www.GossamerSec.com

# Assurance Activity Report for Zebra Devices on Android 11

Version 0.2
04/26/23

***Prepared by:***
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

***Prepared for:***
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Document: AAR-11337

© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

# Revision History

| Revision | Date | Authors | Summary |
|---|---|---|---|
| Version 0.1 | 04/10/23 | Allison Keenan | Initial draft |
| Version 0.2 | 04/26/23 | Allison Keenan | Addressed ECR comments |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**The TOE Evaluation was Sponsored by**:

**Zebra Technologies Corporation**
3 Overlook Point
Lincolnshire, IL 60069-4302

**Evaluation Personnel**:

- Allison Keenan
- Raymond Smoley

**Common Criteria Versions**:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

**Common Evaluation Methodology Versions**:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

# Table of Contents

1. Introduction ..... 8
1.1 Device Equivalence ..... 8
1.2 CAVP Certificates ..... 8
2. Protection Profile SFR Assurance Activities ..... 10
2.1 Security audit (FAU) ..... 10
2.1.1 Audit Data Generation - per TD0663 (MDFPP32:FAU_GEN.1) ..... 10
2.1.2 Audit Data Generation (Bluetooth) - per TD0707 (BT10:FAU_GEN.1/BT) ..... 12
2.1.3 Audit Data Generation (Wireless LAN) (WLANCEP10:FAU_GEN.1/WLAN) ..... 12
2.1.4 Audit Review (MDFPP32:FAU_SAR.1) ..... 14
2.1.5 Audit Storage Protection (MDFPP32:FAU_STG.1) ..... 14
2.1.6 Prevention of Audit Data Loss (MDFPP32:FAU_STG.4) ..... 16
2.2 Cryptographic support (FCS) ..... 16
2.2.1 Cryptographic Key Generation (MDFPP32:FCS_CKM.1) ..... 16
2.2.2 Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) (WLANCEP10:FCS_CKM.1/WLAN) ..... 21
2.2.3 Cryptographic Key Establishment (MDFPP32:FCS_CKM.2/LOCKED) ..... 23
2.2.4 Cryptographic Key Establishment (MDFPP32:FCS_CKM.2/UNLOCKED) ..... 24
2.2.5 Cryptographic Key Distribution (GTK) (WLANCEP10:FCS_CKM.2/WLAN) ..... 28
2.2.6 Cryptographic Key Support (MDFPP32:FCS_CKM_EXT.1) ..... 29
2.2.7 Cryptographic Key Random Generation (MDFPP32:FCS_CKM_EXT.2) ..... 31
2.2.8 Cryptographic Key Generation (MDFPP32:FCS_CKM_EXT.3) ..... 37
2.2.9 Key Destruction (MDFPP32:FCS_CKM_EXT.4) ..... 43
2.2.10 TSF Wipe (MDFPP32:FCS_CKM_EXT.5) ..... 46
2.2.11 Salt Generation (MDFPP32:FCS_CKM_EXT.6) ..... 48
2.2.12 Bluetooth Key Generation (BT10:FCS_CKM_EXT.8) ..... 49
2.2.13 Cryptographic Operation (MDFPP32:FCS_COP.1/CONDITION) ..... 50
2.2.14 Cryptographic Operation (MDFPP32:FCS_COP.1/ENCRYPT) ..... 51
2.2.15 Cryptographic Operation (MDFPP32:FCS_COP.1/HASH) ..... 57


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.16 Cryptographic Operation (MDFPP32:FCS_COP.1/KEYHMAC) ....58
2.2.17 Cryptographic Operation (MDFPP32:FCS_COP.1/SIGN) ....59
2.2.18 HTTPS Protocol (MDFPP32:FCS_HTTPS_EXT.1) ....61
2.2.19 Initialization Vector Generation (MDFPP32:FCS_IV_EXT.1) ....62
2.2.20 Random Bit Generation - per TD0676 (MDFPP32:FCS_RBG_EXT.1) ....62
2.2.21 Cryptographic Algorithm Services (MDFPP32:FCS_SRV_EXT.1) ....65
2.2.22 Cryptographic Key Storage (MDFPP32:FCS_STG_EXT.1) ....67
2.2.23 Encrypted Cryptographic Key Storage (MDFPP32:FCS_STG_EXT.2) ....70
2.2.24 Integrity of Encrypted Key Storage (MDFPP32:FCS_STG_EXT.3) ....71
2.2.25 TLS Client Protocol (PKGTLS11:FCS_TLSC_EXT.1) ....72
2.2.26 Extensible Authentication Protocol-Transport Layer Security (WLANCEP10:FCS_TLSC_EXT.1/WLAN) ....79
2.2.27 TLS Client Support for Mutual Authentication (PKGTLS11:FCS_TLSC_EXT.2) ....82
2.2.28 TLS Client Protocol (WLANCEP10:FCS_TLSC_EXT.2/WLAN) ....84
2.2.29 TLS Client Support for Renegotiation (PKGTLS11:FCS_TLSC_EXT.4) ....84
2.2.30 TLS Client Support for Supported Groups Extension (PKGTLS11:FCS_TLSC_EXT.5) ....85
2.3 User data protection (FDP) ....86
2.3.1 Access Control for System Services (MDFPP32:FDP_ACF_EXT.1) ....86
2.3.2 Access Control for System Resources (MDFPP32:FDP_ACF_EXT.2) ....92
2.3.3 Protected Data Encryption (MDFPP32:FDP_DAR_EXT.1) ....92
2.3.4 Sensitive Data Encryption (MDFPP32:FDP_DAR_EXT.2) ....94
2.3.5 Subset Information Flow Control - per TD0596 (MDFPP32:FDP_IFC_EXT.1) ....97
2.3.6 User Data Storage (MDFPP32:FDP_STG_EXT.1) ....99
2.3.7 Inter-TSF User Data Transfer Protection (Applications) (MDFPP32:FDP_UPC_EXT.1/APPS) ....100
2.3.8 Inter-TSF User Data Transfer Protection (Bluetooth) (MDFPP32:FDP_UPC_EXT.1/BLUETOOTH) ....102
2.4 Identification and authentication (FIA) ....104
2.4.1 Authentication Failure Handling (MDFPP32:FIA_AFL_EXT.1) ....104
2.4.2 Bluetooth User Authorization (BT10:FIA_BLT_EXT.1) ....108
2.4.3 Bluetooth Mutual Authentication (BT10:FIA_BLT_EXT.2) ....109
2.4.4 Rejection of Duplicate Bluetooth Connections (BT10:FIA_BLT_EXT.3) ....109


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

2.4.5 Secure Simple Pairing (BT10:FIA_BLT_EXT.4) ....110
2.4.6 Trusted Bluetooth Device User Authorization (BT10:FIA_BLT_EXT.6) ....111
2.4.7 Untrusted Bluetooth Device User Authorization (BT10:FIA_BLT_EXT.7) ....112
2.4.8 Port Access Entity Authentication (WLANCEP10:FIA_PAE_EXT.1) ....114
2.4.9 Password Management (MDFPP32:FIA_PMG_EXT.1) ....115
2.4.10 Authentication Throttling (MDFPP32:FIA_TRT_EXT.1) ....116
2.4.11 Multiple Authentication Mechanisms (MDFPP32:FIA_UAU.5) ....116
2.4.12 Re-authenticating (Credential Change) - per TD0706 (MDFPP32:FIA_UAU.6/CREDENTIAL) ....118
2.4.13 Re-authenticating (TSF Lock) - per TD0706 (MDFPP32:FIA_UAU.6/LOCKED) ....119
2.4.14 Protected Authentication Feedback (MDFPP32:FIA_UAU.7) ....121
2.4.15 Authentication for Cryptographic Operation (MDFPP32:FIA_UAU_EXT.1) ....122
2.4.16 Timing of Authentication (MDFPP32:FIA_UAU_EXT.2) ....124
2.4.17 X.509 Validation of Certificates - per TD0603 (MDFPP32:FIA_X509_EXT.1) ....125
2.4.18 X.509 Certificate Validation (WLANCEP10:FIA_X509_EXT.1/WLAN) ....128
2.4.19 X.509 Certificate Authentication - per TD0623 (MDFPP32:FIA_X509_EXT.2) ....130
2.4.20 X.509 Certificate Authentication (EAP-TLS) (WLANCEP10:FIA_X509_EXT.2/WLAN) ....132
2.4.21 Request Validation of Certificates (MDFPP32:FIA_X509_EXT.3) ....134
2.5 Security management (FMT) ....135
2.5.1 Management of Security Functions Behavior - per TD0658 (MDFPP32:FMT_MOF_EXT.1) ....135
2.5.2 Specification of Management Functions - per TD0646 & TD0658 (MDFPP32:FMT_SMF_EXT.1) ....137
2.5.3 Specification of Management Functions (BT10:FMT_SMF_EXT.1/BT) ....155
2.5.4 Specification of Management Functions (Wireless LAN) (WLANCEP10:FMT_SMF_EXT.1/WLAN) ....158
2.5.5 Specification of Remediation Actions (MDFPP32:FMT_SMF_EXT.2) ....158
2.5.6 Current Administrator (MDFPP32:FMT_SMF_EXT.3) ....159
2.6 Protection of the TSF (FPT) ....160
2.6.1 Application Address Space Layout Randomization (MDFPP32:FPT_AEX_EXT.1) ....160
2.6.2 Memory Page Permissions (MDFPP32:FPT_AEX_EXT.2) ....161
2.6.3 Stack Overflow Protection (MDFPP32:FPT_AEX_EXT.3) ....162
2.6.4 Domain Isolation (MDFPP32:FPT_AEX_EXT.4) ....163
2.6.5 Kernel Address Space Layout Randomization (MDFPP32:FPT_AEX_EXT.5) ....165


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

2.6.6 Application Processor Mediation (MDFPP32:FPT_BBD_EXT.1) ....166
2.6.7 JTAG Disablement (MDFPP32:FPT_JTA_EXT.1) ....167
2.6.8 Key Storage (MDFPP32:FPT_KST_EXT.1) ....168
2.6.9 No Key Transmission (MDFPP32:FPT_KST_EXT.2) ....169
2.6.10 No Plaintext Key Export (MDFPP32:FPT_KST_EXT.3) ....171
2.6.11 Self-Test Notification (MDFPP32:FPT_NOT_EXT.1) ....172
2.6.12 Reliable time stamps (MDFPP32:FPT_STM.1) ....173
2.6.13 TSF Cryptographic Functionality Testing (MDFPP32:FPT_TST_EXT.1) ....174
2.6.14 TSF Cryptographic Functionality Testing (Wireless LAN) (WLANCEP10:FPT_TST_EXT.1/WLAN) ....175
2.6.15 TSF Integrity Checking (Post-Kernel) (MDFPP32:FPT_TST_EXT.2/POSTKERNEL) ....176
2.6.16 TSF Integrity Checking (Pre-Kernel) (MDFPP32:FPT_TST_EXT.2/PREKERNEL) ....177
2.6.17 Trusted Update: TSF Version Query (MDFPP32:FPT_TUD_EXT.1) ....178
2.6.18 TSF Update Verification (MDFPP32:FPT_TUD_EXT.2) ....179
2.6.19 Application Signing (MDFPP32:FPT_TUD_EXT.3) ....181
2.7 TOE access (FTA) ....182
2.7.1 TSF- and User-initiated Locked State (MDFPP32:FTA_SSL_EXT.1) ....182
2.7.2 Default TOE Access Banners (MDFPP32:FTA_TAB.1) ....184
2.7.3 Wireless Network Access (WLANCEP10:FTA_WSE_EXT.1) ....185
2.8 Trusted path/channels (FTP) ....186
2.8.1 Bluetooth Encryption (BT10:FTP_BLT_EXT.1) ....186
2.8.2 Persistence of Bluetooth Encryption (BT10:FTP_BLT_EXT.2) ....187
2.8.3 Bluetooth Encryption Parameters (BR/EDR) - per TD0640 (BT10:FTP_BLT_EXT.3/BR) ....188
2.8.4 Bluetooth Encryption Parameters (LE) (BT10:FTP_BLT_EXT.3/LE) ....190
2.8.5 Trusted Channel Communication (MDFPP32:FTP_ITC_EXT.1) ....191
2.8.6 Trusted Channel Communication (Wireless LAN) (WLANCEP10:FTP_ITC_EXT.1/WLAN) ....193
3. Protection Profile SAR Assurance Activities ....196
3.1 Development (ADV) ....196
3.1.1 Basic Functional Specification (ADV_FSP.1) ....196
3.2 Guidance documents (AGD) ....196
3.2.1 Operational User Guidance (AGD_OPE.1) ....196


