at compile time
- Address space layout randomization (ASLR)
- Stack-smashing protection: by utilizing a canary on the stack (Apple recommends that developers compile applications using the -fstack-protector-all compiler flag.)

Section 8.7.8 *Inventory of TSF Binaries and Libraries* refers to Annex D for the listing of TSF binaries and libraries. It also states that that all user space binaries (applications as well as shared libraries) are subject to address space layout randomization, thus indicating they are implemnted with stack-based buffer overflow protections.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.6.4 Domain Isolation (FPT_AEX_EXT.4)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_AEX_EXT.4-ASE-01

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

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

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

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

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

**Summary**

Section 8.7.3 *Secure Software Update* and section 8.7.5 *Domain Isolation* in the TSS of [ST] provide relevant description. These sections provide the following description:

- The entire TOE OS is mounted as read-only which prevents any application or attacker from modifying the system. 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 the TOE OS itself. Also, system updates are signed by Apple, along with applications which are signed by their developers with their Apple developer certificate, which prevents any non-TSF software from modifying the TSF software or TSF data.
- All applications are executed in their own domain or 'sandbox' which isolates the application from other applications and the rest of the system.
- Application are restricted from accessing files stored by other applications or from making changes to the TOE 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 the TOE OS.
- When a TOE device with cellular capabilities (i.e., a dialer) is in the lock state, only the Emergency Call dialer is available to make a call. The Emergency Call interface does not support the entering of Unstructured Supplementary Service Data (USSD or Man-Machine Interface (MMI) codes. On TOE devices, USSD/MMI codes start with one or more of the following characters: number sign (#), asterisk (*). The Emergency Call interface ignores (i.e., does not send when the Call button is pressed) codes that start with one or more of these characters; thereby, preventing the code actions from executing.
- The TOE also does not support auxiliary boot modes; therefore, the ability to enter codes using auxiliary boot modes does not exist.
- Every step of the boot 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. This secure boot chain helps ensure that the lowest levels of software aren't tampered with.
- When a TOE device is turned on, its application processor immediately executes code from read-only memory known as 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 iBoot bootloader is signed by Apple before allowing it to load. Because the Apple Root CA public key resides in immutable code, no software (including unverified or unauthorized software) can modify this public key. This is the first step in the chain of trust where each step ensures that the

next is signed by Apple. When the iBoot finishes its tasks, it verifies and runs the TOE OS kernel. For devices with an A9 or earlier A-series processor, an additional Low-Level Bootloader (LLB) stage is loaded and verified by the Boot ROM and in turn loads and verifies iBoot.

- A failure of the Boot ROM to load LLB (on older devices) or iBoot (on newer devices) results in the device entering DFU mode. In the case of a failure in LLB or iBoot to load or verify the next step, startup is halted and the device displays the connect to iTunes screen. This is known as recovery mode. In either case, the device must be connected to iTunes through USB and restored to factory default settings.
- The Boot Progress Register (BPR) is used by the SEP to limit access to user data in different modes and is updated before entering the following modes:
    - Recovery Mode: Set by iBoot on devices with Apple A10 system on a chip (SoCs) and higher
    - DFU Mode: Set by Boot ROM on devices with Apple A12 SoC and higher

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_AEX_EXT.4-ATE-01

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

- ***Test 1:*** *The evaluator shall create and load an app onto the Mobile Device. This app shall attempt to traverse over all file systems and report any locations to which data can be written or overwritten. The evaluator must ensure that none of these locations are part of the OS software, device drivers, system and security configuration files, key material, or another untrusted application's image/data. For example, it is acceptable for a trusted photo editor app to have access to the data created by the camera app, but a calculator application shall not have access to the pictures.*
- ***Test 2:*** *For each available auxiliary boot mode, the evaluator shall attempt to modify a TSF file of their choosing using the software and/or hardware tools described in the TSS. The evaluator shall verify that the modification fails.*

#### Summary

Test 1: The evaluator compiled and loaded the application Find-UI to the device, executed Find-UI, and verified that all locations marked as writable are not part of the OS (i.e. is only located within the applications home directory).

Test 2: The evaluator booted into iTunes/DFU mode from power off state and verified that no service is available to modify files.

## 2.2.6.5 JTAG Disablement (FPT_JTA_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_JTA_EXT.1-ASE-01

*If "disable access through hardware" is selected:*

*The evaluator shall examine the TSS to determine the location of the JTAG ports on the TSF, to include the order of the ports (i.e. Data In, Data Out, Clock, etc.).*

*If "control access by a signing key" is selected:*

*The evaluator shall examine the TSS to determine how access to the JTAG is controlled by a signing key. The evaluator shall examine the TSS to determine when the JTAG can be accessed, i.e. what has the access to the signing key.*

**Summary**

Per FPT_JTA_EXT.1.1, the TSF must disable access through hardware to JTAG.

Section 8.7.2 *Joint Test Action Group (JTAG) Disablement* in the TSS of [ST] describes how JTAG can be disabled. It states the following:

- To use this JTAG-like interface, a development-fused device is required. In a development-fused device, certain hardware fuses in the device are not blown during the manufacturing process that are blown in a production-fused device. 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 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 special 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.

## Guidance Assurance Activities

No assurance activities defined.

## Test Assurance Activities

### Assurance Activity AA-FPT_JTA_EXT.1-ATE-01

***Evaluation Activity Note:*** *The following test requires the developer to provide access to a test platform that provides the evaluator with chip level access.*

*If "disable access through hardware" is selected:*

*The evaluator shall connect a packet analyzer to the JTAG ports. The evaluator shall query the JTAG port for its device ID and confirm that the device ID cannot be retrieved.*

**Summary**

iPhones use a 2-stage interface that resembles the functionality of JTAG but does not implement the JTAG protocol.

The Apple development environment provides a JTAG-like interface (technically it is a serial link and no JTAG). 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 is the JTAG-like interface 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. 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.

The evaluator performed the following verification steps to ensure that the JTAG-like interface is not reachable:

- The evaluator used the special developer's Lightning cable to connect the regular production test devices and verified that the JTAG-like interface was not accessible.
- The evaluator used the same special Lightning cable to connect the development-fused devices and verified that the JTAG-like connection was accessible.

## 2.2.6.6 Key Storage (FPT_KST_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_KST_EXT.1-ASE-01

*The evaluator shall consult the TSS section of the ST in performing the Evaluation Activities for this requirement.*

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

*The evaluator shall ensure that the description also covers how the cryptographic functions in the FCS requirements are being used to perform the encryption functions, including how the KEKs, DEKs, and stored keys are unwrapped, saved, and used by the TOE so as to prevent plaintext from being written to non-volatile storage. The evaluator shall ensure that the TSS describes, for each power-down scenario how the TOE ensures that all keys in non-volatile storage are not stored in plaintext.*

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

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

*For each BAF selected in FIA_UAU.5.1:*

*The evaluator shall determine that the TSS also contains a description of the activities that happen on biometric authentication, relating to the decryption of DEKs, stored keys, and data. In addition how the system ensures that the biometric keying material is not stored unencrypted in persistent storage.*

#### Summary

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes key management.

The evaluator verified that section 8.3.1 describes how the REK, KEKs and DEKs interact by hierarchically wrapping each other, and how these keys are generated or formed upon boot (or from the authentication of the user to the TOE OS by providing the correct passcode). This section also describes which keys are wrapped, by which algorithm (namely AES in Key Wrap cipher mode), and which keys are stored in plaintext, encrypted, and in which type of memory (flash, non-volatile, etc.). The evaluator verified that the only key material that is written unencrypted in TOE OS is the REK (which is not accessible to any part of the system except the SEP), and the 0x89B and 0x835 keys. The latter are, however, stored in block 0 of the flash memory and can be erased very quickly if necessary. These keys are created during boot time and destroyed when the device is powered-off. When the device is locked, the passcode key is erased and can only be regenerated when the user provides the correct password.

Section 8.5.1 *Biometric Authentication* in the TSS of [ST] describes the events during biometric authentication. According to section 8.5, authentication credentials including the password and biometric samples are not stored on the TOE device. The key derivation, combined with the salt and UID, derive the passcode key used to unencrypt keys used to encrypt files on the system.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.6.7 No Key Transmission (FPT_KST_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_KST_EXT.2-ASE-01

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

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

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

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

*For each BAF selected in FIA_UAU.5.1:*

*In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on biometric authentication, including how any plaintext material, including critical security parameters and results of biometric algorithms, are protected and accessed.*

*The evaluator shall ensure that the TSS describes how functions available in the biometric algorithms ensure that no unencrypted plaintext material, including critical security parameters and intermediate results, is transmitted outside the security boundary of the TOE or to other functions or systems that transmit information outside the security boundary of the TOE.*

#### Summary

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes key management. The TOE is the mobile device running the TOE OS which is iOS, depending on the TOE device. In particular section 8.3.1.2 *No plaintext key transmission and export* states the following:

- The TOE security boundary is the entire mobile device
- The TOE does not transmit or export plaintext key material outside of the TOE security boundary.
- Plaintext key material is never logged.
- Biometric credential data is confined in the SEP.
- Biometric keying material, enrollment and authentication templates, the features an algorithm uses to perform biometric authentication for enrollment or verification, threshold values, intermediate calculations, and final match scores never leave the SEP.
- Plaintext keys such as plaintext DEKs, KEKs, and keys stored in the secure key storage are never exported.

In addition, evaluator found in section 8.3.1 a description of the power-up process and password authentication within the TOE OS relating to the generation, derivation, and decryption of DEKs, DEKs themselves, and stored data. Whenever a file is accessed, it gets decrypted by its respective class key. The file is then decrypted with its respective file key. Whenever a file is closed, its file key is stored encrypted in memory. At any time, no key is sent outside the security boundary of the TOE (i.e., outside the device).

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.6.8 No Plaintext Key Export (FPT_KST_EXT.3)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_KST_EXT.3-ASE-01

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

#### Summary

Section 8.2.1 *The Secure Enclave Processor (SEP)* in the TSS of [ST] explicitly states that the main system has no access to other memory areas of the SEP and no keys or key material may be exported.

This section also states that the SEP execution environment shares the main random access memory (RAM) with the application processor. The memory controller provides memory separation between the two processors by distinguishing between the origin of memory fetch or store requests performed by the processors. In addition, the memory controller encrypts/obfuscates (using AES-XEX) all SEP data in RAM using a key shared only between the SEP and the memory controller. This adds an additional level of protection between the SEP memory and the application processor. If the memory separation between the two processors were violated to where the application processor could access the SEP RAM, the application processor would only see encrypted data.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.6.9 Self-Test Notification (FPT_NOT_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_NOT_EXT.1-ASE-01

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

**Summary**

Section 8.7.1 *Secure Boot* and section 8.7.9 *Self-Tests* provides relevant information.

According to these sections, if the TOE encounters a failure of the self-tests the TOE will power itself off and will require a reboot. In case of a failure of the integrity check, the TOE will display the connect to iTunes screen to transition in the recovery mode. In this mode, the TOE device would require a USB connection to access iTunes in order to perform a factory reset.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_NOT_EXT.1-ATE-01

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

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

**Summary**

Test 1: The tester booted the TOE correctly and then booted the TOE after modifying the integrity of the kernel space and user space binaries. The tester was then able to use specialized Apple equipment to perform this test and obtain the boot logs that show that the TOE was unable to boot when the binaries are modified, due to integrity test failures.

## 2.2.6.10 Reliable Time Stamps (FPT_STM.1)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_STM.1-ASE-01

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

**Summary**

Section 8.7.7 *Time* in the TSS of [ST] describes the time functionality and its usage. It states the following:

- The following SFRs make use of time:
  - FAU_GEN.1.2 (Audit record time stamp)
  - FIA_TRT_EXT.1 (Authentication throttling)
  - FIA_UAU.7 (Protected authentication feedback)
  - FIA_X509_EXT.1.1 (TOE certificate expiration validation)

- FIA_X509_EXT.3.1 (Application certificate expiration validation)
- FMT_SMF_EXT.1.1 Function 1 (Password lifetime)
- FMT_SMF_EXT.1.1 Function 2 (Screen-lock timeout management)
- FTA_SSL_EXT.1 (Lock state inactivity timeout)

This section also states that the TOE sets time using either GPS, Apple NTP server, NITZ or the cellular carrier time service. NTP is configured by the administrator.