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

3.2.2 Preparative Procedures (AGD_PRE.1) ..... 197
3.3 Life-cycle support (ALC) ..... 197
3.3.1 Labeling of the TOE (ALC_CMC.1) ..... 197
3.3.2 TOE CM Coverage (ALC_CMS.1) ..... 198
3.3.3 Timely Security Updates (ALC_TSU_EXT.1) ..... 198
3.4 Tests (ATE) ..... 199
3.4.1 Independent Testing - Conformance (ATE_IND.1) ..... 199
3.5 Vulnerability assessment (AVA) ..... 200
3.5.1 Vulnerability Survey (AVA_VAN.1) ..... 200


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

# 1. Introduction

This document presents evaluations results of the Zebra Technologies Corporation Zebra Devices on Android 11 MDFPP32/PKGTLS11/WLANCEP10/BT10 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

## 1.1 Device Equivalence

The TOE encompasses mobile devices that support enterprises and individual users alike and this evaluation includes the following models and versions.

| Product | Model # | CPU | Kernel | Android OS version | Security Patch Level |
|---|---|---|---|---|---|
| 6375 Mobile Handhelds | ET40 | Qualcomm SM6375 | 5.4.147 | Android 11.0 | March 2023 |
| | ET45 | Qualcomm SM6375 | 5.4.147 | Android 11.0 | March 2023 |
| | ET40HC | Qualcomm SM6375 | 5.4.147 | Android 11.0 | March 2023 |
| | ET45HC | Qualcomm SM6375 | 5.4.147 | Android 11.0 | March 2023 |

## 1.2 CAVP Certificates

The TOE performs cryptographic algorithms in accordance with the following NIST standards and has received the following CAVP algorithm certificates.

The TOE's BoringSSL Library (version 7f02881e96e51f1873afcf384d02f782b48967ca, with both Processor Algorithm Accelerators (PAA) and without PAA) provides the following algorithms:

| SFR | Algorithm | NIST Standard | Cert# |
|---|---|---|---|
| FCS_CKM.1 (Key Gen) | RSA IFC Key Generation – 2048/3072 bits | FIPS 186-4, RSA | A3326 |
| | ECDSA ECC Key Generation - P-256/384/521 | FIPS 186-4, ECDSA | A3326 |
| FCS_CKM.2 (Key Establishment) | RSA-based Key Exchange | Vendor affirm 800-56B | N/A |
| | ECC-based Key Exchange - P-256/384/521 | SP 800-56A, CVL KAS ECC | A3326 |
| FCS_COP.1(1) (AES) | AES - 128/256 CBC, GCM, KW | FIPS 197, SP 800-38A/D/F | A3326 |
| FCS_COP.1(2) (Hash) | SHA Hashing - 1/256/384/512 | FIPS 180-4 | A3326 |
| FCS_COP.1(3) (Sign/Verify) | RSA Sign/Verify - 2048/3072 bits | FIPS 186-4, RSA | A3326 |


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

| SFR | Algorithm | NIST Standard | Cert# |
|---|---|---|---|
| | ECDSA Sign/Verify - P-256/384/521 | FIPS 186-4, ECDSA | A3326 |
| FCS_COP.1(4) (Keyed Hash) | HMAC-SHA -1/256/384/512 | FIPS 198-1 & 180-4 | A3326 |
| FCS_RBG_EXT.1 (Random) | DRBG Bit Generation – 256 bits | SP 800-90A (Counter) | A3326 |

Android's LockSettings service (version 77561fc30db9aedc1f50f5b07504aa65b4268b88) provides the TOE'S SP 800-108 key based key derivation function for deriving KEKs.

| SFR | Algorithm | NIST Standard | Cert# |
|---|---|---|---|
| FCS_CKM_EXT.3 | LockSettings service KBKDF 256 bits | SP 800-108 | A1978 |

The TOE's Wi-Fi chipset (BCM43752) provides an AES-CCMP implementation, and the TOE's application processor (Snapdragon 695 [SM6375]) provides additional cryptographic algorithms.

| SFR | Algorithm | NIST Standard | Cert# |
|---|---|---|---|
| FCS_COP.1(1) (AES) (Wi-Fi) | AES 128/256 CCM | FIPS 197, SP 800-38C | 4791 |
| FCS_COP.1(1) (AES) (QTI CEC*) | AES 128/256 CBC | FIPS 197, SP 800-38A | A805 |
| FCS_COP.1(1) (AES) (QTI UFS**) | AES 128/256 XTS | FIPS 197, SP 800-38E | A771 A772 |
| FCS_COP.1(2) (Hash) (QTI CEC) | SHA 1/256 Hashing | FIPS 180-4 | A805 |
| FCS_COP.1(2) (Hash) (DRBG) | SHA 256 Hashing | FIPS 180-4 | A1630 |
| FCS_COP.1(4) (Keyed Hash) (QTI CEC) | HMAC-SHA-1/256 | FIPS 198-1 & 180-4 | A805 |
| FCS_RBG_EXT.1 (Random) (DRBG) | DRBG Bit Generation 256 bits | SP 800-90A (Hash-256) | A1630 |

*QTI CEC – Qualcomm Technologies, Inc. Crypto Engine Core v5.6.0
**QTI UFS - Qualcomm Technologies, Inc. Inline Crypto Engine (UFS) v3.2.0


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

# 2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profile and describes the findings in each case.

The following evidence was used to complete the Assurance Activities:

- Zebra SM6375 Devices on Android 11 Security Target, version 1.4, 4/26/2023
- Administrator Guidance for Zebra Devices (ET4X), version 0.4, 4/04/2023

## 2.1 Security audit (FAU)

### 2.1.1 Audit Data Generation - per TD0663 (MDFPP32:FAU_GEN.1)

#### 2.1.1.1 MDFPP32:FAU_GEN.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.1.2 MDFPP32:FAU_GEN.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.1 of the ST contains links and descriptions of security log and logcat log audit events. This includes all required events and the contents of each audit record.

**Component Guidance Assurance Activities**: The evaluator shall check the administrative 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

type mandated by the PP is described and that the description of the fields contains the information required in FAU_GEN.1.2.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP including those listed in the Management section. The evaluator shall examine the administrative guide and make a determination of which administrative commands are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are security relevant with respect to this PP. The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.

Section 2.6 (Audit Logging) in the Admin Guide indicates that administrators can enable security logging for target devices and these logs can be retrieved via the MDM agent. The security logs can be viewed and exported via the MDM. Relevant logging information can also be captured via security logs and logcat which does not require any additional configuration to be enabled.

Section 8 of the Admin Guide includes details about the audit records which the TOE generates including details encompassing the required content. During testing, the evaluator mapped the entries in the tables in this section to the TOE generated events, showing that the section provides include examples/descriptions of all required audit events. For administrator events, the evaluator ensured an audit record was listed for all security relevant events.

**Component Testing Assurance Activities**: 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.

The evaluator tested the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the provided tables above including all administrative actions. The evaluator collected these audit records while running the security functional tests. When verifying the test results, the evaluator verified that the audit records generated during testing matched the format specified in the administrative guide, and that the fields in each audit record have the proper entries. For each type of audit record, the evaluator found that the


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

TOE correctly generated an audit log matching the vendor specified one. The evaluator collected a sample of each type of audit record and included these samples in the Detailed Test Report for this evaluation.

### 2.1.2 Audit Data Generation (Bluetooth) - per TD0707 & TD0645 (BT10:FAU_GEN.1/BT)

#### 2.1.2.1 BT10:FAU_GEN.1.1/BT

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.2.2 BT10:FAU_GEN.1.2/BT

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.1.3 Audit Data Generation (Wireless LAN) –per TD0194 (WLANCEP10:FAU_GEN.1/WLAN)

#### 2.1.3.1 WLANCEP10:FAU_GEN.1.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: 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.

See MDFPP32:FAU_GEN.1

**Component Testing Assurance Activities**: 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.

See MDFPP32:FAU_GEN.1


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.1.4 Audit Review (MDFPP32:FAU_SAR.1)

#### 2.1.4.1 MDFPP32:FAU_SAR.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.1.4.2 MDFPP32:FAU_SAR.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The evaluation activity for this requirement is performed in conjunction with test for function 32 of FMT_SMF_EXT.1.

See test 32 of MDFPP32:FMT_SMF_EXT.1 for the audit review test

### 2.1.5 Audit Storage Protection (MDFPP32:FAU_STG.1)

#### 2.1.5.1 MDFPP32:FAU_STG.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.1.5.2 MDFPP32:FAU_STG.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.1 of the ST details that for security logs, the TOE stores all audit records in memory, making it only accessible to the logd daemon, and only device owner applications can call the MDM API to retrieve a copy of the logs. Additionally, only new logs can be added. There is no designated method allowing for the deletion or modification of logs already present in memory, but reading the security logs clears the buffer at the time of the read.

The TOE stores Logcat Logs in memory and only allows access by an administrator via an MDM Agent. The TOE prevents deletion of these logs by any method other than USB debugging (and enabling USB Debugging takes the phone out of the evaluated configuration).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

The TOE protects the security log in memory, and only allows access to the logd daemon, which only affords a device owner the MDM API to retrieve a copy of these logs. Because of this, the evaluator had no method to delete or modify the logs present in memory. The TOE also stores Logcat logs in memory buffers; however, it is possible to clear this log as part of debugging access. In CC Mode, the debugging feature must be disabled and therefore, unauthorized users have no access to the logs. Only an authorized administrator can read the audit trail via the TOE's MDM APIs.

Test 1: The evaluator attempted to delete the audit trail as an unauthorized user and confirmed that there were no access controls of any kind to access or delete the logs.

Test 2: The evaluator tested both modification and removal and found no way to attempt modification or removal of the admin protected audit logs while the device was configured with the CC requirement that USB debugging be disabled and disallowed through the MDM APIs


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.1.6 Prevention of Audit Data Loss (MDFPP32:FAU_STG.4)

#### 2.1.6.1 MDFPP32:FAU_STG.4.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.1 of the ST states the security logs and logcat logs are stored in memory in a circular log buffer of 10KB/64KB, respectively. Logcat logs alone have a configurable size, able to be set by an MDM API. There is no limit to the size that the Logcat log buffer can be configured to and it is limited to the size of the system's memory. Once the log is full, it begins overwriting the oldest message and continues overwriting the oldest message with each new auditable event. These logs persist until they are either overwritten or the device is restarted.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.2 Cryptographic support (FCS)

### 2.2.1 Cryptographic Key Generation (MDFPP32:FCS_CKM.1)

#### 2.2.1.1 MDFPP32:FCS_CKM.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Section 6.2 of the ST states that the TOE provides generation of asymmetric keys including

| Algorithm | Key/Curve Sizes | Usage |
|---|---|---|
| RSA, FIPS 186-4 | 2048/3072 | API/Application & Sensitive Data Protection (DAR.2) |
| ECDSA, FIPS 186-4 | P-256/384/521 | API/Application |
| ECDHE keys (not domain parameters) | P-256/384 | TLS KeyEx (WPA2 w/ EAP-TLS & HTTPS) |

The TOE's cryptographic algorithm implementations have received NIST algorithm certificates. The TOE itself does not generate any RSA/ECDSA authentication key pairs for TOE functionality (the user or administrator must load certificates for use with WPA2 with EAP-TLS authentication); however, the TOE provides key generation APIs to mobile applications to allow them to generate RSA/ECDSA key pairs. The TOE generates only ECDH key pairs (as BoringSSL does not support DH/DHE cipher suites) and does not generate domain parameters (curves) for use in TLS Key Exchange.

The TOE will provide a library for application developers to use for Sensitive Data Protection (SDP). This library (class) generates asymmetric RSA keys for use to encrypt and decrypt data that comes to the device while in a locked state. Any data received for a specified application (that opts into SDP via this library), is encrypted using the public key and stored until the device is unlocked. The public key stays in memory no matter the state of the device (locked or unlocked). However, when the device is locked, the private key is evicted from memory and unavailable for use until the device is unlocked. Upon unlock, the private key is re-derived and used to decrypt data received and encrypted while locked.

**Component Guidance Assurance Activities**: 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.

Section 3.1 (Entering into Common Criteria State) in the Admin Guide provides settings and instructions for configuring the TOE into Common Criteria Mode. It states that there is no additional configuration required to ensure key generation, key sizes, hash sizes, and all other cryptographic functions meet NIAP requirements.

Section 10.1 (Cryptographic APIs) in the Admin Guide describes the APIs for both RSA and ECDSA Key Generation. Within these APIs are described the variables that should be used to utilize/select specific key sizes for each key generation scheme.

**Component Testing Assurance Activities**: 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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) )


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

- 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):

a. confirm the private and public keys are 32-byte values

b. confirm the three least significant bits of the first byte of the private

key are zero

c. confirm the most significant bit of the last byte is zero

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

e. calculate the expected public key from the private key and confirm it

matches the supplied public key


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

- 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.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.2.

### 2.2.2 Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) (WLANCEP10:FCS_CKM.1/WLAN)

#### 2.2.2.1 WLANCEP10:FCS_CKM.1.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.2 of the ST states the TOE adheres to IEEE 802.11-2012 for key generation. The TOE's wpa_supplicant provides PRF384 for WPA2 derivation of 128-bit AES Temporal Key (using the HMAC implementation provided by BoringSSL) and employs its BoringSSL AES-256 DRBG when generating random values used in the EAP-TLS and 802.11 4-way handshake. The TOE supports the AES-128 CCMP encryption mode. The TOE has successfully


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

completed certification (including WPA2 Enterprise) and received Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance. The Wi-Fi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.

| Device Name | Model Number | Wi-Fi Alliance Certificate Numbers |
|---|---|---|
| 6375 Mobile Handhelds | ET40 | WFA120159 |
| 6375 Mobile Handhelds | ET45 | WFA119406 |

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

(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.

Test 1 – The access point was configured to have a one hour cryptoperiod. The TOE connected to the access point and after an hour, the TOE and access point renegotiated the keys. The evaluator saw the renegotiation in a packet capture.

Test 2 - The TOE was configured to connect to an access point and a wireless packet capture was started. The TOE was connected and disconnected after more than a minute while a number of broadcast packet were observed using wireshark. The evaluator filtered the capture further to demonstrate the 4-way handshake and encrypted broadcast packets. The evaluator then decrypted the packet capture and demonstrated the PTK and GTK were derived.

See Section 1.2 for identification of CAVP certificates that map to this requirement (AES and HMAC).

### 2.2.3 Cryptographic Key Establishment (MDFPP32:FCS_CKM.2/LOCKED)

#### 2.2.3.1 MDFPP32:FCS_CKM.2.1/LOCKED

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The test for SP800-56A and SP800-56B key establishment schemes is performed in association with FCS_CKM.2.1(1).

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.4 Cryptographic Key Establishment (MDFPP32:FCS_CKM.2/UNLOCKED)

#### 2.2.4.1 MDFPP32:FCS_CKM.2.1/UNLOCKED

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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

Section 6.2 of the ST states that the TOE provides an SDP library for applications that uses a hybrid crypto scheme based on 3072-bit RSA based key establishment. Applications can utilize this library to implement SDP that encrypts incoming data received while the phone is locked in a manner compliant with this requirement.

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

Section 3.4 of the guidance details that no additional configuration is needed for the cryptographic modules in order to be compliant.

**Component Testing Assurance Activities**: 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.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.5 Cryptographic Key Distribution (GTK) (WLANCEP10:FCS_CKM.2/WLAN)

#### 2.2.5.1 WLANCEP10:FCS_CKM.2.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.2 of the ST states the TOE adheres to RFC 3394 and 802.11-2012 standards and unwraps the GTK (sent encrypted with the WPA2 KEK using AES Key Wrap in an EAPOL-Key frame). The TOE, upon receiving an EAPOL frame, will subject the frame to a number of checks (frame length, EAPOL version, frame payload size, EAPOL-Key type, key data length, EAPOL-Key CCMP descriptor version, and replay counter) to ensure a proper EAPOL message and then decrypt the GTK using the KEK, thus ensuring that it does not expose the Group Temporal Key (GTK).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

See the test case for WLANCEP10:FCS_CKM.1.

See Section 1.2 for identification of CAVP certificates that map to this requirement

### 2.2.6 Cryptographic Key Support (MDFPP32:FCS_CKM_EXT.1)

#### 2.2.6.1 MDFPP32:FCS_CKM_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.6.2 MDFPP32:FCS_CKM_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.6.3 MDFPP32:FCS_CKM_EXT.1.3


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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).

Section 6.2 of the ST states the TOE includes a Root Encryption Key (REK) stored in a 256-bit fuse bank within the application processor. The TOE generates the REK/fuse value during manufacturing using its hardware DRBG. The application processor protects the REK by preventing any direct observation of the value and prohibiting any ability to modify or update the value. The application processor loads the fuse value into an internal hardware crypto register and the Trusted Execution Environment (TEE) provides trusted applications the ability to derive KEKs from the REK (using an SP 800-108 KDF to combine the REK with a salt). Additionally, the when the REK is loaded, the


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

fuses for the REK become locked, preventing any further changing or loading of the REK value. The TEE does not allow trusted applications to use the REK for encryption or decryption, only the ability to derive a KEK from the REK. The TOE includes a TEE application that calls into the TEE in order to derive a KEK from the 256-bit REK/fuse value and then only permits use of the derived KEK for encryption and decryption as part of the TOE key hierarchy. More information regarding Trusted Execution Environments may be found here: http://www.globalplatform.org/mediaguidetee.asp.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.2.7 Cryptographic Key Random Generation (MDFPP32:FCS_CKM_EXT.2)

#### 2.2.7.1 MDFPP32:FCS_CKM_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

- 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.

Section 6.2 of the ST states that the TOE utilizes its approved RBGs to generate DEKs. When generating AES keys for itself (for example, the TOE'S sensitive data encryption keys or for the Secure Key Storage), the TOE utilizes the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to generate a 256-bit AES key. The TOE also utilizes that same DRBG when servicing API requests from mobile applications wishing to generate AES keys (either 128 or 256-bit).

In all cases, the TOE generates DEKs using a compliant RBG seeded with sufficient entropy so as to ensure that the generated key cannot be recovered with less work than a full exhaustive search of the key space.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Guidance Assurance Activities**: 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.

Section 6.2 of the TSS explains that the TOE supports Data Encryption Key (DEK) generation using its approved RBGs for use in the TOE'S sensitive data encryption keys or for the Secure Key Storage. The TOE RBGs are capable of generating AES 256-bit DEKs in response to applications and services on the device. The 256-bit length matches the FCS_RBG_EXT.1 requirement.

**Component Testing Assurance Activities**: 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 |


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

| | | |
|---|---|---|
| Length of derived keying material | L | L |
| Length of input values | I length | I 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

-- 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.

o Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

o Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

o 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.

o Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

o Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

o 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.

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.8 Cryptographic Key Generation (MDFPP32:FCS_CKM_EXT.3)

#### 2.2.8.1 MDFPP32:FCS_CKM_EXT.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.8.2 MDFPP32:FCS_CKM_EXT.3.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

- 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.

Section 6.2 of the ST states the TOE takes the user-entered password and conditions/stretches this value before combining the factor with other KEK.

The TOE generates all non-derived KEKs using the RAND_bytes() API call from its BoringSSL AES-256 CTR_DRBG to ensure a full 128/256-bits of strength for asymmetric/symmetric keys, respectively. And the TOE combines KEKs by encrypting one KEK with the other so as to preserve entropy.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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 5 maps the data fields to the notations used in SP 800-108 and SP 800-56C.

Table 5: Notations used in SP 800-108 and SP 800-56C

| Data Fields | Notations | |
|---|---|---|
| | SP 800-108 | SP 800-56C |


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

| | | |
|---|---|---|
| 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 | K1 (key derivation key) | Z (shared secret) |
| Pseudorandom salt values | n/a | s |
| Randomness extraction MAC | n/a | MAC |

Counter Mode Tests:


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

- 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.

--- 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).


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

- 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.

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.9 Key Destruction (MDFPP32:FCS_CKM_EXT.4)

#### 2.2.9.1 MDFPP32:FCS_CKM_EXT.4.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.2.9.2 MDFPP32:FCS_CKM_EXT.4.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.2 of the ST states The TOE clears sensitive cryptographic material (plaintext keys, authentication data, other security parameters) from memory when no longer needed or when transitioning to the device's locked state (in the case of the Sensitive Data Protection keys). Public keys (such as the one used for Sensitive Data Protection) can remain in memory when the phone is locked, but all crypto-related private keys are evicted from memory upon device lock. No plaintext cryptographic material resides in the TOE'S Flash as the TOE encrypts all keys stored in Flash. When performing a full wipe of protected data, the TOE cryptographically erases the protected data by clearing the Data-At-Rest DEK. Because the TOE'S keystore resides within the user data partition, the TOE effectively cryptographically erases those keys when clearing the Data-At-Rest DEK. In turn, the TOE clears the Data-At-Rest DEK and Secure Key Storage SEK through a secure direct overwrite (BLKSECDISCARD ioctl) of the wear-leveled Flash memory containing the key followed by a read-verify.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 – The vendor provided a special build of the TOE. The evaluator used that build to run a series of memory dump tests that dumped the memory on the device. The evaluator took the memory dumps and searched those dumps with a hex search tool to search for known keys. The evaluator was unable to find any of the keys in the dump files.