### Guidance Assurance Activities

#### Assurance Activity AA-FPT_STM.1-AGD-01

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

#### Summary

[CCGUIDE] section 5.6.4 *Timestamp Configuration* provides guidance for setting up the date and time. The TOE can use several time sources to automatically update the time: Network, Identity and Time Zone (NITZ), Global Positioning Satellites (GPS), Network Time Protocol (NTP) standards, or the cellular carrier time service. When configured and maintained using one of these time sources, the time may be considered reliable. Only the NTP is configurable by the mobile device administrator. To do this, the administrator uses a Configuration Profile with the Time Server Payload using the *timeServer* and *timeZone* keys.

### Test Assurance Activities

#### Assurance Activity AA-FPT_STM.1-ATE-01

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

#### Summary

Test 1: Setting of time is only possible when NTP is disabled. The evaluator disabled NTP, changed the time, and verified that the new time is displayed at the top of the screen.

## 2.2.6.11 TSF Cryptographic Functionality Testing (FPT_TST_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TST_EXT.1-ASE-01

*The evaluator shall examine the TSS to ensure that it specifies the self-tests that are performed at start-up. This description must include an outline of the test procedures conducted by the TSF (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The TSS must include any error states that they TSF may enter when self-tests fail, and the conditions and actions necessary to exit the error states and resume normal operation. The evaluator shall verify that the TSS indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user or operator.*

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

#### Summary

Section 8.7.9 *Self-Tests* in the TSS of [ST] describes self-tests.

It states the following:

- Self-tests are performed by the three cryptographic modules of the TOE: Apple corecrypto Module [Apple ARM, User, Software, SL1], Apple corecrypto Module [Apple ARM, Kernel, Software, SL1], and Apple corecrypto Module [Apple ARM, Secure Key Store, Hardware, SL2]
- Self-tests are run by the TOE OS when the TOE device is powered up.
- The TOE performs the FIPS 140-3 power-on self-tests for its cryptographic algorithms, along with a software integrity test using HMAC-SHA-256.
- If any of the defined self-tests fails, the TOE device powers itself off automatically.

More information about the self-tests including the test procedures can be found in sections 8.7.9.1 through 8.7.9.3 of [ST] where a section is dedicated to each of the three cryptographic modules. The evaluator examined these sections and verified that the description of self-tests includes algorithm self-tests. Each section provides a table listing the algorithms tests performed by that cryptographic module. Known answer tests include AES, DRBG, and HMAC-SHA algorithms. alogirthm

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

No assurance activities defined.

## 2.2.6.12 TSF Self-Test (FPT_TST_EXT.1/VPN)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TST_EXT.1-VPN-ASE-01

*The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF on startup; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. If some of the tests are performed by the TOE platform, the evaluator shall check the TSS to ensure that those tests are identified, and that the ST for each platform contains a description of those tests. Note that the tests that are required by this component are those that support security functionality in the VPN Client PP-Module, which may not correspond to the set of all self-tests contained in the platform STs.*

*The evaluator shall examine the TSS to ensure that it describes how the integrity of stored TSF executable code is cryptographically verified when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator shall check to ensure that the cryptographic requirements listed are consistent with the description of the integrity verification process.*

*The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. For checks implemented entirely by the platform, the evaluator ensures that the operational guidance for the TOE references or includes the platform-specific guidance for each platform listed in the ST.*

#### Summary

Description of self-tests is provided in the TSS of [ST] in section 8.7.9 *Self-Tests* . According to this section, self-tests are performed by the three cryptographic modules of the TOE as outlined below. The TOE is an entire mobile device hence all self-tests are performed entirely by the TOE itself and not the TOE platform.

Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1]

This cryptographic 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 which is invoked by the the TOE OS startup process upon device power on, runs all required module self-tests.

Self-tests performed by this module includes:

- Power-up tests: these tests (as listed in table 13 "Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1] Cryptographic Algorithm Tests" of [ST] ) are performed each time the module starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the required tests fail, the device powers itself off. To rerun the self-tests on demand, the user must reboot the device. If the followings tests succeed, the device continues with its normal power-up process.
- Software / Firmware integrity tests: A software integrity test is performed on the runtime image of the Apple TOE OS corecrypto Cryptographic Module for Apple 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. If the followings tests succeed, the device continues with its normal power-up process.
- Conditional tests covering pair-wise consistency tests and SP 800-90A assurance tests.

Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1]

This cryptographic 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, which is invoked by the TOE OS Kernel startup process upon device initialization, runs all required module self-tests. If the self-tests succeed, the module's instance is maintained in the memory of the TOE OS Kernel on the device and made available to each calling kernel service without reloading.

Self-tests performed by this module includes:

- Power-up tests: these tests are performed each time the module 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. If the tests succeed, the device continues with its normal power-up process.
- Cryptographic algorithm tests specified in table 14 "Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1] Cryptographic Algorithm Tests" of [ST] .
- Software / Firmware integrity test: this test is performed on the runtime image of the module. 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. If the test succeeds, the device continues with its normal power-up process.
- Conditional tests covering continuous RNG test, pair-wise consistency test, and SP800-90 assurance test

Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2]

This module runs all required module self-tests pertaining to the firmware. These self-tests are invoked automatically when starting the module. Also during startup of the hardware, the hardware DRBG invokes its independent self-test. If the self-test fails in either the firmware or hardware DRBG the device immediately shuts down to prevent any operation.

Self-tests performed by this module includes:

- Power-up tests: these tests are performed each time the TOE devices start and must be completed successfully for the module to operate in FIPS approved mode. If any of the following tests fails the device powers itself off. To run the self-tests on demand, the user must reboot the device. If the tests succeed, the device continues with its normal power-up process.
- Cryptographic algorithm tests specified in table 15 "Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2] Cryptographic Algorithm Tests" of [ST] .
- Software / Firmware integrity tests: this test is performed on the runtime image of the TOE devices. The module's HMAC-SHA-256 is used as an FIPS approved algorithm for the integrity test. If the test fails, then the device powers itself off. If the test succeeds, the device continues with its normal power-up process.
- Conditional tests covering pair-wise consistency test and SP800-90 assurance test

In addition, section 8.7.9 of the TSS provides the following rationale:

> *These tests are sufficient to demonstrate that the TSF is operating correctly since they include each of the cryptographic modules included in the TSF. If the self-tests fail then the TSF will not operate. In addition, the secure boot process begins in hardware and builds a chain of trust through software using the self-tested corecrypto modules, where each step ensures that the next is properly vetted before handing over control to that TSF executable. Secure boot of the devices ensures that the lowest levels of software are not tampered with and that only trusted operating system software from Apple loads at startup. In the devices, security begins in immutable code called the Boot ROM, which is laid down during chip fabrication and known as the hardware root of trust and continues through the loading of the TSF executables.*

The evaluator found the rationale sound and reasonable.

Furthermore, the evaluator verified that the cryptographic requirements listed are consistent with the description of the integrity verification process.

As seen above the TSS describes how each cryptographic module performs software/firmware integrity tests. For each cryptographic module if the software/firmware integrity test fails, then the TOE device powers off. If the test succeeds, the device continues with its normal power-up process.

## Guidance Assurance Activities

### Assurance Activity AA-FPT_TST_EXT.1-VPN-AGD-01

*If not present in the TSS, the evaluator ensures that the operational guidance describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. For checks implemented entirely by the platform, the evaluator ensures that the operational guidance for the TOE references or includes the platform-specific guidance for each platform listed in the ST.*

### Summary

Description of self-tests is provided in the TSS of [ST] in section 8.7.9 *Self-Tests* . According to this section, self-tests are performed by the three cryptographic modules of the TOE as outlined below. The TOE is an entire mobile device hence all self-tests are performed entirely by the TOE itself and not the TOE platform and thus no platform-specific guidance is required.

Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1]

This 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 TOE OS startup process upon device power on.

Self-tests performed by this module includes:

- Power-up tests: these tests (as listed in Table 13 of [ST] ) are performed each time the module starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the tests fails the device shuts down automatically. To rerun the self-tests on demand, the user must reboot the device. If the tests succeed, the device continues with its normal power-up process.
- Software / Firmware integrity tests: A software integrity test is performed on the runtime image of the crypto module. 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. If the test succeeds, the device continues with its normal power-up process.
- Conditional tests covering pair-wise consistency tests and SP 800-90A assurance tests.

Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1]

This 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 TOE OS Kernel startup process upon device initialization. If the self-tests succeed, the Apple corecrypto Kernel Cryptographic Module for Apple ARM instance is maintained in the memory of the TOE OS Kernel on the device and made available to each calling kernel service without reloading.

Self-tests performed by this module includes:

- Power-up tests: these tests are performed each time the module starts and must be completed successfully for the module to operate in the FIPS approved mode. If any of the tests fails the device shuts down automatically. To run the self-tests on demand, the user may reboot the device. If the tests succeed, the device continues with its normal power-up process.
- Cryptographic algorithm tests specified in Table 14 of [ST] .
- Software / Firmware integrity test: this test is performed on the runtime image of the module. 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. If the test succeeds, the device continues with its normal power-up process
- Conditional tests covering continuous RNG test, pair-wise consistency test, and SP800-90 assurance test

Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2]

This module runs all required module self-tests pertaining to the firmware. These self-tests are invoked automatically when starting the module. Also during startup of the hardware, the hardware DRBG invokes its independent self-test. If the self-test fails in either the firmware or hardware DRBG the device immediately shuts down to prevent any operation.

Self-tests performed by this module includes:

- Power-up tests: these tests are performed each time the TOE devices start and must be completed successfully for the module to operate in FIPS approved mode. If any of the tests fails the device shuts down automatically. To run the self-tests on demand, the user must reboot the device. If the tests succeed, the device continues with its normal power-up process.
- Cryptographic algorithm tests specified in Table 15 of [ST] .
- Software / Firmware integrity tests: this test is performed on the runtime image of the TOE OS. The module's HMAC-SHA-2565 is used as an FIPS approved algorithm for the integrity test. If the test fails, then the device powers itself off. If the test succeeds, the device continues with its normal power-up process.
- Conditional tests covering pair-wise consistency test and SP800-90 assurance test.

As stated above, the TSS describes how each cryptographic module peforms software/firewmare integrity tests. For each cryptographic module if the software/firmware integrity test fails, then the TOE device powers off. If the test succeeds, the device continues with its normal power-up process.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TST_EXT.1-VPN-ATE-01

*The evaluator shall perform the following tests:*

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

#### Summary

The evaluator booted the TOE correctly (Test 1) and then booted the TOE after modifying the integrity of the kernel space and user space binaries (Test 2). The evaluator was then able to use specialized Apple equipment to perform this test and obtain the boot logs that show that the TOE was unable to boot when the binaries are modified, due to integrity test failures.

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

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TST_EXT.1-WLAN-ASE-01

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

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

#### Summary

Section 8.7.9 *Self-tests* in the TSS of [ST] describes the self-tests performed by the TOE. Section 8.7.1 *Secure Boot* describes the secure boot process.

Section 8.7.9 splits into several sections describing the self-test implemented by the following three cryptographic modules of the TOE. Self-tests include the power-on tests, cryptographic algorithm tests (known-answer tests or pair-wise consistency test), software/firmware integrity tests, critical function tests, and conditional tests.

1. Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1]
2. Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1]
3. Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2]

These sections clearly explain how the integrity of the libraries (libcorecrypto.dylib, libSystem.dylib and libcommoncrypto.dylib and the runtime image of the TOE OS kernel) is checked using the HMAC-SHA-256 algorithm. These sections also describe that if any of these tests should fail, the device shuts down automatically. Also, section 8.7.1 explains that if one step of the boot process, including the integrity verification of the stored TSF executable, fails, then the system does not boot and displays the "Connect to iTunes" screen, for TOE device recovery.

The evaluator found that section 8.7.9.1 provides an argument that the tests are sufficient:

> *The TOE OS 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 the TOE OS.*

## Guidance Assurance Activities

### Assurance Activity AA-FPT_TST_EXT.1-WLAN-AGD-01

> *The evaluator shall ensure that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases.*

#### Summary

Section 8.7.9 *Self-Tests* in the [ST] describes the self-tests performed by the TOE. Section 8.7.1 *Secure Boot* describes the secure boot process.

Section 8.7.9 is split into several sections described the different self-tests implemented by the TOE. This includes power-up test, cryptographic algorithms tests (know-answer-tests or pair-wise consistency test), software/firmware integrity tests, and critical function tests. These self-tests are performed for the TOE's Apple CoreCrypto Cryptographic Module for ARM, Apple CoreCrypto Kernel Cryptographic Module for ARM, and Apple Secure Key Store Cryptographic Module. These sections clearly explain how the integrity of the libraries (libcorecrypto.dylib, libSystem.dylib and libcommoncrypto.dylib and the runtime image of the iOS/iPadOS kernel) is checked using the HMAC-SHA-256 algorithm. These sections also describe that if any of these tests should fail, the device shuts down automatically.

Section 8.7.1 explains that if one step of the boot process, including the integrity verification of the stored TSF executable, fails, then the system does not boot and displays the "Connect to iTunes" screen, for device recovery.

## Test Assurance Activities

### Assurance Activity AA-FPT_TST_EXT.1-WLAN-ATE-01

*The evaluator shall perform the following tests:*

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

**Summary**

For both tests, the evaluator booted the TOE correctly and then booted the TOE after modifying the integrity of the kernel space and user space binaries. The evaluator was then able to use specialized Apple equipment to perform this test and obtain the boot logs that show that the TOE was unable to boot when the binaries are modified, due to integrity test failures.

## 2.2.6.14 TSF Integrity Checking (Post-Kernel) (FPT_TST_EXT.2/POSTKERNEL)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TST_EXT.2-POSTKERNEL-ATE-01

*The evaluation activity shall be completed in conjunction with FPT_TST_EXT.2/PREKERNEL for all executable code specified.*

**Summary**

The evaluation activity is performed in conjunction with FPT_TST_EXT.2/PREKERNEL.

## 2.2.6.15 TSF Integrity Checking (Pre-Kernel) (FPT_TST_EXT.2/PREKERNEL)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TST_EXT.2-PREKERNEL-ASE-01

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

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

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

**Summary**

Section 8.7.1 *Secure Boot* in the TSS of [ST] describes the boot process of the TOE. It states the following:

- Every 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 a TOE device is turned on, its application processor immediately executes code from read-only memory known as 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 iBoot bootloader is signed by Apple before allowing it to load. Because the Apple Root CA public key resides in immutable code, no software (including unverified or unauthorized software) can modify this public key. This is the first step in the chain of trust where each step ensures that the next is signed by Apple. When the iBoot finishes its tasks, it verifies and runs the TOE OS kernel. For devices with an A9 or earlier A-series processor, an additional Low-Level Bootloader (LLB) stage is loaded and verified by the Boot ROM and in turn loads and verifies iBoot.
- A failure of the Boot ROM to load LLB (on older devices) or iBoot (on newer devices) results in the device entering DFU mode. In the case of a failure in LLB or iBoot to load or verify the next step, startup is halted and the device displays the connect to iTunes screen. This is known as recovery mode. In either case, the device must be connected to iTunes through USB and restored to factory default settings.
- The Boot Progress Register (BPR) is used by the SEP to limit access to user data in different modes and is updated before entering the following modes:
    - Recovery Mode: Set by iBoot on devices with Apple A10 system on a chip (SoCs) and higher as well as Apple M1 SoC
    - DFU Mode: Set by Boot ROM on devices with Apple A12 SoC and higher as well as Apple M1 SoC

Section 8.7.1 also states the following:

- 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. This secure boot chain helps ensure that the lowest levels of software aren't tampered with.

According to section 8.7.5 *Domain Isolation* in the TSS of [ST] the TOE does not support auxiliary boot modes.

## Guidance Assurance Activities

No assurance activities defined.

## Test Assurance Activities

### Assurance Activity AA-FPT_TST_EXT.2-PREKERNEL-ATE-01

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

*The evaluator shall perform the following tests:*

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

- ***Test 2:*** *The evaluator shall modify a TSF executable that is integrity protected and cause that executable to be successfully loaded by the TSF. The evaluator observes that an integrity violation is triggered and the TOE does not boot. (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that the module was modified so that it was rendered unable to run because its format was corrupt).*
- ***Test 3:*** *[conditional] If the ST author indicates that the integrity verification is performed using a public key, the evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1. The evaluator shall digitally sign the TSF executable with a certificate that does not have the Code Signing purpose in the extendedKeyUsage field and verify that an integrity violation is triggered. The evaluator shall repeat the test using a certificate that contains the Code Signing purpose and verify that the integrity verification succeeds. Ideally, the two certificates should be identical except for the extendedKeyUsage field.*

**Summary**

Test 1: The evaluator rebooted the TOE and verified that reboot was successful.

Test 2: Using specialized Apple equipment, the evaluator modified the integrity values that are verified during the boot process and observed the boot process halted when the TOE was rebooted.

Test 3 is not applicable.

## 2.2.6.16 TSF Integrity Testing (FPT_TST_EXT.3)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TST_EXT.3-ATE-01

*Testing for this element is performed in conjunction with the Evaluation Activities for FPT_TST_EXT.2.1/PREKERNEL.*

**Summary**

The evaluation activity is performed in conjunction with FPT_TST_EXT.2/PREKERNEL.

## 2.2.6.17 Trusted Update: TSF Version Query (FPT_TUD_EXT.1)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.1-ATE-01

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

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

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

**Summary**

The evaluator could query the version of TOE OS, the hardware model of the device, and the version of all installed apps using

- `Settings -> General -> About -> Software Version`
- `Settings -> General -> About -> Model Name`
- `Settings -> General -> iPhone Storage` (and then selecting a specific app)

## 2.2.6.18 TSF Update Verification (FPT_TUD_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.2-ASE-01

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

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

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

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

**Summary**

Section 8.7.3 *Secure Software Update* in the TSS of [ST] describes the secure update progress. When updates are available, the TOE users Finder application on macOS versions 10.15.0 (Catalina) and higher, through iTunes on macOS versions prior to 10.15.0 and on a PC. All updates are digitally signed with the Apple code signing certificate and users cannot downgrade to older TOE OS versions (prevented by the System Software Authorization process).

The TOE OS software updates can be installed using the Finder application on macOS version 10.15.0 and higher, using iTunes on macOS versions prior to 10.15.0 and on PCs, or over-the-air (OTA) on the device via HTTPS trusted channel. With iTunes and Finder, a full copy of the latest OS 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 macOS Server so that the TOE devices do not need to access Apple servers to obtain the necessary update data. Software updates may also be cached on macOS version 10.13.0 and higher running the built-in caching service (in the client software).

During a TOE OS upgrade, Finder/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 Exclusive Chip Identification (ECID).

This section also states that software updates are digitally signed and verified using a hardware-protected asymmetric key used for signature verification. The startup process described in section 8.7.1 *Secure Boot* helps ensure that only Apple-signed code can be installed on a TOE device. The Apple Root CA public key, which resides in immutable code as described in section 8.7.1, is used to verify the signed updates.

The ST author does not indicate in the TSS that software updates to system software running on other processors is verified. The evaluator found this to be consistent with FPT_TUD_EXT.2.1 which selects "no other processor system software".

Section 8.5.2 *X.509v3 Certificates* describes X.509 certificate validation including certificates using for code signing in according to FIA_X509_EXT.1 including validation of extendedKeyUsage(EKU) against rules defined in FIA_X509_EXT.1.

## Guidance Assurance Activities

No assurance activities defined.

## Test Assurance Activities

### Assurance Activity AA-FPT_TUD_EXT.2-ATE-01

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

- ***Test 1:*** *The tester shall try to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.*
- ***Test 2:*** *The tester shall digitally sign the update with a key disallowed by the device and verify that installation fails. The tester shall attempt to install an update signed with the allowed key and verify that installation succeeds.*
- ***Test 3:*** *[conditional] The tester shall digitally sign the update with an invalid certificate and verify that update installation fails. The tester attempt to install an update that was digitally signed using a valid certificate and a certificate that contains the purpose and verify that the update installation succeeds.*
- ***Test 4:*** *[conditional] The tester shall repeat these test for the software executing on each processor listed in the first selection. The tester shall attempt to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.*

#### Summary

Test 1: The evaluator fetched a valid but unsigned iBEC image (2nd boot loader that is executed after the ROM boot loader is started), tried to load the image onto the device and observed that the image is rejected by the boot loader. The positive test is to verify that properly signed images are booted is simply to reboot the device with an official TOE OS image and verify that the boot is successful.

Test 2: The evaluator signed the boot loader binary with a development RSA key, verified that the newly generated key is listed in the manifest file accompanying the iBEC image, tried to load the image onto the device and observed that the image is rejected by the boot loader. Then the evaluator used an iBEC image that was signed with Apple key, loaded it onto the device, and verified that the image load succeeded.

Test 3: The test specified in the PP is not applicable. Apple uses a CA that is dedicated for software signing. The CA and its keys is not used for any other purpose. The root certificate is hard coded into the ROM code (validated by the evaluator's code inspection). The certificate path validation verifies that the software certificate used to sign an image traces back to the root certificate. Therefore, the verification of the certificate extension is not required as all certificates that are deemed valid based on the certificate chain validation are only used for software signatures as defined by Apple policy.

Test 4 is not applicable.

## 2.2.6.19 Application Signing (FPT_TUD_EXT.3)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.3-ASE-01

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

#### Summary

Section 8.7.10 *Application integrity* in the TSS of [ST] states the following:

Apple issues certificates to TOE OS application developers that developers use to sign their applications. The TOE OS checks each application’s signature to ensure that it was signed using a valid Apple-issued certificate by using a hardware-protected asymmetric key. This applies to both application installation and application execution.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.3-ATE-01

***Evaluation Activity Note:*** *The following test does not have to be tested using the commercial application store.*

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

#### Summary

Test 1 is performed in conjunction with the assurance activities for FPT_TUD_EXT.5 as stated in [PP_MDF_V3.2].

## 2.2.6.20 Trusted Update Verification (FPT_TUD_EXT.4)

### TSS Assurance Activities

No assurance activities defined.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.4-ATE-01

*Testing for this element is performed in conjunction with the Evaluation Activities for FPT_TUD_EXT.2 and FPT_TUD_EXT.5.*

#### Summary

This testing is performed in conjunction with the assurance activities for FPT_TUD_EXT.2 and FPT_TUD_EXT.5 as stated in [PP_MDF_V3.2].