Test 2 – Not applicable. This test does not apply as the TOE does not store any plaintext keys in Flash and does not overwrite key values in Flash (but instead uses block erases).

Test 3 - Not applicable. This test does not apply as the TOE does not store any plaintext keys in Flash and does not overwrite key values in Flash (but instead uses block erases).

## 2.2.10 TSF Wipe (MDFPP32:FCS_CKM_EXT.5)

### 2.2.10.1 MDFPP32:FCS_CKM_EXT.5.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.2.10.2 MDFPP32:FCS_CKM_EXT.5.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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).

Section 6.2 of the ST states the TOE stores all protected data in encrypted form within the user data partition (either protected data or sensitive data). Upon request, the TOE cryptographically erases the Data-At-Rest DEK protecting the user data partition and the SDP Master KEK protecting sensitive data files in the user data partition, clears those keys from memory, reformats the partition, and then reboots. The TOE's clearing of the keys follows the requirements of FCS_CKM_EXT.4.

**Component Guidance Assurance Activities**: 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.

Section 2.1 Data Protection details that the Zebra devices use file based encryption by default.

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

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

Method 1 for File-based Methods:

Test 1: 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

evaluator shall use a tool provided by the developer to examine the same data location in memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).

Method 2 for Volume-based Methods:

Test 1: 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).

See Section 1.2 for a listing of applicable CAVP certificates.

Test 1 (for File-based Methods) – The evaluator used a debug version of the TOE to access the location of several important files, necessary for the TOE to access and decrypt the File-Based Encryption (FBE) DEKs. The evaluator used a script to record the LBA (Logical Block Address) of the files' data blocks and then dumped the blocks from the raw partition (thus obtaining the ciphertext that resides on the Flash filesystem). The evaluator then wiped the TOE and then accessed the TOE's partitions. The evaluator found that none of the recorded files were present any longer in the TOE's partitions/filesystems.

Test 1 (for Volume-based methods) – Not applicable. The TOE uses file-based encryption and not volume based encryption.

### 2.2.11 Salt Generation (MDFPP32:FCS_CKM_EXT.6)

#### 2.2.11.1 MDFPP32:FCS_CKM_EXT.6.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Section 6.2 of the ST states the TOE generates salt nonces (which are just salt values used in WPA2) using its /dev/urandom.

| Salt value and size | RBG origin | Salt storage location |
|---|---|---|
| User password salt (128-bit) | BoringSSL's AES-256 CTR_DRBG | Flash filesystem |
| TLS client_random (256-bit) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |
| TLS pre_master_secret (384-bit) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |
| TLS ECDHE private value (256, 384) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |
| WPA2 4-way handshake supplicant nonce (SNonce) | BoringSSL's AES-256 CTR_DRBG | N/A (ephemeral) |

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.2.12 Bluetooth Key Generation (BT10:FCS_CKM_EXT.8)

#### 2.2.12.1 BT10:FCS_CKM_EXT.8.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

The ST states the TOE generates public/private ECDH key pairs every blue connection establishment.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test - The evaluator set up each of the TOE devices one at a time to snoop Bluetooth connections and then advertise for Bluetooth. The evaluator used a test device to repeatedly attempt to pair with the TOE device – the attempts were alternately accepted and rejected (cancelled on the TOE device). The test device was unpaired immediately after every successful pairing. After several attempts were concluded, the Bluetooth log was collected from the TOE device. The evaluator found that the public keys were different in every case indicating that the public key pairs change for every pairing attempt.

## 2.2.13 Cryptographic Operation (MDFPP32:FCS_COP.1/CONDITION)

### 2.2.13.1 MDFPP32:FCS_COP.1.1/CONDITION

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

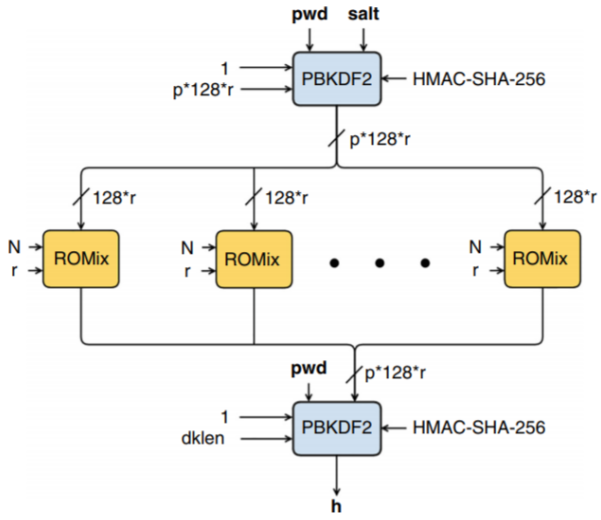
Section 6.2 of the ST states The TOE stretches the user's password to create a password derived key. The TOE stretching function uses a series of steps to increase the memory required for key derivation (thus thwarting GPU-acceleration, off-line brute force, and precomputed dictionary attacks) and ensure proper conditioning and stretching of the user's password.

The TOE conditions the user's password using two iterations of PBKDFv2 w HMAC-SHA-256 in addition to some ROMix operations in an algorithm named scrypt. Scrypt consists of one iteration of PBKDFv2, followed by a series of ROMix operations, and finished with a final iteration of PBKDFv2. The ROMix operations increase the memory


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

required for key derivation, thus thwarting GPU-acceleration (which can greatly decrease the time needed to brute force PBKDFv2 alone). The time needed to derive keying material does not impact or lessen the difficulty faced by an attacker's exhaustive guessing as the combination of the password derived KEK with REK value entirely prevents offline attacks and the TOE's maximum incorrect login attempts.

The following scrypt diagram shows how the password and salt are used with PBKDFv2 and ROMix to fulfil the requirements for password conditioning.




The resulting derived key from this operation is combined with keys chaining to the Application Processor REK and then used to decrypt the FBE DEKs and also to derive the User Keystore Daemon Value.

**Component Guidance Assurance Activities**: There are no guidance evaluation activities for this component.

**Component Testing Assurance Activities**: There are no test evaluation activities for this component. No explicit testing of the formation of the submask from the input password is required.

## 2.2.14 Cryptographic Operation (MDFPP32:FCS_COP.1/ENCRYPT)

### 2.2.14.1 MDFPP32:FCS_COP.1.1/ENCRYPT

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

PT = CT[i-1]

The ciphertext computed in the 1000 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 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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:


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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).


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.15 Cryptographic Operation (MDFPP32:FCS_COP.1/HASH)

#### 2.2.15.1 MDFPP32:FCS_COP.1.1/HASH

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.2 of the ST states the TOE uses byte-wise hashing operations as part of signatures as well as part of HMAC (keyed hashing) operations.

**Component Guidance Assurance Activities**: 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.

Section 3.2 of the guidance details that after the device is in the CC State, no additional configuration is required to ensure key generation, key sizes, hash sizes, and all other cryptographic functions meet NIAP requirements.

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

The TSF may implement either bit-oriented or byte-oriented; both implementations are not required. 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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: 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 ith message is 512 + 99*i, where 1 <= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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 ith message is 512 + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 5: Pseudorandomly Generated Messages Test

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

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.16 Cryptographic Operation (MDFPP32:FCS_COP.1/KEYHMAC)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.2.16.1 MDFPP32:FCS_COP.1.1/KEYHMAC

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.2 of the ST states the TOE uses HMAC as part of the TLS ciphersuites and makes HMAC functionality available to mobile applications. For TLS, the TOE uses HMAC using SHA-1 (with a 160-bit key) to generate a 160-bit MAC, SHA-256 (with a 256-bit key) to generate a 256-bit MAC, SHA-384 (with a 384-bit key) to generate a 384-bit MAC. For mobile applications, the TOE provides all of the previous HMACs as well as SHA-512 (with a 512-bit key) to generate a 512-bit MAC. FIPS 198-1 & 180-4 dictate the block size used, and they specify block sizes/output MAC lengths of 512/160, 512/160, 1024/384, and 1024/512-bits for HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 respectively.

**Component Guidance Assurance Activities**: There are no guidance evaluation activities for this component.

**Component Testing Assurance Activities**: 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.

See Section 1.2 for a listing of applicable CAVP certificates.

Per NIAP Policy Letter #5 CAVP Mapping, 802.11-2012 key generation is addressed by Wi-Fi Alliance certification and HMAC certification. The 802.11ac-2013 protocol is implemented in the wpa_supplicant. The wpa_supplicant uses BoringSSL and its CAVP number is listed in section 1.2 which includes HMAC SHA-384. See WLANEP10:FCS_CKM.1 for the Wi-Fi Alliance certificate number.

### 2.2.17 Cryptographic Operation (MDFPP32:FCS_COP.1/SIGN)

#### 2.2.17.1 MDFPP32:FCS_COP.1.1/SIGN


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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: ECDSA Algorithm Tests

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: RSA Signature Algorithm Tests

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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.18 HTTPS Protocol (MDFPP32:FCS_HTTPS_EXT.1)

#### 2.2.18.1 MDFPP32:FCS_HTTPS_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.18.2 MDFPP32:FCS_HTTPS_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.18.3 MDFPP32:FCS_HTTPS_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

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

Other tests are performed in conjunction with FCS_TLSC_EXT.1.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 – This was tested in FCS_TLSC_EXT.1.1, test case 1.

Test 2 - This was tested in FCS_TLSC_EXT.1.3, test case 1.

## 2.2.19 Initialization Vector Generation (MDFPP32:FCS_IV_EXT.1)

### 2.2.19.1 MDFPP32:FCS_IV_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.2 of the ST states the TOE generates IVs by reading from /dev/urandom for use with all keys. In all cases, the TOE uses /dev/urandom and generates the IVs in compliance with the requirements of table 13 in Appendix F of MDFPP32.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.2.20 Random Bit Generation - per TD0676 (MDFPP32:FCS_RBG_EXT.1)

### 2.2.20.1 MDFPP32:FCS_RBG_EXT.1.1


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.20.2 MDFPP32:FCS_RBG_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.20.3 MDFPP32:FCS_RBG_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: Documentation shall be produced and the evaluator shall perform the activities in accordance with Appendix D - Entropy Documentation And Assessment, the 'Clarification to the Entropy Documentation and Assessment'.

The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development, includes the security functions described in FCS_RBG_EXT.1.3.

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

The TOE provides a number of different RBGs including:

1. A SHA-256 Hash_DRBG provided in the hardware of the Application Processor.
2. An AES-256 CTR_DRBG provided by BoringSSL. This is the only accredited and supported DRBG present in the system and available to independently developed applications. As such, the TOE provides mobile applications access (through an Android Java API) to random data drawn from its AES-256 CTR_DRBG.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

The TOE initializes its AP Hash_DRBG with enough data from its AP hardware noise source to ensure at least 256-bits of entropy. The TOE then uses its AP Hash_DRBG to continuously fill the Linux Kernel Random Number Generator (LKRNG) input pool, and the LKRNG makes entropy easily available to the rest of the system (e.g., the BoringSSL DRBG draws from the LKRNG).

The TOE seeds its BoringSSL AES-256 CTR_DRBG using 384-bits of data from /dev/urandom, thus ensuring at least 256-bits of entropy. The TOE uses its BoringSSL DRBG for all random generation including salts.

The entropy used to seed these DRBG is documented in proprietary entropy documentation. The applicable entropy documentation has been reviewed and submitted to NIAP for approval. Note that the entropy analysis has been accepted by NIAP.

Section 10.1 (Cryptographic APIs) in the Admin Guide contains the cryptographic APIs associated with the security functions described in FCS_RBG_EXT.1.3

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

The evaluator shall perform the following tests.

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

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

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



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

See Section 1.2 for a listing of applicable CAVP certificates.

### 2.2.21 Cryptographic Algorithm Services (MDFPP32:FCS_SRV_EXT.1)

#### 2.2.21.1 MDFPP32:FCS_SRV_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (cryptographic algorithms) described in these requirements.

Section 9 (FDP_DAR_EXT.2 & FCS_CKM.2(2)- Sensitive Data Protection Overview) in the Admin Guide describes the APIs for files that require sensitive data protection. Section 10 (API Specification) of the Admin Guide contains the Cryptographic APIs. There is a specific API for each cryptographic algorithm specified in the FCS_SRV_EXT.1.1 requirement.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: 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.

Test 1 – The developer provided a test program that invoked each of the cryptographic APIs in the documentation. The evaluator mapped the service SFRs with the APIs used in the test program and verified the test program against the documentation to ensure completeness and correctness.

### 2.2.22 Cryptographic Algorithm Services (MDFPP32:FCS_SRV_EXT.2)

#### 2.2.22.1 MDFPP32:FCS_SRV_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: The evaluator shall verify that the API documentation for the secure key storage includes the cryptographic operations by the stored keys.

Section 10 (API Specification) of the Admin Guide contains the Cryptographic APIs. There is a specific API for each cryptographic algorithm specified in the FCS_SRV_EXT.2.1 requirement.

**Component Testing Assurance Activities**: The evaluator shall write, or the developer shall provide access to, an application that requests cryptographic operations of stored keys by the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. The evaluator shall use these APIs to test the functionality of the secure key storage according to the Evaluation Activities in FCS_STG_EXT.1.

Test 1 – The developer provided a test program that invoked each of the cryptographic APIs in the documentation. The evaluator mapped the service SFRs with the APIs used in the test program and verified the test program against the documentation to ensure completeness and correctness.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

## 2.2.23 Cryptographic Key Storage (MDFPP32:FCS_STG_EXT.1)

### 2.2.23.1 MDFPP32:FCS_STG_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.23.2 MDFPP32:FCS_STG_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.23.3 MDFPP32:FCS_STG_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.23.4 MDFPP32:FCS_STG_EXT.1.4

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.23.5 MDFPP32:FCS_STG_EXT.1.5

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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'.

Section 6.2 of the ST states the TOE provides the user, administrator, and mobile applications the ability to import and use asymmetric public and private keys into the TOE'S software-based Secure Key Storage. Certificates are stored in files using UID-based permissions and an API virtualizes the access. Additionally, the user and administrator can request the TOE to destroy the keys stored in the Secure Key Storage. While normally mobile applications cannot use or destroy the keys of another application, applications that share a common application developer (and are thus signed by the same developer key) may do so. In other words, applications with a common developer (and which explicitly declare a shared UUID in their application manifest) may use and destroy each other's keys located within the Secure Key Storage.

The TOE also provides additional protections on keys beyond including key attestation, to allow enterprises and application developers the ability to ensure which keys have been generated securely within the phone.

**Component Guidance Assurance Activities**: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (import, use, and destruction) described in these requirements. The API documentation shall include the method by which applications restrict access to their keys/secrets in order to meet FCS_STG_EXT.1.4'.

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

The table entry "Certificate Management" in Section 3.6 (Common Criteria Related Settings) of the Admin Guide explains how to import and remove certificates. The table entry "Encryption" in Section 3.6 of the Admin Guide explains how to wipe a device. Section 10 (API Specification) of the Admin Guide contains the cryptographic APIs. These APIs address importing, using and deleting private keys.

**Component Testing Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 – The evaluator used a program that tests generated and imported certificates. This test operates on both RSA and ECDSA keys. The evaluator developed a test script that generates several test applications. The applications use RSA keys and ECDSA keys. The applications generate and import cases for both RSA and ECDSA keys and attempt to access keys from apps with the same and different developer signatures. The test demonstrates only applications with the same developer can access/destroy a key.

Test 2 – See test case 1 which also addresses the access to imported and generated keys across applications.

Test 3 - See test case 1 which also addresses the destruction of imported and generated keys across applications.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.2.24 Encrypted Cryptographic Key Storage (MDFPP32:FCS_STG_EXT.2)

#### 2.2.24.1 MDFPP32:FCS_STG_EXT.2.1

**TSS Assurance Activities**: 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.

Section 6.2 of the ST states the TOE employs a key hierarchy that protects all DEKs and KEKs by encryption with either the REK or by the REK and password derived KEK.

The TOE encrypts Long-term Trusted channel Key Material (LTTCKM, i.e., Bluetooth and WiFi keys) values using AES-256 GCM encryption and stores the encrypted values within their respective configuration files.

All keys are 256-bits in size. All keys are generated using the TOE'S BoringSSL AES-256 CTR_DRBG or application processor SHA-256 Hash_DRBG. By utilizing only 256-bit KEKs, the TOE ensures that all keys are encrypted by an equal or larger sized key.

In the case of Wi-Fi, the TOE utilizes the 802.11-2012 KCK and KEK keys to unwrap (decrypt) the WPA2 Group Temporal Key received from the access point. The TOE protects persistent Wi-Fi keys (user certificates and private keys) by storing them in the Android Key Store.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.2.24.2 MDFPP32:FCS_STG_EXT.2.2

**TSS Assurance Activities**: 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.

See MDFPP:FCS_STG_EXT.2.1. Additionally, Section 6.2 (FCS_STG_EXT.3) in the ST states that the TOE protects the integrity of all DEKs and KEKs (other than LTTCKM keys) stored in Flash by using authenticated encryption/decryption methods (CCM, GCM).

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.2.25 Integrity of Encrypted Key Storage (MDFPP32:FCS_STG_EXT.3)

#### 2.2.25.1 MDFPP32:FCS_STG_EXT.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.2.25.2 MDFPP32:FCS_STG_EXT.3.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Section 6.2 of the ST states the TOE protects the integrity of all DEKs and KEKs (including LTTCKM keys) stored in Flash by using authenticated encryption/decryption methods (CCM, GCM).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.2.26 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

#### 2.2.26.1 PKGTLS11:FCS_TLS_EXT.1.1

**TSS Assurance Activities**: None Defined

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

Section 5 of the ST contains TLS sections are consistent with selections in the dependent components.

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.2.27 TLS Client Protocol (PKGTLS11:FCS_TLSC_EXT.1)

#### 2.2.27.1 PKGTLS11:FCS_TLSC_EXT.1.1

**TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

The ST, section 6.2, references the SFR for the list of ciphersuites. The ST states that no configuration is necessary for selecting the ciphersuites. The TOE inherently (without requiring any configuration) supports the supported ciphersuites and evaluated elliptic curves; neither the user nor the administrator need configure anything in order for the TOE to support these curves.

**Guidance Assurance Activities**: 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.

No configuration is necessary to use the claimed ciphersuites.

**Testing Assurance Activities**: 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.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

These tests are repeated for HTTPS and TLS.

Test 1: The evaluator established a TLS session using the interfaces stated above for each of the claimed ciphersuites in turn. The evaluator used a network sniffer to capture the TLS session negotiation and observed that the expected TLS cipher is negotiated.

Test 2: The evaluator established a TLS session using the interfaces stated above. The evaluator configured the server to send a certificate with the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session was successfully negotiated. The evaluator reconfigured the test server to retry the TLS session using a certificate that is missing the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is not successfully negotiated.

Test 3: The evaluator established a TLS session using the interfaces stated above. A modified test server negotiates an ECDSA ciphersuite, but returns an RSA certificate. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is not negotiated successfully.

Test 4: The evaluator configured a test server to accept only the TLS_NULL_WITH_NUL_NULL ciphersuite. The evaluator then attempted to establish a TLS session using the interfaces stated above to that test server. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is not successfully negotiated.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 5: The evaluator obtained a packet captures of the TLS session negotiation using the interfaces stated above and a test server with Mutual Authentication configured on the test server. The evaluator made connection attempts from the client to the test server. The server implementation of the TLS protocol was modified as stated in the 7 scenarios described by the Assurance Activity. The evaluator inspected each packet captures to ensure that the connections are rejected for each scenario.

#### 2.2.27.2 PKGTLS11:FCS_TLSC_EXT.1.2

**TSS Assurance Activities**: 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.

The ST, section 6.2, states the TOE supports Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers. The TOE supports client (mutual) authentication. The TOE in its evaluated configuration and, by design, supports elliptic curves for TLS (P-256 and P-384) and has a fixed set of supported curves (thus the admin cannot and need not configure any curves).

No additional configuration is needed to restrict allow the device to use the supported cipher suites, as only the claimed cipher suites are supported in the aforementioned library as each of the aforementioned ciphersuites are supported on the TOE by default or through the use of the TLS library.

While the TOE supports the use of wildcards in X.509 reference identifiers (SAN only), the TOE does not support certificate pinning.

**Guidance Assurance Activities**: 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.

Section 3.1 (Entering into Common Criteria State) in the Admin Guide provides instructions for the options that need to be set in the evaluated configuration. No additional configuration is required to ensure key generation, key sizes, hash sizes, and all other cryptographic functions meet NIAP requirements.

Section 10.3 (Certificate Validation, TLS, HTTPS, and Bluetooth) in the Admin Guide describes the TLS versions and ciphersuites that the TOE supports consistent with those identified in the requirements. It states, that by default, the device is restricted to only support TLS ciphersuites that are RFC compliant and claimed under the MDFPP. As such, no configuration is needed to restrict or allow ciphersuites to be compliant.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Testing Assurance Activities**: 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. (TD0499 applied)

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 5.1: [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 5.2: [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 5.3: [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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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 5.4: [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.

These tests are repeated for HTTPS and TLS.

Test 1: The evaluator attempted a connection with a valid SAN and the connection was accepted. The evaluator attempted to connect to a server with no SAN. The connection was rejected.

Test 2: The evaluator attempted to connect to a server with a server certificate that contains a bad SAN extension. The connection was rejected.

Test 3: The evaluator attempted to connect to a server with a server certificate that does not contain the SAN extension. The connection was rejected as expected because the TOE requires SAN always.

Test 4: The evaluator attempted to connect to a server with a server certificate that does contain an identifier in the SAN that matches. The connection succeeded.

Test 5: The evaluator attempted to connect to a server with a server certificate that contains various wildcard combinations. The connections were rejected.