## 2.2.6.21 Application Verification (FPT_TUD_EXT.5)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.5-ASE-01

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

#### Summary

Section 8.7.10 *Application integrity* in the TSS of [ST] states the following:

Apple issues certificates to TOE OS application developers that developers use to sign their applications. The TOE OS checks each application's signature to ensure that it was signed using a valid Apple-issued certificate by using a hardware-protected asymmetric key. This applies to both application installation and application execution.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.5-ATE-01

- ***Test 1:*** *The evaluator shall write, or the developer shall provide access to, an application. The evaluator shall try to install this application without a digitally signature and shall verify that installation fails. The evaluator shall attempt to install an application digitally signed with an appropriate certificate, and verify that installation succeeds.*
- ***Test 2:*** *The evaluator shall digitally sign the application with an invalid certificate and verify that application installation fails. The evaluator shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. This test may be performed in conjunction with the Evaluation Activities for FIA_X509_EXT.1.*

- ***Test 3:*** *If necessary, the evaluator shall configure the device to limit the public keys that can sign application software according to the AGD guidance. The evaluator shall digitally sign the application with a certificate disallowed by the device or configuration and verify that application installation fails. The evaluator shall attempt to install an application digitally signed with an authorized certificate and verify that application installation succeeds.*

**Summary**

Tests 1 & 2: Tooling does not allow creation of unsigned applications. The evaluator tested installation of a validly signed app by generating an IPA package, verifying the correct signature, and generating a new certificate using CodeSigning purpose using KeyChain. The evaluator located the binary and invoked a command to sign the application. The evaluator then zipped it back together and successfully pushed the application to the TOE using Xcode. The evaluator removed the application from the TOE and pushed the modified app on the TOE and verified that the app is not usable.

Test 3 is not applicable as the signing key cannot be managed by any user.

## 2.2.6.22 Trusted Update Verification (FPT_TUD_EXT.6)

### TSS Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.6-ASE-01

*The evaluator shall verify that the TSS describes the mechanism that prevents the TSF from installing software updates that are an older version that the currently installed version.*

**Summary**

Section 8.7.3 *Secure Software Update* in the TSS of [ST] describes the secure update process. It states that to prevent TOE devices from downgrading to older TOE OS versions, the TOE OS uses the System Software Authorization process that is described in detailed in this section. Also, the SEP utilizes System Software Authorization to ensure the integrity of its software and prevent downgrade installations.

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FPT_TUD_EXT.6-ATE-01

*The evaluator shall repeat the following tests to cover all allowed software update mechanisms as described in the TSS. For example, if the update mechanism replaces an entire partition containing many separate code files, the evaluator does not need to repeat the test for each individual file.*

- ***Test 1:*** *The evaluator shall attempt to install an earlier version of software (as determined by the manufacturer). The evaluator shall verify that this attempt fails by checking the version identifiers or cryptographic hashes of the privileged software against those previously recorded and checking that the values have not changed.*
- ***Test 2:*** *The evaluator shall attempt to install a current or later version and shall verify that the update succeeds.*

**Summary**

Test 1: Support from Apple is required to push an older version of the OS to the device. The evaluator used Apple tools to push an older version of the OS to the device and verified that it was rejected by the device.

Test 2: The evaluator performed a standard software update on the TOE and verified the new version was downloaded.

## 2.2.7 TOE access (FTA)

### 2.2.7.1 TSF- and User-initiated Locked State (FTA_SSL_EXT.1)

#### TSS Assurance Activities

##### Assurance Activity AA-FTA_SSL_EXT.1-ASE-01

*The evaluator shall verify the TSS describes the actions performed upon transitioning to the locked state.*

*[The evaluation shall verify that the AGD guidance describes the method of setting the inactivity interval and of commanding a lock.] The evaluator shall verify that the TSS describes the information allowed to be displayed to unauthorized users.*

##### Summary

Section 8.7.6 *Device Locking* in the TSS of [ST] states that when the TOE device is locked, the class key for the 'Complete Protection class' is wiped 10 seconds after locking, making files in that class inaccessible.

#### Guidance Assurance Activities

##### Assurance Activity AA-FTA_SSL_EXT.1-AGD-01

*The evaluation shall verify that the AGD guidance describes the method of setting the inactivity interval and of commanding a lock. The evaluator shall verify that the TSS describes the information allowed to be displayed to unauthorized users.*

##### Summary

[CCGUIDE] section 5.6.3 *Device/Session Locking* provides related guidance. It states that the TOE device is locked after a configurable time of user inactivity or upon request of the user. This can be defined by an administrator using a Configuration Profile by setting the configuration key *maxInactivity* in the Passcode Payload to the desirable time. Also, a sample Configuration Profile for configuring passcode restrictions is provided in *Appendix A: Configuration Profiles* of [CCGUIDE] .

Section 8.3.1 *Overview of Key Management* in the TSS of [ST] describes the information allowed to be displayed to unauthorized users.

#### Test Assurance Activities

##### Assurance Activity AA-FTA_SSL_EXT.1-ATE-01

- ***Test 1:*** *The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF_EXT.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.*
- ***Test 2:*** *The evaluator shall command the TSF to transition to the locked state according to the AGD guidance as both the user and the administrator. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.*

**Summary**

Test 1: The evaluator waited for the device to lock automatically and verified that the screen is overwritten with non-sensitive data. The evaluator verified that only unlock, use of the camera, use of information center, display of information, or use of Control Center or Notification Center was possible.

Test 2: The evaluator re-performed Test 1 but locked the device manually instead with the same results.

## 2.2.7.2 Default TOE Access Banners (FTA_TAB.1)

### TSS Assurance Activities

#### Assurance Activity AA-FTA_TAB.1-ASE-01

*The TSS shall describe when the banner is displayed.*

**Summary**

Section 8.8.3 *Lock Screen / Access Banner Display* in the TSS of the [ST] states that an advisory warning message regarding unauthorized use of the TOE can be defined using an image that is presented during the lock

### Guidance Assurance Activities

No assurance activities defined.

### Test Assurance Activities

#### Assurance Activity AA-FTA_TAB.1-ATE-01

*The evaluator shall also perform the following test:*

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

**Summary**

Test 1: The evaluator used the Apple Configurator 2 to create a configuration profile with a specific lock screen message, loaded the profile on the device, and verified that the information was displayed on the lock screen.

## 2.2.7.3 Wireless Network Access (FTA_WSE_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FTA_WSE_EXT.1-WLAN-ASE-01

*[TD0470] The evaluator shall examine the TSS to determine that it defines SSIDs as the attribute to specify acceptable networks.*

### Summary

Section 8.9.1 *EAP-TLS and TLS* in the TSS of [ST] describes the attributes that can be used to specify acceptable networks. This section states that when an application attempts to establish a trusted channel, the TOE will compare the Subject Alternative Name (SAN) contained within the peer certificate (specifically the SAN fields, IP Address, and Wildcard certificate if applicable) to the Fully Qualified Domain Name (FQDN) of the requested server. The Common Name (CN) is ignored. If the FQDN in the certificate does not match the expected SAN for the peer, then the application cannot establish the connection.

## Guidance Assurance Activities

### Assurance Activity AA-FTA_WSE_EXT.1-WLAN-AGD-01

*[TD0470] The evaluator shall examine the operational guidance to determine that it contains guidance for configuring the list of SSID that the WLAN Client is able to connect to.*

### Summary

Section 5.6.7 *Enable/Disable Cellular, Wi-Fi, Wi-Fi Hotspot, Bluetooth, NFC, UWB* of [CCGUIDE]

provides guidance to the adminstrator to configure the list of SSID that the WLAN client is able to connect to. This involves using the *SSID_STR* key in the Wi-Fi Payload in a Configuration Profileand also make sure the *DomainName* key must not be set.

## Test Assurance Activities

### Assurance Activity AA-FTA_WSE_EXT.1-WLAN-ATE-01

*[TD0470] The evaluator shall also perform the following test:*

- *Test 1: The evaluator configures the TOE to allow a connection to a wireless network with a specific SSID. The evaluator also configures the test environment such that the allowed SSID and an SSID that is not allowed are both “visible” to the TOE. The evaluator shall demonstrate that they can successfully establish a session with the allowed SSID. The evaluator will then attempt to establish a session with the disallowed SSID, and observe that the attempt fails.*

### Summary

Test 1: The evaluator deployed a configuration profile on the TOE that allows the TOE to only connect to WiFi networks installed by WiFi payloads. The evaluator could then connect the TOE to the test Access point (AP). The evaluator removed the profile and tried to force the TOE to connect to the same WiFi AP and verified that it failed.

Test 2: This test is removed by TD0470.

# 2.2.8 Trusted path/channels (FTP)

## 2.2.8.1 Bluetooth Encryption (FTP_BLT_EXT.1)

## TSS Assurance Activities

### Assurance Activity AA-FTP_BLT_EXT.1-ASE-01

*The evaluator shall verify that the TSS describes the use of encryption, the specific Bluetooth protocol(s) it applies to, and whether it is enabled by default.*

*The evaluator shall verify that the TSS includes the protocol used for encryption of the transmitted data and the key generation mechanism used.*

**Summary**

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- The TOE 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.
- The TOE generates a new ephemeral ECDH key pair for every new connection attempt. No data can be transferred via Bluetooth until pairing has been completed. The TOE terminates the connection if the remote device stops encryption while connected to the TOE.
- discoverable). Connections via BR/EDR and LE are secured using 128-bit AES Counter with CBC-MAC (AES-CCM-128) mode. No other key sizes are supported; thus, smaller key sizes cannot be negotiated.

### Guidance Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.1-AGD-01

*The evaluator shall verify that the operational guidance includes instructions on how to configure the TOE to require the use of encryption during data transmission (unless this behavior is enforced by default).*

**Summary**

Section 5.3.4 *Bluetooth Configuration* states the following: iOS/iPadOS requires that remote Bluetooth devices use an encrypted connection. Connections via Bluetooth BR/EDR and LE are secured using AES 128 in CCM mode. Further information about Bluetooth security is found in [BT]. This behavior requires no additional configuration by the mobile device administrator.

### Test Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.1-ATE-01

*There are no test EAs for this component. Testing for this SFR is addressed through the evaluation of FTP_BLT_EXT.3/BR and, if claimed, FTP_BLT_EXT.3/LE.*

**Summary**

[MOD_BT_V1.0] specifies no tests for this Assurance Activity.

## 2.2.8.2 Persistence of Bluetooth Encryption (FTP_BLT_EXT.2)

### TSS Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.2-ASE-01

*The evaluator shall verify that the TSS describes the TSF's behavior if a remote device stops encryption while connected to the TOE.*

**Summary**

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- No data can be transferred via Bluetooth until pairing has been completed. The TOE terminates the connection if the remote device stops encryption while connected to the TOE.

### Guidance Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.2-AGD-01

> *The evaluator shall verify that the operational guidance describes how to enable/disable encryption (if configurable).*

#### Summary

Section 5.3.4 *Bluetooth Configuration* of [CCGUIDE] states the following:

- iOS/iPadOS requires that remote Bluetooth devices use an encrypted connection. Connections via Bluetooth BR/EDR and LE are secured using AES 128 in CCM mode. This behavior requires no additional configuration by the mobile device administrator.

### Test Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.2-ATE-01

> *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*
>
> *Step 1: Initiate pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE.*
>
> *Step 2: After pairing has successfully finished and while a connection exists between the TOE and the remote device; turn off encryption on the remote device. This can be done using commercially-available tools.*
>
> *Step 3: Verify that the TOE either restarts encryption with the remote device or terminates the connection with the remote device.*

#### Summary

The evaluator used BlueZ to disable the requirement for a secure connection on the Linux test system. Then the evaluator paired the Linux test system with the TOE, normally, with encryption, which was shown by pcap output captured using hcidump. The evaluator then disabled encryption on the Linux test system using hcitool and verified the Bluetooth connection was terminated.

## 2.2.8.3 Bluetooth Encryption Parameters (BR/EDR) (FTP_BLT_EXT.3/BR)

### TSS Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.3-BR-ASE-01

> *The evaluator shall examine the TSS and verify that it specifies the minimum key size for BR/EDR encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.*

#### Summary

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- Connections via BR/EDR and LE are secured using 128-bit AES Counter with CBC-MAC (AES-CCM-128) mode. No other key sizes are supported; thus, smaller key sizes cannot be negotiated.

### Guidance Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.3-BR-AGD-01

*The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for BR/EDR encryption, if configurable.*

#### Summary

Section 5.3.4 *Bluetooth Configuration* of [CCGUIDE] states the following:

- The TOE requires that remote Bluetooth devices use an encrypted connection. Connections via Bluetooth BR/EDR and LE are secured using AES-128 in CCM mode. This behavior requires no additional configuration by the mobile device administrator.