Test 6: The TOE does not support the optional URI or Service Name used as reference identifiers.

Test 7: certificate pinning is not supported

#### 2.2.27.3 PKGTLS11:FCS_TLSC_EXT.1.3

**TSS Assurance Activities**: 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.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

The selection for override was not made.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: The evaluator shall demonstrate that using an invalid certificate 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.

(TD0513 applied)

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.

These tests are repeated for HTTPS and TLS.

Test 1: The evaluator configured server using a certificate with a valid certification path terminating in a certificate which was not configured in the TOE as trusted. The evaluator observed that the TOE rejected the certificate. The evaluator then loaded the trusted CA certificate(s) needed to validate the server's certificate, and demonstrated that the connection succeeded. The evaluator then deleted the CA certificate that was loaded in the previous test part, and showed that the connection again failed.

Test 2: This test has been performed in FIA_X509_EXT.1 test case 3.

Test 3: This test has been performed in FIA_X509_EXT.1 test case 2.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 4: This test has been performed in FCS_TLSC_EXT.1.2.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.2.28 Extensible Authentication Protocol-Transport Layer Security (WLANCEP10:FCS_TLSC_EXT.1/WLAN)

### 2.2.28.1 WLANCEP10:FCS_TLSC_EXT.1.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.28.2 WLANCEP10:FCS_TLSC_EXT.1.2/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.28.3 WLANCEP10:FCS_TLSC_EXT.1.3/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.28.4 WLANCEP10:FCS_TLSC_EXT.1.4/WLAN


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.28.5 WLANCEP10:FCS_TLSC_EXT.1.5/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.2.28.6 WLANCEP10:FCS_TLSC_EXT.1.6/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component. The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

Section 6.2 (FCS_TLSC_EXT.1/2/WLAN) in the ST states that the TOE supports TLS versions 1.0, 1.1 and 1.2 and also supports the selected ciphersuites utilizing SHA-1, SHA-256, and SHA-384 listed in Section 5 for FCS_TLSC_EXT.1/WLAN. The evaluator also verified in the operational guidance that to additional configuration is needed.

**Component Guidance Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Section 3.1 (Entering into Common Criteria State) in the Admin Guide provides the instructions for settings that must be enabled or disabled in the evaluated configuration. No additional configuration is required to ensure key generation, key sizes, hash sizes, and all other cryptographic functions meet NIAP requirements.

The table entry “Certificate Management” in Section 3.6 (Common Criteria Related Settings) of the Admin Guide explains how to import and remove certificates. The table entry “Wi-Fi Settings” in Section 3.6 explains how to configure certificates for Wi-Fi access.

Section 10.3 (Certificate Validation, TLS, HTTPS, and Bluetooth) in the Admin Guide describes the TLS versions and ciphersuites that the TOE supports consistent with those identified in the requirements. It states, that by default, the device is restricted to only support TLS ciphersuites that are RFC compliant and claimed under the MDFPP. As such, no configuration is needed to restrict or allow ciphersuites to be compliant.

**Component Testing Assurance Activities**: 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:



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

o 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.

o 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.

o 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.

o [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. (TD0492 applied)

o 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.

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

Test 1 – The evaluator established a connection with each ciphersuite specified in the ST. The evaluator verified the correctness of the ciphersuite using the server logging options (the evaluator has a wireshark capture as well).

Test 2 –The evaluator created a server certificate with a valid chain and attempted to establish a connection. The connection was accepted. The evaluator then created a server certificate without the Server Authentication purpose in the extendedKeyUsage field and verified the connection was rejected.

Test 3 –The evaluator attempted to establish a connection using an ECDSA certificate while using an RSA ciphersuite. The connection was rejected.

Test 4 - The evaluator then attempted to establish a connection using the TLS_NULL_WITH_NULL_NULL ciphersuite. The connection was rejected.

Test 5 –The evaluator attempted to establish a connection using created packets that modified each of the required options. In all cases the connection was denied as expected.

### 2.2.29 TLS Client Support for Mutual Authentication (PKGTLS11:FCS_TLSC_EXT.2)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.2.29.1 PKGTLS11:FCS_TLSC_EXT.2.1

**TSS Assurance Activities**: 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.

Section 6.2 of the ST states the TOE supports mutual (client) authentication. The ST says no other configuration beyond a certificate is required to support the ST claims.

**Guidance Assurance Activities**: 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.

Section 3.1 (Entering into Common Criteria State) in the Admin Guide provides the instructions for settings that must be enabled or disabled in the evaluated configuration. No additional configuration is required to ensure key generation, key sizes, hash sizes, and all other cryptographic functions meet NIAP requirements.

**Testing Assurance Activities**: 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.

This test is repeated for TLS and HTTPS.

Test 1: The evaluator established a TLS session using a test server that was not configured for mutual authentication. The evaluator observed that the TLS connection was successful and the TOE did not send a certificate or a certificate verify message.

Test 2: The evaluator established a TLS session using a test server that was configured for mutual authentication. The evaluator observed that the TLS connection was successful and the TOE did send both a certificate and a certificate verify message


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.2.30 TLS Client Protocol (WLANCEP10:FCS_TLSC_EXT.2/WLAN)

### 2.2.30.1 WLANCEP10:FCS_TLSC_EXT.2.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall verify that the TSS describes the supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured.

Section 6.2 of the ST states that the TOE in its evaluated configuration and by design, supports only evaluated elliptic curves for TLS (P-256 and P-384) and has a fixed set of supported curves (thus there is no admin configuration required).

**Component Guidance Assurance Activities**: If the TSS indicates that the supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that the operational guidance includes instructions on configuration of the supported Elliptic Curves Extension.

Section 6.2 (FCS_TLSC_EXT.1/2) of the ST states that the TOE in its evaluated configuration and by design, supports only evaluated elliptic curves for TLS (P-256 and P-384) and has a fixed set of supported curves (thus there is no admin configuration required).

**Component Testing Assurance Activities**: The evaluator shall perform the following test:

Test 1: The evaluator shall configure a server to perform ECDHE key exchange using each of the TOE's supported curves and shall verify that the TOE successfully connects to the server. (TD0244 applied)

Test 1- The evaluator configured the test server for each curve claimed by the ST. The client was able to establish a connection with each claimed curve.

## 2.2.31 TLS Client Support for Renegotiation (PKGTLS11:FCS_TLSC_EXT.4)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.2.31.1 PKGTLS11:FCS_TLSC_EXT.4.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: 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.

Section 6.2 of the ST states the TOE includes the ‘renegotiation_info’ TLS extension in its TLS client hello message.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.2.32 TLS Client Support for Supported Groups Extension (PKGTLS11:FCS_TLSC_EXT.5)

#### 2.2.32.1 PKGTLS11:FCS_TLSC_EXT.5.1

**TSS Assurance Activities**: The evaluator shall verify that TSS describes the Supported Groups Extension.

Section 6.2 of the ST states the TOE supports the secp256r1 and secp384r1 groups in its TLS client hello message ‘supported_groups’ extension.

**Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Testing Assurance Activities**: 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.

Test 1- The evaluator configured the test server for each curve claimed by the ST. The client was able to establish a connection with each claimed curve.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.3 User data protection (FDP)

### 2.3.1 Access Control for System Services (MDFPP32:FDP_ACF_EXT.1)

#### 2.3.1.1 MDFPP32:FDP_ACF_EXT.1.1

**TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Section 6.3 of the ST states the TOE provides the following categories of system services to applications.

1. Normal - A lower-risk permission that gives an application access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval (though the user always has the option to review these permissions before installing).
2. Dangerous - A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system cannot automatically grant it to the requesting application. For example, any dangerous permissions requested by an application will be displayed to the user and require confirmation before proceeding or some other approach can be taken to avoid the user automatically allowing the use of such facilities.
3. Signature - A permission that the system is to grant only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
4. SignatureOrSystem - A permission that the system is to grant only to packages in the Android system image or that are signed with the same certificates. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission is used for certain special situations where multiple vendors have applications built in to a system image which need to share specific features explicitly because they are being built together.

An example of a normal permission is the ability to check whether the device is connected to a network: android.permission.ACCESS_NETWORK_STATE. This permission allows an application to query whether the phone currently has a network connection (whether that be through Wi-Fi, USB tethering, cellular, etc.), and an application that does not request (or declare) this permission have its query rejected (and would not learn the device's networking state).

An example of a dangerous privilege would be access to location services to determine the location of the mobile device: android.permission.ACCESS_FINE_LOCATION. The TOE controls access to Dangerous permissions during the running of the application. The TOE prompts the user to review the application's requested permissions (by displaying a description of each permission group, into which individual permissions map, that an application requested access to). If the user approves, then the application is allowed to continue running. If the user disapproves, the devices continues to run, but cannot use the services protected by the denied permissions. Thereafter, the mobile device grants that application during execution access to the set of permissions declared in its Manifest file.

An example of a signature permission is the android.permission.BIND_VPN_SERVICE that an application must declare in order to utilize the VpnService APIs of the device. Because the permission is a Signature permission, the mobile device only grants this permission to an application (2nd installed app) that requests this permission and that has been signed with the same developer key used to sign the application (1st installed app) declaring the permission (in the case of the example, the Android Framework itself).

An example of a signatureOrSystem permission is the android.permission.CONTROL_LOCATION_UPDATES, which allows the system to allow or disallow the cellular radio to update the device's location. The device grants this


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

permission to requesting applications that either have been signed with the same developer key used to sign the Android application declaring the permissions or that reside in the “system” directory within Android (which for Android 4.4 and above, are applications residing in the /system/priv-app/ directory on the read-only system partition). Put another way, the device grants systemOrSignature permissions by Signature or by virtue of the requesting application being part of the “system image”.

Additionally, Android includes the following flags that layer atop the base categories.

1. privileged - this permission can also be granted to any applications installed as privileged apps on the system image. Please avoid using this option, as the signature protection level should be sufficient for most needs and works regardless of exactly where applications are installed. This permission flag is used for certain special situations where multiple vendors have applications built in to a system image which need to share specific features explicitly because they are being built together.
2. system - Old synonym for 'privileged'.
3. development - this permission can also (optionally) be granted to development applications (e.g., to allow additional location reporting during beta testing).
4. appop - this permission is closely associated with an app op for controlling access.
5. pre23 - this permission can be automatically granted to apps that target API levels below API level 23 (Marshmallow/6.0).
6. installer - this permission can be automatically granted to system apps that install packages.
7. verifier - this permission can be automatically granted to system apps that verify packages.
8. preinstalled - this permission can be automatically granted to any application pre-installed on the system image (not just privileged apps) (the TOE does not prompt the user to approve the permission).

For older applications (those targeting Android’s pre-23 API level, i.e., API level 22 [lollipop] and below), the TOE will prompt a user at the time of application installation whether they agree to grant the application access to the requested services. Thereafter (each time the application is run), the TOE will grant the application access to the services specified during install.

For newer applications (those targeting API level 23 or later), the TOE grants individual permissions at application run-time by prompting the user for confirmation of each permissions category requested by the application (and only granting the permission if the user chooses to grant it).

The Android 11.0 (Level 30) API (details found here https://developer.android.com/reference/packages) provides services to mobile applications.