### Test Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.3-BR-ATE-01

*The evaluator shall perform the following tests:*

- ***Test 1:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*
  *Step 1: Initiate BR/EDR pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.*
  *Step 2: Use a Bluetooth packet sniffer to verify that the encryption key size negotiated for the connection is at least as large as the minimum encryption key size defined for the TOE.*
- ***Test 2:*** *(conditional): If the encryption key size is configurable, configure the TOE to support a different minimum key size, then repeat Test 1 and verify that the negotiated key size is at least as large as the new minimum value.*
- ***Test 3:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*
  *Step 1: Initiate BR/EDR pairing with the TOE from a remote Bluetooth device that has been configured to have a maximum encryption key size of 1 byte. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.*
  *Step 2: Verify that the encryption key size suggested by the remote device is not accepted by the TOE and that the connection is not completed.*

#### Summary

Test 1: Using BlueZ software, the evaluator initiated BR/EDR pairing with the TOE and verified the key length of 16 bytes in traffic captured using hcidump.

Test 2 is not applicable because the key size is not configurable in the TOE.

Test 3: The evaluator used a modified Linux kernel which changed the Bluetooth key size to 1 and attempted the same pairing with the TOE and found that the connection was not established.

## 2.2.8.4 Bluetooth Encryption Parameters (LE) (FTP_BLT_EXT.3/LE)

### TSS Assurance Activities

#### Assurance Activity AA-FTP_BLT_EXT.3-LE-ASE-01

*The evaluator shall examine the TSS and verify that it specifies the minimum key size for LE encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.*

**Summary**

Section 8.9.2 *Bluetooth* in the TSS of [ST] states the following:

- Connections via BR/EDR and LE are secured using 128-bit AES Counter with CBC-MAC (AES-CCM-128) mode. No other key sizes are supported; thus, smaller key sizes cannot be negotiated.

## Guidance Assurance Activities

### Assurance Activity AA-FTP_BLT_EXT.3-LE-AGD-01

*The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for LE encryption, if configurable.*

**Summary**

Section 5.3.4 *Bluetooth Configuration* of [CCGUIDE] states the following:

- The TOE requires that remote Bluetooth devices use an encrypted connection. Connections via Bluetooth BR/EDR and LE are secured using AES-128 in CCM mode. This behavior requires no additional configuration by the mobile device administrator.

## Test Assurance Activities

### Assurance Activity AA-FTP_BLT_EXT.3-LE-ATE-01

*The evaluator shall perform the following tests:*

- ***Test 1:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*
  *Step 1: Initiate LE pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.*
  *Step 2: Use a Bluetooth packet sniffer to verify that the encryption key size negotiated for the connection is at least as large as the minimum encryption key size defined for the TOE.*
- ***Test 2:*** *(conditional): If the encryption key size is configurable, configure the TOE to support a different minimum key size, then repeat Test 1 and verify that the negotiated key size is at least as large as the new minimum value.*
- ***Test 3:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*
  *Step 1: Initiate LE pairing with the TOE from a remote Bluetooth device that has been configured to have a maximum encryption key size of 1 byte. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.*
  *Step 2: Verify that the encryption key size suggested by the remote device is not accepted by the TOE and that the connection is not completed.*

**Summary**

Test 1: Using BlueZ software, the evaluator initiated LE pairing with the TOE and verified the key length of 16 bytes in traffic captured using hcidump.

Test 2 is not applicable because the key size is not configurable in the TOE.

Test 3: The evaluator used a modified Linux kernel which changed the Bluetooth key size to 1 and attempted to the same pairing with the TOE and found that the connection was not established.

## 2.2.8.5 Trusted Channel Communication (FTP_ITC_EXT.1)

### TSS Assurance Activities

#### Assurance Activity AA-FTP_ITC_EXT.1-ASE-01

*The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to access points, VPN Gateways, and other trusted IT products in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specifications. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.*

*If OTA updates are selected, the TSS shall describe which trusted channel protocol is initiated by the TOE and is used for updates.*

**Summary**

Section 8.9 *Trusted Path/Channels (FTP)* in the TSS of [ST] describes the trusted channel protocols supported by the TOE including EAP-TLS, TLS, HTTPS, IPsec, Wi-Fi, and Bluetooth as outlined in table 16 "Protocols used for trusted channels" .

Section 8.9.1 *EAP-TLS and TLS* discusses the EAP-TLS and TLS protocols which states that the TOE supports EAP-TLS using TLS versions 1.0, 1.1, and 1.2 with ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC5246
- 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

and TLS version 1.2 with 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
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
- 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

Also, certificate pinning is supported by the TOE.

Section 8.9.2 *Bluetooth* describes the Bluetooth protocol which states that the TOE supports Bluetooth 4.2 and 5.0 including Basic Rate/Enhanced Data Rate (BR/EDR) and Low Energy (LE). Additional specification about the Bluetooth protocol are specific in this section about how to set up and use the Bluetooth protocol.

Section 8.9.3 *Wireless LAN (WLAN)* describes the wireless LAN protocol in accordance to IEEE 802.11 (2012) which includes the TOE implementing CRT with CCMP with 128-bit AES. Also, the TOE implements AES key wrapping in accordance to NIST SP 800-38F.

Section also 8.9.3 states that newer device models support AES-CCMP-256 with 256-bit keys following IEEE 802.11ac-2013 as well as AES-GCMP-256 with 256-bit key sizes, respectively, following IEEE 802.11ac-2013.

Section 8.9.4 *VPN* describes the VPN including description of the AlwaysOn VPN, the IPsec and IKE protocols. This section describes the specific modes and algorithms used by IKE2 and IPsec as follows:

- IKEv2 (as defined in RFCs 7296 and 4307)
- Tunnel Mode
- Symmetric algorithms for IKE and ESP encryption (AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256)
- Integrity mechanisms (HMAC-SHA1, HMAC-SHA-256, HMAC-SHA-384, and HMAC- SHA-512),
- Key Exchange (Diffie-Hellman Groups):
  - DH Group 14(2048-bit MODP),
  - DH Group 15(3072-bit MODP),
  - DH Group 19(256-bit Random ECP), and
  - DH Group 20(384-bit Random ECP)

For IKE, some of the characteristics include the TOE supports IKEv2 Phase 1 and Phase 2 SAs. Also, during negotiation the TOE will only negotiate the configured algorithms which must include an IKEv2/IKE_SA at least that of IKEv2 CHILD_SA.

The ST selects OTA updates. Section 8.7.3 *Secure Software Update* describes trusted update including how OTA updates are performed. Per this section, OTA is performed via HTTPS trusted channel. During a TOE OS upgrade the TOE device itself connects to the Apple installation authorization server.

## Guidance Assurance Activities

### Assurance Activity AA-FTP_ITC_EXT.1-AGD-01

> *The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to access points, VPN Gateways, and other trusted IT products.*

### Summary

[CCGUIDE] section 5.3 *Network Protocols* provided related guidance for configuring the following network protocols supported by the TOE:

- EAP-TLS
- TLS
- IPsec
- Bluetooth
- VPN

Guidance pertaining to EAP-TLS is provided in section 5.3.1 *EAP-TLS Configuration* of [CCGUIDE] . Such guidance is already assessed in related assurance activities.

Guidance pertaining to TLS is provided in section 5.3.2 *TLS Configuration* of [CCGUIDE] . Such guidance is already assessed in related assurance activities.

Guidance pertaining to IPsec is provided in partly in section 5.3.3 *IPsec Configuration* and furthermore in section 5.3.5 *VPN Configuration* of [CCGUIDE] . Such guidance is already assessed in related assurance activities.

Guidance pertaining to Bluetooth is provided in section 5.3.4 *Bluetooth Configuration* of [CCGUIDE] . Such guidance is already assessed in related assurance activities.

Guidance pertaining to VPN is provided in section 5.3.5 *VPN Configuration* of [CCGUIDE] . This section particularly states that in the evaluated configuration, the VPN must be in its Always-On configuration which is configured by the administrator using the VPN Policy Payload with the key *VPNType* setting to 'AlwaysOn' and the key 'ProtocolType' to IKEv2. Section 5.3.6 *Keys for Configuring Network Protocols* outline in Table 12 "Essential Keys for the VPN Payload" the additional keys and key values that must or must not be used in the evaluated configuration.

### Test Assurance Activities

#### Assurance Activity AA-FTP_ITC_EXT.1-ATE-01

*The evaluator shall also perform the following tests for each protocol listed:*

- ***Test 1:*** *The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext and that a protocol analyzer identifies the traffic as the protocol under testing.*
- ***Test 2:*** *[conditional] If IPsec is selected, the evaluator shall ensure that the TOE is able to initiate communications with a VPN Gateway, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 3:*** *[conditional]If OTA updates are selected, the evaluator shall trigger an update request according to the operational guidance and shall ensure that the communication is successful.*
- ***Test 4:*** *For any other selected protocol (not tested in Test 1, 2, or 3), the evaluator shall ensure that the TOE is able to initiate communications with a trusted IT product using the protocol, setting up the connection as described in the operational guidance and ensuring that the communication is successful.*

#### Summary

Test 1 is covered by all tests using HTTPS, such as those in FCS_TLSC_EXT.1.

Test 2 is covered by testing performed for FDP_IFC_EXT.1 and VPN testing.

Test 3 is covered by testing performed for FMT_SMF_EXT.1 Test 15 and FPT_TUD_EXT.6 Test 2.

Test 4: TLS testing has been previously described. Bluetooth tests are covered by the testing for FTP_BLT_EXT SFRs.

## 2.2.8.6 Trusted Channel Communication (FTP_ITC_EXT.1(2))

### TSS Assurance Activities

#### Assurance Activity AA-FTP_ITC_EXT.1-2-MDMA-ASE-01

*The evaluator shall examine the TSS to determine that the methods of Agent-Server communication are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of remote TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.*

#### Summary

Per FTP_ITC_EXT.1(2) {AGENT}, the TOE uses HTTPS to communicate with an MD agent.

Per section 8.5.3 *MDM Server Reference ID* , the MDM Agent-Server communication is over HTTPS. This description is supported by the description provided in section 8.10.2 *MDM Agent Alerts* of the TSS which states that the MDM Agent initiates communication with the MDM Server in response to a push notification by establishing an HTTPS connection to the MDM Server URL.

This description is also consistent with the definition of FTP_TRP.1(2) (trusted path for the MDM agent) which specifies HTTPS as the protocol used for protecting remote TOE administration.

## Guidance Assurance Activities

### Assurance Activity AA-FTP_ITC_EXT.1-2-MDMA-AGD-01

*The evaluator shall confirm that the operational guidance contains instructions for configuring the communication channel between the MDM Agent and the MDM Server and conditionally, the MAS Server for each supported method.*

#### Summary

[CCGUIDE] section 4.3.3 *Configure MDM Agent and MDM Communications* provides information to configure the communications between the MDM Agent and MDM Server. It states as follows:

MDM Agent-Server communication is achieved securely using the MDM protocol which is built on top of HTTP, transport layer security (TLS), and push notifications that use HTTP PUT over TLS (secure sockets layer (SSL)). A managed mobile device uses an identity to authenticate itself to the MDM server over TLS (SSL). This identity can be included in the profile as a Certificate Payload or can be generated by enrolling the mobile device with Simple Certificate Enrollment Protocol (SCEP).

Also, the MDM Agent communications use the TOE OS Security Framework as described in section 5.3.2 *TLS Configuration* of [CCGUIDE] . Thus, configuring the TOE's TLS protocol as per section 5.3.2 automatically configures the MDM Agent communications. If an additional CA certificate needs to be added to support the MDM Server, information is provided in this section. The evaluator examined these sections and determined that they provide related guidance.

## Test Assurance Activities

### Assurance Activity AA-FTP_ITC_EXT.1-2-MDMA-ATE-01

*For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:*

- ***Test 1:*** *The evaluators shall ensure that communications using each specified (in the operational guidance) Agent-Server communication method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 2:*** *The evaluator shall ensure, for each method of Agent-Server communication, the channel data is not sent in plaintext.*
- ***Test 3:*** *The evaluator shall ensure, for each communication channel with the MDM Server, that a protocol analyzer identifies the traffic as the protocol under testing.*

*Further evaluation activities are associated with the specific protocols.*

#### Summary

The evaluator deployed a WAP (WiFi Access Point) from a test laptop and connected the TOE to it. For both IPsec and HTTPs/TLS, the evaluator connected the TOE to the laptop and used Wireshark to verify that the communication channels were indeed encrypted. Wireshark identified the traffic as either IKE/IPSec or HTTPS/TLS.

For Test 1, the evaluator made a configuration change via MDM and deployed an application to a managed device.

For Tests 2 and 3, the evaluator deployed configuration changes, deployed an application, and enrolled into an MDM server.

# 2.2.8.7 Trusted Channel Communication (Wireless LAN) (FTP_ITC_EXT.1/WLAN)

## TSS Assurance Activities

### Assurance Activity AA-FTP_ITC_EXT.1-WLAN-ASE-01

*The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to an access point in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specification. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.*

### Summary

Section 8.9.3 *Wireless LAN (WLAN)* in the TSS of [ST] describes the WLAN protocol implemented by the TOE. Section 8.9.1 *EAP-TLS and TLS* describes the EAP-TLS and TLS protocols and ciphersuites supported by the TOE.

Section 8.9.3 specifies which protocols are implemented according to the IEEE 802.11 (2012) standard for WLAN and which algorithms are supported by the TOE. Also, this section states that newer device models support AES-CCMP-256 with 256-bit keys following IEEE 802.11ac-2013 as well AES-GCMP-256 with 256-bit key sizes, respectively, following IEEE 802.11ac-2013. Additionally, section 8.9.1 specifies additional information about the EAP-TLS and TLS protocols certificates and ciphersuites that can be requested by applications.

## Guidance Assurance Activities

### Assurance Activity AA-FTP_ITC_EXT.1-WLAN-AGD-01

*The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to the access point, and that it contains recovery instructions should a connection be unintentionally broken.*

### Summary

[CCGUIDE] section 5.5.6 *X.509 Certificate Configuration* states that to enforce the verification of the server name defined with the X.509 certificate during the WPA-EAP handshake between the TOE and the remote access point, the policy must contain the server name to be expected in the certificate with the TLSTrustedServerNames dictionary key in the Wi-FI Payload EAPClientConfiguration Dictionary of the Configuration Profiles. Additionally, a sample Configuration Profile for configuring the WLAN is provided in chapter Appendix C *Configuration Profiles* of [CCGUIDE]

Additionally, configuration for EAP-TLS is provided in [CCGUIDE] section 5.3.1 *EAP-TLS Configuration* which states that no additional configuration is needed if the automatic recovery of a broken Wi-Fi connection.

## Test Assurance Activities

### Assurance Activity AA-FTP_ITC_EXT.1-WLAN-ATE-01

*The evaluator shall perform the following tests:*

- *Test 1: The evaluators shall ensure that the TOE is able to initiate communications with an access point using the protocols specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.*

- *Test 2: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.*
- *Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, modification of the channel data is detected by the TOE.*
- *Test 4: The evaluators shall physically interrupt the connection from the TOE to the access point (e.g., moving the TOE host out of range of the access point, turning the access point off). The evaluators shall ensure that subsequent communications are appropriately protected, at a minimum in the case of any attempts to automatically resume the connection or connect to a new access point.*

*Further assurance activities are associated with the specific protocols.*

**Summary**

Test 1: The evaluator deployed a configuration profile on the TOE and verified that the TOE could successfully connect to the test AP.

Test 2: The evaluator used the same profile as in Test 1 and used Wireshark to sniff the wireless traffic when the connection is established with the AP. The evaluator could verify that the traffic is encrypted using EAP-TLS.

Test 3: The evaluator verified that Test 2 uses the EAP-TLS protocol.

Test 4: The evaluator deployed the same configuration profile on the TOE, terminated the connection on the AP side, and verified that the connection was terminated on the TOE. The evaluator restarted the AP on the test laptop and verified that the TOE automatically resumes the connection.

## 2.2.8.8 Trusted Path (for Enrollment) (FTP_TRP.1(2))

### TSS Assurance Activities

#### Assurance Activity AA-FTP_TRP.1-2-MDMA-ASE-01

*The evaluator shall examine the TSS to determine that the methods of remote enrollment are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of enrollment are consistent with those specified in the requirement, and are included in the requirements in the ST.*

**Summary**

Per FTP_TRP.1(2), the TOE uses HTTPS to provide a trusted channel for remote enrollment.

Section 8.5.3 *MDM Server Reference ID* states that the enrollment is initiated by the TOE device and performed over an HTTPS connection. The evaluator verifies that the ST includes the respective requirements FCS_HTTPS_EXT.1 and FCS_TLSC_EXT.1 (as mandated by the MDF PP).

### Guidance Assurance Activities

#### Assurance Activity AA-FTP_TRP.1-2-MDMA-AGD-01

*The evaluator shall confirm that the operational guidance contains instructions for establishing the enrollment sessions for each supported method.*

**Summary**

Section 4.3 *Mobile Device Supervision and Configuration* provides guidance for device enrollment. It states that the methods include:

- The Apple Business Manager (ABM), which provides an automated and enforced method of automatically enrolling new devices
- Using Apple's Profile Manager, which provides a manual method of enrolling mobile devices
- Using the Apple Configurator 2, which provides both automated and manual methods of enrolling mobile devices
- Using Email or a Website, which provides a way to distribute an enrollment profile to a mobile device

Guidance for enrolling via the Apple Business Manager (ABM) is provided in subsection 4.3.1.1 *Apple Business Manager* . Additional guidance is provided in [ABM_Guide] , [PM_Help] , and [DEV_MAN] .

Guidance for enrolling via the Apple Profile Manager is provided in subsection 4.3.1.2 *Apple Profile Manager* . Additional guidance is provided in [PM_Help] .

Guidance for enrolling via the Apple Configurator is provided in section 4.3.1.3 *Apple Configurator* . Additional guidance is provided in subsection [AConfig] .

Guidance for enrolling via other methods such as email or website is provided in subsection 4.3.1.4 *Other Methods* .

The evaluator examined these sections including the referenced supporting guides and determined that they provide sufficient guidance for establishing enrollment sessions for each supported method.

## Test Assurance Activities

### Assurance Activity AA-FTP_TRP.1-2-MDMA-ATE-01

*For each MDM Agent/platform listed as supported in the ST:*

- ***Test 1:*** *The evaluators shall ensure that communications using each specified (in the operational guidance) enrollment method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 2:*** *For each method of enrollment supported, the evaluator shall follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish enrollment sessions without invoking the trusted path.*
- ***Test 3:*** *The evaluator shall ensure, for each method enrollment, the channel data is not sent in plaintext.*

*Further evaluation activities are associated with the specific protocols.*

### Summary

These tests are included in testing for FTP_ITC_EXT.1(2).

# 2.3 Security Assurance Requirements

## 2.3.1 Development (ADV)

### 2.3.1.1 Basic functional specification (ADV_FSP.1)

#### Assurance Activity AA-ADV_FSP.1-ADV-01

*There are no specific evaluation activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1 Security Functional Requirements in* [MDFPPv3.2] *, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other evaluation activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.*

**Summary**

The evaluator performed this assurance activity while performing other activities described for AGD, ATE, and AVA SARs. Based on successful assessment of those activities, the evaluator concluded that the provided evidence contain sufficient interface information, thus an adequate functional specification has been provided.

## 2.3.2 Guidance documents (AGD)

### 2.3.2.1 Operational user guidance (AGD_OPE.1)

#### Assurance Activity AA-AGD_OPE.1-AGD-01

*Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements in* [MDFPPv3.2] *and evaluation of the TOE according to the* [CEM] *. The following additional information is also required.*

*The operational guidance shall contain a list of natively installed applications and any relevant version numbers. If any third party vendors are permitted to install applications before purchase by the end user or enterprise, these applications shall also be listed.*

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

*The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:*

- *Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).*
- *Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.*

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

**Summary**

**Installed apps**

[CCGUIDE] chapter 7 *Installed Apps* contains Table 222 "Built-in and Presintalled Apps" listing the apps that come with the TOE devices. Built-in apps cannot be removed while preinstalled apps are included with purchased devices but may be removed by the user or administrator.

It is noted that Table 22 applies to both iPad and iPhone devices, however the evaluator was only concerned with the app listing for the iPhone TOE for this evaluation

**Cryptographic modules**

[CCGUIDE] section 5.2 *Cryptographic Support Functions* provides information about the cryptographic modules that provide cryptographic support to the TOE. It states that the TOE comes with the following three cryptographic modules that provide the cryptographic support for the TOE:

- Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1] (User Space)
- Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1] (Kernel Space)
- Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2]

Section 5.2 also contains the following warning:

> " ***Warning:*** *The use of other cryptographic engines beyond those listed above was neither evaluated nor tested during the mobile device's Common Criteria evaluation.* "

**Secure software updates**

[CCGUIDE] section 5.6.10 *Secure Software Updates* describes the procedure for obtaining and verifying updates to the TOE. The process is as follows:

- All TOE updates are digitally signed by Apple.
- The TOE's feature System Software Authorization prevents downgrading of the TOE OS version.
- Updates can be installed using iTunes or over-the-air (OTA) on the device. With iTunes, a full copy of the iOS/iPadOS is downloaded (via TLS) and installed. OTA software updates download only the components required to complete an update.
- Updates may also be installed manually using Finder on macOS versions 10.15.0 (Catalina) and higher, or manually using iTunes on macOS versions prior to 10.15.0 and on PCs. A USB connection between the computer and the device is necessary to perform updates using Finder or iTunes.
- The TOE OS updates can be cached on a local network server running the caching service on macOS server so that TOE devices do not need to access Apple servers to obtain the necessary update data.
- Users can update the TOE OS on their device by following the instructions provided in the user guide [iPhone_UG] section *Restart, update, reset, and restore* subsection *Update iOS on iPhone*.

**Security functionality provided by the TOE:**

[CCGUIDE] section 3.2 *TOE Security Functionality* lists at a high-level the security functionality provided by the TOE in the evaluated configuration which covers:

- Security audit
- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Protection of the TOE Security Functionality (TSF)
- TOE access
- Trusted path/channels

While assessing other assurance activities for the guidance documentation, the evaluator also verified that the guidance covers the security functionalities listed above. The evaluator noticed that in particular, [CCGUIDE] chapters 5 and 6 are largely structured based on these security functionalities, for example, section 5.5 and section 5.6 are labeled as *Identification & Authentication* and *Security Management* , respectively. This structure makes it easy for the reader to follow as well as for the evaluator to verify which security functionalities from [PP_MDF_V3.2] , [MOD_MDM_AGENT_V1.0] , [MOD_BT_V1.0], [MOD_VPNC_V2.3], [PP_WLAN_CLI_EP_V1.0] , and [PKG_TLS_V1.1] are covered in the guidance/assurance activities.

Furthermore, section 3.9 *Un-evaluated Functionalities* of [CCGUIDE] explicitly lists the security functionalities that are outside the scope of the evaluated configuration.

## 2.3.2.2 Preparative procedures (AGD_PRE.1)

### Assurance Activity AA-AGD_PRE.1-AGD-01

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

### Summary

While performing other assurance activities the evaluator determined that sufficient guidance was provided for the TOE to address all claimed platforms. The guidance provided for the TOE applies to all platforms claimed in [ST] .