While Android provides a large number of individual permissions, they are generally grouped into categories or features that provide similar functionality. The table below shows system service categories.

Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet). Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

| System Services | Description |
|---|---|
| Camera | Imaging capture (video and photos) |
| Microphone | Audio capture |


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

| System Services | Description |
|---|---|
| Location | Location (GPS, GLONASS, etc) |
| Data Storage Protection | App data, App cache |
| Contacts/Address Book | Contacts and Address book |
| Calendar | Calendar |
| File / Storage Access | Access to internal and external storage data |
| Device Identifier information | Unique device identifying information (IMEI, Device/CPU ID, etc.) |
| Text Messages | SMS and RMS messaging |
| Telephony | Calls, call logs, voicemail |
| Cellular (Mobile) data | Cellular networks |
| Bluetooth | Bluetooth devices, paired devices, current connections. |
| NFC | Near Field Communications |
| Network Access | Current network state and network details |
| VPN | Virtual Private Networks |
| Body Sensors | Accelerometer, Bluetooth monitors |
| Fingerprint / Biometrics | Fingerprint, Iris, other authentication biometrics |
| Credential Access | Password metrics, quality, complexity |

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: 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.

Test 1: The access control in the TOE is based on Android permissions. There are no system services to which no applications are allowed access.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 2: In order to test the permissions, the vendor provided two versions of the same test application, one signed with a trusted platform certificate, and the second signed with an untrusted developer certificate. The evaluator ran each application attempting to access the APIs protected by those permissions. The evaluator verified that the developer signed application output showed that the TOE correctly rejected access to the privileged permissions and protected APIs. The evaluator verified that the platform signed application output showed that the TOE correctly granted access to the privileged permissions and protected APIs.

Test 3 – The evaluator used an application that declared all Dangerous level permissions in the manifest file. The evaluator executed the application and confirmed that the TOE behaved correctly. Where the logs noted that the user had not explicitly granted the Dangerous permission, the test application's attempt to call the API failed. The evaluator ran this test again and simulated the user explicitly granting authorization and confirmed that the TOE behaved correctly. The logs showed that because user authorization was explicitly granted for all Dangerous permissions, the test application's attempt to call the APIs succeeded.

Test 4 –The evaluator used an application that has all Dangerous and Normal level permissions declared in the application manifest. No user approval was provided for Dangerous level permissions. The TOE granted it access to the Normal APIs while correctly rejecting access to the Dangerous ones. The evaluator used another test application that declared no permissions in its manifest file. This application was not granted any Normal permission (nor any other permissions) and could not access any of the protected APIs. Through testing the permission access control, the evaluator found that the protected APIs could only be accessed when the associated permissions were granted by the TOE.

#### 2.3.1.2 MDFPP32:FDP_ACF_EXT.1.2

**TSS Assurance Activities**: 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.

Applications with a common developer have the ability to allow sharing of data between their applications. A common application developer can sign their generated APK with a common certificate or key and set the permissions of their application to allow data sharing. When the different applications' signatures match and the proper permissions are enabled, information can then be shared as needed.

The TOE supports Enterprise profiles to provide additional separation between application and application data belonging to the Enterprise profile. Applications installed into the Enterprise versus Personal profiles cannot access each other's secure data, applications, and can have separate device administrators/managers. This functionality is built into the device by default and does not require an application download. The Enterprise administrative app (an MDM agent application installed into the Enterprise Profile) may enable cross-profile contacts search, in which case, the device owner can search the address book of the enterprise profile. Please see the Admin Guide for


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

additional details regarding how to set up and use Enterprise profiles. Ultimately, the enterprise profile is under control of the personal profile. The personal profile can decide to remove the enterprise profile, thus deleting all information and applications stored within the enterprise profile. However, despite the "control" of the personal profile, the personal profile cannot dictate the enterprise profile to share applications or data with the personal profile; the enterprise profile MDM must allow for sharing of contacts before any information can be shared.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: 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.

Test 1 – The TSF provides two separate access control policies. The first access control policy is specified on a per-app basis. This allows applications to have their own individualized sandbox for storing all application data. The evaluator used a test application to write a known string to the application's data directory and then used a second application to attempt to access this file. The TOE prohibited access and the application received a "Permission denied" error from the TOE. Next the evaluator used a third application which had a (shared) UID matching that of the first test application and verified that this application could access the data in the application's data directory created by the first test application.

The second access control policy is specified on the application group level. The evaluator configured a Work Profile (a group of managed enterprise applications) on the TOE. The evaluator used a test application installed in both the personal and Work profiles of the TOE. The evaluator then created and read some known data in the personal profile and attempted to access it from the work profile and found that permission was denied. The evaluator then created and read some known data in the work profile and attempted to access it from the personal profile and found that permission was denied.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: None Defined

### 2.3.2 Access Control for System Resources (MDFPP32:FDP_ACF_EXT.2)

#### 2.3.2.1 MDFPP32:FDP_ACF_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: For each selected resource, the evaluator shall cause data to be placed into the Enterprise group's instance of that shared resource. The evaluator shall install an application into the Personal group that attempts to access the shared resource information and verify that it cannot access the information.

Test 1 – The evaluator used the address book, calendar, and keychain for each application group (Personal and Work). For calendar and keychain, the TOE does not provide any mechanism to share data between different application groups and the evaluator confirmed that there was no data visible between the Personal and Work profiles. For contacts, the evaluator first confirmed that the contacts in each profile were not visible to the other profile. The evaluator then enabled contacts sharing between profiles and confirmed that the Personal profile could now view contacts in the Work profile. Note this setting only allows sharing of the Work Profile information with the Personal profiles (not the other way around).

### 2.3.3 Protected Data Encryption (MDFPP32:FDP_DAR_EXT.1)

#### 2.3.3.1 MDFPP32:FDP_DAR_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.3.3.2 MDFPP32:FDP_DAR_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.3 of the ST states the TOE provides Data-At-Rest AES-256 XTS hardware encryption for all data stored on the TOE in the user data partition (which includes both user data and TSF data). The TOE also has TSF data relating to key storage for TSF keys not stored in the system's Android Key Store. The TOE separately encrypts those TSF keys and data. Additionally, the TOE includes read-only filesystems (system and vendor) in which the TOE'S system executables, libraries, and their configuration data reside.

For its Data-At-Rest encryption of the data partition on the internal Flash (where the TOE stores all user data and all application data), the TOE uses an AES-256 bit DEK with XTS feedback mode to encrypt each file in the data partition using dedicated application processor hardware. The TOE uses File Based Encryption (FBE) to encrypt files either using Device Encryption (DE) or Credential Encryption (CE), where the latter files the TOE combines a key chaining to the REK with the user's password to derive the CE encryption keys.

**Component Guidance Assurance Activities**: 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.

DAR requires no additional configuration.

**Component Testing Assurance Activities**: 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.

Test 1 – The TOE always applies encryption to the userdata partition, thus one need not enable encryption (as it's not possible to disable encryption). The evaluator browsed to the encryption settings under the Settings application to verify that the phone reported that it was encrypted. To ensure that this encryption is tied to the


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

user's password, the evaluator made sure that a screenlock password was configured on the TOE device. This, in return requires the user to enter the password prior to booting up the phone.

The evaluator then created application data in the default application data directory which is not accessible by standard users. The evaluator also used a userdebug version of the TOE to start an ADB shell and analyze the created file. The evaluator made sure the file, its parent directory, and its contents were freely observable while the device was decrypted. The evaluator also used it to print out the inode index number.

The evaluator then rebooted the phone, allowed it to fully boot to the initial unlock screen, but did not enter a password (allowing the phone to remain locked and encrypted). The evaluator then started a root shell, and navigated to the same directory, listing the directories and their contents. The evaluator found that the directory and file names were encrypted, and unrecognizable. The evaluator tried to search for part of the string to ensure it wasn't just that directory names were encrypted, however the evaluator found that the garbled names did not relate to correct file descriptors since everything was still encrypted. The evaluator also attempted to see if any files were listed under the same inode, however none were found.

The evaluator used a binary search tool to search the userdata partition block device (which represents the logical memory) for the string the evaluator wrote to the previously created file. The tool reported no matches anywhere in the userdata partition, confirming that the all data within the userdata partition are encrypted.

Finally, the evaluator then entered the password, the phone unlocked, and the ADB shell was severed. After starting another ADB shell, the evaluator navigated once more to the application's data directory. The evaluator found that having logged in (and provided the password), the test file was again accessible.

### 2.3.4 Sensitive Data Encryption (MDFPP32:FDP_DAR_EXT.2)

#### 2.3.4.1 MDFPP32:FDP_DAR_EXT.2.1

**TSS Assurance Activities**: 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.).


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Section 6.3 of the ST states the TOE provides a Java library for Sensitive Data Protection (SDP) that application developers can use to opt-in for sensitive data protection. When developers opt-in for SDP, all data that is received on the device destined for that application is treated as sensitive. This library calls into the TOE to generate an RSA key that acts as a master KEK for the SDP encryption process. When an application that has opted-in for SDP receives incoming data while the device is locked, an AES symmetric DEK is generated to encrypt that data. The public key from the master RSA KEK above is then used to encrypt the AES DEK. Once the device is unlocked, the RSA KEK private key is re-derived and can be used to decrypt the AES DEK for each piece of information that was stored while the device was locked. The TOE then takes that decrypted data and re-encrypts it following FDP_DAR_EXT.1.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: 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.

Test 1 – The evaluator installed an application provided by the vendor that created a sensitive data repository. The evaluator then used that application to create sensitive data and read the created data.

#### 2.3.4.2 MDFPP32:FDP_DAR_EXT.2.2

**TSS Assurance Activities**: 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.

See MDFPP:FDP_DAR_EXT.2.1.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: 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.

Test 1 – The evaluator performed this test in conjunction with FCS_CKM_EXT.4. The evaluator performed tests in FCS_CKM_EXT.4 Test 1 for all key material no longer needed while in the locked state and ensured that keys for the asymmetric scheme are addressed in the tests performed when transitioning to the locked state. In this



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

referenced test, the evaluator used the application from FDP_DAR_EXT.2.1 Test 1 to encrypt, decrypt, and dump key values that can be used for testing.

#### 2.3.4.3 MDFPP32:FDP_DAR_EXT.2.3

**TSS Assurance Activities**: 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 passwordderived 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.

Section 6.2 (FCS_STG_EXT.2) in the ST states that the TOE employs a key hierarchy that protects all DEKs and KEKs by encryption with either the REK or by the REK and password derived KEK.

See also MDFPP:FDP_DAR_EXT.2.1 for a description of keys for sensitive data protection.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.4.4 MDFPP32:FDP_DAR_EXT.2.4

**TSS Assurance Activities**: 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.

Section 6.3 of the ST states the TOE provides a Java library for Sensitive Data Protection (SDP) that application developers can use to opt-in for sensitive data protection. When developers opt-in for SDP, all data that is received



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

on the device destined for that application is treated as sensitive. This library calls into the TOE to generate an RSA key that acts as a master KEK for the SDP encryption process. When an application that has opted-in for SDP receives incoming data while the device is locked, an AES symmetric DEK is generated to encrypt that data. The public key from the master RSA KEK above is then used to encrypt the AES DEK. Once the device is unlocked, the RSA KEK private key is re-derived and can be used to decrypt the AES DEK for each piece of information that was stored while the device was locked. The TOE then takes that decrypted data and re-encrypts it following FDP_DAR_EXT.1.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.3.5 Subset Information Flow Control - per TD0596 (MDFPP32:FDP_IFC_EXT.1)

#### 2.3.5.1 MDFPP32:FDP_IFC_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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).