# 2.3.3 Life-cycle support (ALC)

## 2.3.3.1 Labelling of the TOE (ALC_CMC.1)

### Assurance Activity AA-ALC_CMC.1-ALC-01

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

### Summary

[ST] section 1.2 *TOE Reference* identifies the TOE as "Apple iOS 15: iPhones" and references *Annex A: Devices Covered by this Evaluation* which lists the devices that are covered by this evaluation and gives the technical characteristics of each device. These devices use either A9 processor (iPhone 6s, iPhone 6s Plus, iPhone SE), A10 Fusion processor (iPhone 7, iPhone 7 Plus), A11 Bionic processor (iPhone 8, iPhone 8 Plus, iPhone X), A12 Bionic (iPhone Xs, iPhone Xs Max, iPhone XR), A13 Bionic (iPhone 11, iPhone 11 Pro, iPhone 11 Pro Max, iPhone SE (2nd gen)). A14 Bionic (iPhone 12 mini, iPhone 12, iPhone 12 Pro, iPhone 12 Pro Max). A15 Bionic (iPhone 13, iPhone 13 mini, iPhone 13 Pro, iPhone 13 Pro Max)

The guidance documentation, for example, [CCGUIDE] contains TOE reference as Apple iOS 15 on iPhone.

In addition, for the delivery of the evaluated TOE devices, [CCGUIDE] section 4 *Secure Delivery and Installation* states the following:

The evaluated mobile devices are intended for authorized mobile device users of entities such as business organizations and government agencies.

The normal distribution channels for a regular end user to obtain these devices include:

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

**Business specific distribution channel**

There is a distinct online store for Business customers with a link from the "Apple Store" to Apple and Business: (https://www.apple.com/business/). Additionally, link to "Shop for Business" is provided (https://www.apple.com/retail/business/).

**Government specific distribution channel**

Government customers can use the link: https://www.apple.com/r/store/government/

**Additional**

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

## 2.3.3.2 TOE CM coverage (ALC_CMS.1)

### Assurance Activity AA-ALC_CMS.1-ALC-01

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

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

**Summary**

The evaluator examined the lifecycle evidence provided by the developer which contains the following information:

The developer uses the Xcode Integrated Development Environment (IDE) that is used by developers for building apps for macOS, iOS, iPadOS, watchOS, and tvOS. It is the only officially-supported tool for creating and publishing applications to Apple's application store. Xcode provides all of the tools needed to create an app within one software package; namely, a text editor, a compiler, and a build system.

As a code editor, Xcode supports a huge variety of programming languages - C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit, and Swift. It uses Cocoa, Carbon, and Java programming models.

The vendor provided "Apple Platform Security" [AP_SEC] which states the following: Beginning with iOS 14 and iPadOS 14, Apple modified the C compiler toolchain used to build the iBoot bootloader to improve its security. The modified toolchain implements code designed to prevent memory- and type-safety issues that are typically encountered in C programs. For example, it helps prevent most vulnerabilities in the following classes:

- Buffer overflows, by ensuring that all pointers carry bounds information that is verified when accessing memory
- Heap exploitation, by separating heap data from its metadata and accurately detecting error conditions such as double free errors
- Type confusion, by ensuring that all pointers carry runtime type information that's verified during pointer cast operations
- Type confusion caused by use after free errors, by segregating all dynamic memory allocations by static type

The "Apple Platform Security" [AP_SEC] provides a section *Security of runtime process in iOS and iPadOS* to ensure runtime security by using a "sandbox," declared entitlements, and Address Space Layout Randomization (ASLR). It states the following:

- **Sandboxing**
  - All third-party apps are "sandboxed," so they are restricted from accessing files stored by other apps or from making changes to the device. Sandboxing is designed to prevent apps from gathering or modifying information stored by other apps. Each app has a unique home directory for its files, which is randomly assigned when the app is installed. If a third-party app needs to access information other than its own, it does so only by using services explicitly provided by iOS and iPadOS.
  - System files and resources are also shielded from the users' apps. Most iOS and iPadOS system files and resources run as the nonprivileged user "mobile," as do all third-party apps. The entire operating system partition is mounted as read-only. Unnecessary tools, such as remote login services, aren't included in the system software, and APIs don't allow apps to escalate their own privileges to modify other apps or iOS and iPadOS.
- **Use of entitlements**
  - Access by third-party apps to user information, and to features such as iCloud and extensibility, is controlled using declared entitlements. Entitlements are key-value pairs that are signed in to an app and allow authentication beyond runtime factors, like UNIX user ID. Since entitlements are digitally signed, they can't 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 app or daemon.
  - In addition, apps can only perform background processing through system-provided APIs. This allows apps to continue to function without degrading performance or dramatically impacting battery life.
- **Address Space Layout Randomization**
  - Address Space Layout Randomization (ASLR) helps protect against the exploitation of memory corruption bugs. Built-in apps use ASLR to help randomize all memory regions upon launch. In addition to work upon launch, ASLR randomly arranges the memory addresses of executable code, system libraries, and related programming constructs, further reducing the likelihood of many exploits. For example, a return-to-libc attack attempts to trick a device into executing malicious code by manipulating memory addresses of the stack and system libraries.

Randomizing the placement of these makes the attack more difficult to execute, especially across multiple devices. **Xcode, and the iOS or iPadOS development environments, automatically compile third-party programs with ASLR support turned on**.

## 2.3.4 Tests (ATE)

### 2.3.4.1 Independent testing - conformance (ATE_IND.1)

#### Assurance Activity AA-ATE_IND.1-ATE-01

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

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

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

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

**Summary**

The Detailed Test Report [DTR] specifies all the tests covering the assurance activities from [PP_MDF_V3.2], [MOD_MDM_AGENT_V1.0], [MOD_VPNC_V2.3], [PP_WLAN_CLI_EP_V1.0], [PKG_TLS_V1.1], and [MOD_BT_V1.0]. The [DTR] is provided to the certification body but is not published.

The test environment was set up according to a setup strategy that followed the evaluated configuration requirements specified in the guidance, supplemented by configurations required to perform testing. The test configuration is made up of the TOE connected to a private WLAN network which also hosts a Linux system and a macOS Server. The macOS Server is created on a macOS system upon which the macOS Server software, available from the Apple app store, is installed.

The Linux server runs Fedora 35 and provides the following support for all tests.

- WLAN access point functionality,
- Internet access,
- network sniffer tools,
- a VPN Gateway with the strongSwan IKE daemon and the Linux kernel IPsec support.

using the following installed tools

- Iptables version 1.8.7 for enabling masquerading in the Linux kernel

- netcat version 7.91 (installed from Linux distro repository)
- dnsmasq version 2.86 (installed from Linux distro repository)
- hostapd, version 2.9 (regular tests) and 2.4 (tests deliberately breaking the connection)
- tcpdump, version 4.99.1 (using libpcap version 1.10.1)
- hcidump and bdaddr from the Bluez Linux Bluetooth protocol stack version 5.62. (Bluetooth-specific protocol analyzer and manipulator)
- OpenSSL with s_server and s_client applications version 1.1.1l
- Wireshark 3.4.9 with libpcap 1.10.1.
- Strongswan 5.9.4

The macOS server runs macOS 11 and provides the following support for MDM-related tests.

- MDM server
- Apple Configurator 2

using the following installed tools

- Apple Configurator 2.13 (AKA Apple Configurator 2)
- Xcode 12.2

## 2.3.5 Vulnerability assessment (AVA)

### 2.3.5.1 Vulnerability survey (AVA_VAN.1)

#### Assurance Activity AA-AVA_VAN.1-AVA-01

*The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in mobile devices and the implemented communication protocols in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report.*

*For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.*

#### Summary

The evaluator searched for publicly known vulnerabilities applicable to iOS using the following sources.

- Apple security content disclosure statements for releases of iOS 15.1 related to this evaluation:
  - https://support.apple.com/en-us/HT212976, iOS 15.2, released December 14, 2021.
  - https://support.apple.com/en-us/HT213043, iOS 15.2.1, released January 12, 2022.
  - https://support.apple.com/en-us/HT213053, iOS 15.3, released January 26, 2022.
  - https://support.apple.com/en-us/HT213093, iOS 15.3.1, released February 10, 2022.
  - https://support.apple.com/en-us/HT213182, iOS 15.4, released March 14, 2022.
  - https://support.apple.com/en-us/HT213219, iOS 15.4.1, released March 31, 2022.
  - https://support.apple.com/en-us/HT213258, iOS 15.5, released May 16, 2022.
  - https://support.apple.com/en-us/HT213346, iOS 15.6, released July 20, 2022.

  - https://support.apple.com/en-us/HT213412, iOS 15.6.1, released August 18, 2022.
  - https://support.apple.com/en-us/HT213445, iOS 15.7, released September 12, 2022.

  **Note:** Not all releases of iOS contain security fixes for publicly known vulnerabilities. iOS release 15.1.1 contained no published security fixes.

- MITRE Common Vulnerabilities and Exposures (CVE) List:
  - https://cve.mitre.org/cve/search_cve_list.html
- NIST National Vulnerability Database (NVD):
  - https://web.nvd.nist.gov/view/vuln/search
- Cybersecurity and Infrastructure Security Agency (CISA):
  - https://www.cisa.gov/known-exploited-vulnerabilities-catalog

The following search terms were used on the MITRE, NIST and CISA web sites:

- ios iphone
- ios core tls
- ios core crypto
- ios common crypto
- ios http
- ios https
- ios tcp
- ios ip
- ios bluetooth
- ios ipsec
- ios vpn
- ios mdm
- ios mobile
- ios touchid
- ios faceid
- broadcom wi-fi

In addition to the lists of fixes published by the vendor, the evaluator performed manual searches on multiple occasions between 2021-10-25 and 2021-10-28, and again on 2021-12-13, 2022-01-14, 2022-02-08, 2022-02-21, 2022-03-15, 2022-04-01, 2022-05-17, 2022-05-26, 2022-07-20, 2022-08-09, 2022-08-18, 2022-09-01, 2022-09-12, 2022-09-26 and 2022-10-10.

The developer publishes security content disclosure statements providing information about vulnerabilities fixed in each release of iOS after 15.1. The developer also provides continuous updates to the TOE, therefore the TOE version tested in this evaluation, iOS 15.1, is no longer available. As of the date of this report, the current version is iOS 15.7. The evaluator's CVE search found no vulnerabilities apart from the ones listed in the security content disclosure statements, all of which have been fixed in subsequent releases of iOS.

# A Appendixes

## A.1 References

ABM_Guide **Apple Business Manager User Guide**
Date April 2022
Location https://support.apple.com/guide/apple-business-manager/welcome/web

AConfig **Apple Configurator 2 User Guide (online)**
Date received 2021
Location https://support.apple.com/guide/apple-configurator-2/welcome/mac

AP_SEC **Apple Platform Security**
Author(s) Apple Inc.
Version May 2022
Date May 2022
File name agd/apple-platform-security-guide.pdf

APFS_DOC **File system formats available in Disk Utility on Mac**
Date 2022
Location https://support.apple.com/en-euro/guide/disk-utility/dsku19ed921c/21.0/mac/12.0

CC **Common Criteria for Information Technology Security Evaluation**
Version 3.1R5
Date April 2017
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf

CCEVS-TD0194 **Update to Audit of FTP_ITC_EXT.1/WLAN**
Date 2017-04-11
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0194

CCEVS-TD0439 **EAP-TLS Revocation Checking**
Date 2019-08-29
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0439

CCEVS-TD0470 **Wireless Network Restrictions**
Date 2020-01-22
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0470

CCEVS-TD0491 **Update to FMT_SMF_EXT.4 Test 2**
Date 2020-01-15
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0491

CCEVS-TD0492 **TLS-EAP Ciphers and TLS versions for WLAN Client**
Date 2020-01-22
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0492

CCEVS-TD0497 **SFR Rationale, Consistency of SPD, and Implicitly Satisfied SFRs**
Date 2020-10-26
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0497

CCEVS-TD0499 **Testing with pinned certificates**
Date 2020-02-04
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0499

CCEVS-TD0513 **CA Certificate loading**
Date 2020-05-26
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0513

CCEVS-TD0517 **WLAN Client Corrections for X509 and TLSC**
Date 2020-06-19
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0517

CCEVS-TD0596 **VPN Traffic Permitted in FDP_IFC_EXT.1**
Date 2021-07-30
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0596

CCEVS-TD0600 **Conformance claim sections updated to allow for MOD_VPNC_V2.3**
Date 2021-08-10
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0600

CCEVS-TD0622 **VPNC MOD FTP_DIT_EXT.1 corrections**
Date 2022-02-07
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0622

CCEVS-TD0623 **TD0623: FIA_X509_EXT.2.1 Protocol Selection**
Date 2022-02-11
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0623

CCEVS-TD0640 **TD0640: Handling BT devices that do not support encryption**
Date 2022-06-15
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0640

CCEVS-TD0646 **TD0623: FIA_X509_EXT.2.1 Protocol Selection**
Date 2022-02-11
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0623

CCEVS-TD0653 **TD0653: MDF v3.2 ASE References**
Date 2022-06-21
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0653

CCEVS-TD0660 **TD0660: Mislabeled SFRs in MDM Agent Auditable Events Table**
Date 2022-07-11
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0660

CCEVS-TD0663 **TD0663: Audit Listing for MDF Moved to Guidance**
Date 2022-08-26
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0663

CCEVS-TD0671 **TD0671: Bluetooth PP-Module updated to allow for new PP and PP-Module Versions**
Date 2022-09-27
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0671

CCEVS-TD0673 **TD0673: MDM-Agent PP-Module updated to allow for new PP and PP-Module Versions**
Date 2022-09-27
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0673

CCGUIDE **Apple iOS 15: iPhones and Apple iPadOS 15: iPads Common Criteria Configuration Guide**
Version 1.2
Date 2022-09-23
File name agd/VID11237_VID11238_CC-guide_v1.2.pdf

CEM **Common Methodology for Information Technology Security Evaluation**
Version 3.1R5
Date April 2017
Location http://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R5.pdf

CFG_MDF-MDMA-VPNC-BT_V1.0 **PP-Configuration for Mobile Device Fundamentals, Mobile Device Management Agents, Virtual Private Network Clients, and Bluetooth**
Author(s) NIAP
Version 1.0
Date 2022-03-07
Location https://www.niap-ccevs.org/MMO/PP/CFG_MDF-MDMA-VPNC-BT_v1.0.pdf

| | | |
|---|---|---|
| CKTSREF | **Certificate, Key, and Trust Services (API Reference)** | |
| | Date | 2021 |
| | Location | https://developer.apple.com/documentation/security/certificate_key_and_trust_services |
| COREBT | **Core Bluetooth Framework** | |
| | Date | 2022 |
| | Location | https://developer.apple.com/documentation/corebluetooth |
| DeployRef | **Apple Platform Deployment** | |
| | Date | 2022 |
| | Location | https://support.apple.com/guide/deployment/welcome/web |
| DEV_MAN | **Device Management** | |
| | Version | April 2021 |
| | Date received | 2021-05-04 |
| | Location | https://developer.apple.com/documentation/devicemanagement |
| DTR | **Apple iOS 15: iPhones Detailed Test Report** | |
| | Version | 1.1 |
| | Date | 2022-11-03 |
| | File name | dtr/VID11237-iOS_15-DTR.v1.1.pdf |
| iPhone_UG | **iPhone User Guide** | |
| | Author(s) | Apple Inc. |
| | Version | iOS 15 |
| | Date | 2022 |
| | Location | https://support.apple.com/guide/iphone/welcome/ios |
| LOGGING | **Logging** | |
| | Date | 2022 |
| | Location | https://developer.apple.com/documentation/os/logging?language=objc |
| MDFPPv3.2 | **Protection Profile for Mobile Device Fundamentals** | |
| | Version | 3.2 |
| | Date | 2021-04-15 |
| | Location | https://www.niap-ccevs.org/MMO/PP/PP_MDF_V3.2.pdf |
| MOD_BT_V1.0 | **PP-Module for Bluetooth** | |
| | Version | 1.0 |
| | Date | 2019-04-15 |
| | Location | https://www.niap-ccevs.org/Profile/Info.cfm?PPID=425&id=425 |
| MOD_MDM_AGENT_V1.0 | **PP-Module for MDM Agents** | |
| | Version | 1.0 |
| | Date | 2019-04-25 |
| | Location | https://www.niap-ccevs.org/Profile/Info.cfm?PPID=441&id=441 |
| MOD_VPNC_V2.3 | **PP-Module for Virtual Private Network (VPN) Clients** | |
| | Version | 2.3 |
| | Date | 2017-10-05 |
| | Location | https://www.niap-ccevs.org/Profile/Info.cfm?PPID=461&id=461 |

| | | |
|---|---|---|
| PKG_TLS_V1.1 | **Functional Package for Transport Layer Security (TLS)** | |
| | Version | 1.1 |
| | Date | 2019-03-01 |
| | Location | https://www.niap-ccevs.org/Profile/Info.cfm?PPID=439&id=439 |
| PM_Help | **Profile Manager User Guide for macOS Monterey** | |
| | Date | 2022 |
| | Location | https://support.apple.com/guide/profile-manager/welcome/mac |
| PP_MDF_V3.2 | **Protection Profile for Mobile Device Fundamentals** | |
| | Version | 3.2 |
| | Date | 2021-04-15 |
| | Location | https://www.niap-ccevs.org/Profile/Info.cfm?PPID=455&id=455 |
| PP_WLAN_CLI_EP_V1.0 | **General Purpose Operating Systems Protection Profile/ Mobile Device Fundamentals Protection Profile Extended Package (EP) Wireless Local Area Network (WLAN) Clients** | |
| | Version | 1.0 |
| | Date | 2016-02-08 |
| | Location | https://www.niap-ccevs.org/Profile/Info.cfm?PPID=386&id=386 |
| PROFS_LOGS | **Profiles and Logs** | |
| | Date | 2022 |
| | Location | https://developer.apple.com/bug-reporting/profiles-and-logs/?platforms=ios |
| ST | **Apple iOS 15: iPhones Security Target** | |
| | Version | 1.2 |
| | Date | 2022-09-29 |
| | File name | ase/st-iOS15_PUBLIC_V1.2.pdf |
| TRUST_STORE | **List of available trusted root certificates in iOS 15, iPadOS 15, macOS 12, tvOS 15, and watchOS 8** | |
| | Date | 2022 |
| | Location | https://support.apple.com/en-us/HT212773 |

# A.2 Glossary

**Augmentation**
The addition of one or more requirement(s) to a package.

**Authentication data**
Information used to verify the claimed identity of a user.

**Authorised user**
A user who may, in accordance with the SFRs, perform an operation.

**Class**
A grouping of CC families that share a common focus.

**Component**
The smallest selectable set of elements on which requirements may be based.

**Connectivity**
The property of the TOE which allows interaction with IT entities external to the TOE. This includes exchange of data by wire or by wireless means, over any distance in any environment or configuration.

**Dependency**
A relationship between components such that if a requirement based on the depending component is included in a PP, ST or package, a requirement based on the component that is depended upon must normally also be included in the PP, ST or package.

**Deterministic RNG (DRNG)**
An RNG that produces random numbers by applying a deterministic algorithm to a randomly selected seed and, possibly, on additional external inputs.

**Element**
An indivisible statement of security need.

**Entropy**
The entropy of a random variable X is a mathematical measure of the amount of information gained by an observation of X.

**Evaluation**
Assessment of a PP, an ST or a TOE, against defined criteria.

**Evaluation Assurance Level (EAL)**
An assurance package, consisting of assurance requirements drawn from CC Part 3, representing a point on the CC predefined assurance scale.

**Evaluation authority**
A body that implements the CC for a specific community by means of an evaluation scheme and thereby sets the standards and monitors the quality of evaluations conducted by bodies within that community.

**Evaluation scheme**
The administrative and regulatory framework under which the CC is applied by an evaluation authority within a specific community.

**Exact conformance**
a subset of Strict Conformance as defined by the CC, is defined as the ST containing all of the requirements in the Security Requirements section of the PP, and potentially requirements from Appendices of the PP. While iteration is allowed, no additional requirements (from the CC parts 2 or 3) are allowed to be included in the ST. Further, no requirements in the Security Requirements section of the PP are allowed to be omitted.

**Extension**
The addition to an ST or PP of functional requirements not contained in Part 2 and/or assurance requirements not contained in Part 3 of the CC.

**External entity**
Any entity (human or IT) outside the TOE that interacts (or may interact) with the TOE.

**Family**
A grouping of components that share a similar goal but may differ in emphasis or rigour.

**Formal**
Expressed in a restricted syntax language with defined semantics based on well-established mathematical concepts.

**Guidance documentation**
Documentation that describes the delivery, preparation, operation, management and/or use of the TOE.

**Identity**
A representation (e.g. a string) uniquely identifying an authorised user, which can either be the full or abbreviated name of that user or a pseudonym.

**Informal**
Expressed in natural language.

**Object**
A passive entity in the TOE, that contains or receives information, and upon which subjects perform operations.

**Operation (on a component of the CC)**
Modifying or repeating that component. Allowed operations on components are assignment, iteration, refinement and selection.

**Operation (on an object)**
A specific type of action performed by a subject on an object.

**Operational environment**
The environment in which the TOE is operated.

**Organisational Security Policy (OSP)**
A set of security rules, procedures, or guidelines imposed (or presumed to be imposed) now and/or in the future by an actual or hypothetical organisation in the operational environment.

**Package**
A named set of either functional or assurance requirements (e.g. EAL 3).

**PP evaluation**
Assessment of a PP against defined criteria.

**Protection Profile (PP)**
An implementation-independent statement of security needs for a TOE type.

**Random number generator (RNG)**
A group of components or an algorithm that outputs sequences of discrete values (usually represented as bit strings).

**Refinement**
The addition of details to a component.

**Role**
A predefined set of rules establishing the allowed interactions between a user and the TOE.

**Secret**
Information that must be known only to authorised users and/or the TSF in order to enforce a specific SFP.

**Secure state**
A state in which the TSF data are consistent and the TSF continues correct enforcement of the SFRs.

**Security attribute**
A property of subjects, users (including external IT products), objects, information, sessions and/or resources that is used in defining the SFRs and whose values are used in enforcing the SFRs.

**Security Function Policy (SFP)**
A set of rules describing specific security behaviour enforced by the TSF and expressible as a set of SFRs.

**Security objective**
A statement of intent to counter identified threats and/or satisfy identified organisation security policies and/or assumptions.

**Security Target (ST)**
An implementation-dependent statement of security needs for a specific identified TOE.

**Seed**
Value used to initialize the internal state of an RNG.

**Selection**
The specification of one or more items from a list in a component.

**Semiformal**
Expressed in a restricted syntax language with defined semantics.

**ST evaluation**
Assessment of an ST against defined criteria.

**Subject**
An active entity in the TOE that performs operations on objects.

**Target of Evaluation (TOE)**
A set of software, firmware and/or hardware possibly accompanied by guidance.

**TOE evaluation**
Assessment of a TOE against defined criteria.

**TOE resource**
Anything useable or consumable in the TOE.

**TOE Security Functionality (TSF)**
A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs.

**Transfers outside of the TOE**
TSF mediated communication of data to entities not under control of the TSF.

**True RNG (TRNG)**
A device or mechanism for which the output values depend on some unpredictable source (noise source, entropy source) that produces entropy.

**Trusted channel**
A means by which a TSF and a remote trusted IT product can communicate with necessary confidence.

**Trusted path**
A means by which a user and a TSF can communicate with necessary confidence.

**TSF data**
Data created by and for the TOE, that might affect the operation of the TOE.

**TSF Interface (TSFI)**
A means by which external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF.

**User**
See external entity

**User data**
Data created by and for the user, that does not affect the operation of the TSF.