Section 6.3 of the ST states the TOE will route all traffic other than traffic necessary to establish the VPN connection to the VPN gateway (when the gateway's configuration specifies so). The TOE includes an interceptor kernel module


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

that controls inbound and output packets. When a VPN is active, the interceptor will route all incoming packets to the VPN and conversely route all outbound packets to the VPN before they are output.

Note that when the TOE tries to connect to a Wi-Fi network, it performs a standard captive portal check which sends traffic that bypasses the full tunnel VPN configuration in order to detect whether the Wi-Fi network restricts Internet access until one has authenticated or agreed to usage terms through a captive portal. If the administrator wishes to deactivate the captive portal check (in order to prevent the plaintext traffic), they may do this by following the instructions in the Admin Guide.

The only exception to all traffic being routed to the VPN is in the instance of ICMP echo requests. The TOE uses ICMP echo responses on the local subnet to facilitate network troubleshooting and categorizes it as a part of ARP. As such, if an ICMP echo request is issued on the subnet the TOE is part of, it will respond with an ICMP echo response, but no other instances of traffic will be routed outside of the VPN.

**Component Guidance Assurance Activities**: 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.

Section 6.0 of the guidance details VPN configuration.

Always-on VPN—The VPN can be configured so that apps don't have access to the network until a VPN connection is established, which prevents apps from sending data across other networks.

Always-on VPN supports VPN clients that implement VpnService. The system automatically starts that VPN after the device boots. Device owners and profile owners can direct work apps to always connect through a specified VPN. Additionally, users can manually set Always-on VPN clients that implement VpnService methods using **Settings > More > VPN**. Always-on VPN can also be enabled manually from the **Settings** menu

**Component Testing Assurance Activities**: Test 1: 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



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 -

Step 1 – The evaluator connected the TOE device via Wi-Fi to an access point and then sent pings to and from the TOE from multiple addresses and also attempted to browse several websites.

Step 2 - The evaluator then enabled a VPN connection using a PSK and performed the same actions as in step 1.

Step 3 & 4 - The evaluator examined the packet captures from Step 1 and Step 2 and confirmed that the VPN network was accessible (and PROTECTED) only when the VPN was enabled. Other addresses (ping and web page access) were possible in plain text only when the VPN was not enabled. When the VPN connection was established, traffic to and from the TOE was blocked or discarded depending on whether a suitable route through the VPN was available

### 2.3.6 User Data Storage (MDFPP32:FDP_STG_EXT.1)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.3.6.1 MDFPP32:FDP_STG_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.3 of the ST states that the TOE'S Trusted Anchor Database consists of the built-in certs and any additional user or admin/MDM loaded certificates. The built-in certs are individually stored in the device's read-only system image in the /system/etc/security/cacerts directory, and the user can individually disable certs through Android's user interface [Settings->Security-> Trusted Credentials]. Because the built-in CA certificates reside on the read-only system partition, the TOE places a copy of any disabled built-in certificate into the /data/misc/user/X/cacerts-removed/ directory, where 'X' represents the user's number (which starts at 0). The TOE stores added CA certificates in the corresponding /data/misc/user/X/cacerts-added/ directory and also stores a copy of the CA certificate in the user's Secure Key Storage (residing in the /data/misc/keystore/user_X/ directory). The TOE uses Linux file permissions that prevent any mobile application or entity other than the TSF from modifying these files. Only applications registered as an administrator (such as an MDM Agent Application) have the ability to access these files, staying in accordance to the permissions established in FMT_SMF_EXT.1 and FMT_MOF_EXT.1.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.3.7 Inter-TSF User Data Transfer Protection (Applications) (MDFPP32:FDP_UPC_EXT.1/APPS)

#### 2.3.7.1 MDFPP32:FDP_UPC_EXT.1/APPS.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Testing Assurance Activities**: None Defined

#### 2.3.7.2 MDFPP32:FDP_UPC_EXT.1/APPS.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.3 of the ST states the TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using TLS, HTTPS, and Bluetooth DR/EDR and LE. Mobile applications can use the following Android APIs for TLS, HTTPS, and Bluetooth respectively:

SSL:

javax.net.ssl.SSLContext:

https://developer.android.com/reference/javax/net/ssl/SSLSocket

Developers then need to swap SocketFactory for SecureSocketFactory, part of a private library provided by Google.

Developers can request this library by emailing: niapsec@google.com

HTTPS:

javax.net.ssl.HttpsURLConnection:

https://developer.android.com/reference/javax/net/ssl/HttpsURLConnection

Developers then need to swap HTTPSUrlConnections for SecureUrl part of a private library provided by Google.

Developers can request this library by emailing: niapsec@google.com

Bluetooth:

android.bluetooth:

http://developer.android.com/reference/android/bluetooth/package-summary.html

**Component Guidance Assurance Activities**: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (protection channel) described


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

in these requirements, and verify that the APIs implemented to support this requirement include the appropriate settings/parameters so that the application can both provide and obtain the information needed to assure mutual identification of the endpoints of the communication as required by this component.

The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocol(s) selected for use by the applications.

Section 10 of the guidance details the list of evaluated cryptographic APIs. This list includes cryptographic APIs, Key management, and certificate management, TLS, and HTTPS.

**Component Testing Assurance Activities**: 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.

Test 1 – For TLS/HTTPS, see Test Case FCS_TLSC_EXT.1.1 where an application is used to make TLS/HTTPS connections with all claimed ciphers. In each case an application data packet (showing the session is encrypted) is sent prior to disconnecting. For VPN, see FMT_MOF_EXT.1 test cases 2-3 where per-App VPN policies were configured demonstrating that VPNs could be assigned to specific applications.

Test 2 – This was performed with test case 1 where successful connections were made and the secure protocols were observed in the packet capture.

### 2.3.8 Inter-TSF User Data Transfer Protection (Bluetooth) (MDFPP32:FDP_UPC_EXT.1/BLUETOOTH)

#### 2.3.8.1 MDFPP32:FDP_UPC_EXT.1/BLUETOOTH.1

**TSS Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.3.8.2 MDFPP32:FDP_UPC_EXT.1/BLUETOOTH.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.3 of the ST states the TOE supports a means for non-TSF applications to initiate Bluetooth BD/EDR and LE connections.

The TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using Bluetooth DR/EDR and LE. Mobile applications can use the following Android APIs for Bluetooth respectively:

Bluetooth:

android.bluetooth:

http://developer.android.com/reference/android/bluetooth/package-summary.html

This matches the associated requirement in the ST.

**Component Guidance Assurance Activities**: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (protection channel) described in these requirements, and verify that the APIs implemented to support this requirement include the appropriate settings/parameters so that the application can both provide and obtain the information needed to assure mutual identification of the endpoints of the communication as required by this component.

The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocol(s) selected for use by the applications.

Section 4 (Bluetooth Configuration) in the Admin Guide provides the instructions and steps for establishing a secure channel to Bluetooth and how to interact with a Bluetooth device.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Section 10 in the Admin Guide details the evaluated cryptographic APIs that developers can use, this includes APIs for cryptographic APIs, Key management, APIs used by applications for configuring the reference identifier, APIs for validation checks, and TLS, HTTPS, Bluetooth BR/EDR, BLE.

**Component Testing Assurance Activities**: 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.

Test 1 – For Bluetooth BR/EDR, the evaluator cleared the BT HCI snoop log on the TOE and paired the TOE device to another TOE peer device. The evaluator also forced data to go from TOE device to the peer device using file sharing. See Test Case BT10:FIA_BLT_EXT.2-t1 where the evaluator included captures of file transfer traffic in order to show mutual authentication. Those captures also show that the communication channel is encrypted.

For Bluetooth LE, the evaluator cleared the BT HCI snoop log on the TOE and paired the TOE device to another TOE peer device using Bluetooth LE APIs. The evaluator browsed the available services authorized by the device to generate some LE traffic. See Test Case BT10:FTP_BLT_EXT.3/LE-1 where the evaluator included captures of the LE activity and demonstrated the correct keys present. Those captures also show that the communication channel is encrypted.

Test 2 – This was performed with test case 1 where successful connections were made and the secure protocols were observed in the packet capture.

## 2.4 Identification and Authentication (FIA)

### 2.4.1 Authentication Failure Handling (MDFPP32:FIA_AFL_EXT.1)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.4.1.1 MDFPP32:FIA_AFL_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.1.2 MDFPP32:FIA_AFL_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.1.3 MDFPP32:FIA_AFL_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.1.4 MDFPP32:FIA_AFL_EXT.1.4

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.1.5 MDFPP32:FIA_AFL_EXT.1.5

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.4.1.6 MDFPP32:FIA_AFL_EXT.1.6

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states The TOE maintains in persistent storage, for each user, the number of failed password logins since the last successful login (the phone, in its evaluated configuration, only supports password authentication), and upon reaching the maximum number of incorrect logins, the TOE performs a full wipe of all protected data (and in fact, wipes all user data). An administrator can adjust the number of failed logins for the cryptlock screen from the default of ten failed logins to a value between 0 (deactivate wiping) and 50 through an MDM. The TOE validates passwords by providing them to Android's Gatekeeper (which runs in the Trusted Execution Environment). If the presented password fails to validate, the TOE increments the incorrect password counter before displaying a visual error to the user. Android's Gatekeeper keeps this password counter in persistent secure storage and increments the counter before validating the password. Upon successful validation of the password, this counter is reset back to zero. By storing the counter persistently, and by incrementing the counter prior to validating it, the TOE ensures a correct tally of failed attempts even if it loses power.

**Component Guidance Assurance Activities**: The evaluator shall verify that the AGD guidance describes how the administrator configures the maximum number of unique unsuccessful authentication attempts.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Section 3.6 of the guidance details how to configure the maximum number of authentication failures.

**Component Testing Assurance Activities**: 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.

Test 1a – The evaluator configured the retry limit to 50 in order to test a maximum configured value and then attempted that number of failed attempts to ensure that it was enforced. After the 49$^{th}$ attempt, the evaluator rebooted the phone by removing the battery and then attempted to use the incorrect password again for the 50$^{th}$ time. As expected, the power off did not affect the failed login count and the phone initiated a factory reset.

Test 1b – N/A – the TOE does not support any BAF.

Test 2 – See Test 1. The TSS explains that the TOE validates passwords by providing them to Android's Gatekeeper (which runs in the Trusted Execution Environment). If the presented password fails to validate, the TOE increments the incorrect password counter before displaying a visual error to the user. Android's Gatekeeper keeps this password counter in persistent secure storage and increments the counter before validating the password. Upon successful validation of the password, this counter is reset back to zero. By storing the counter persistently, and by incrementing the counter prior to validating it, the TOE ensures a correct tally of failed attempts even if it loses power.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.4.2 Bluetooth User Authorization (BT10:FIA_BLT_EXT.1)

#### 2.4.2.1 BT10:FIA_BLT_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall examine the TSS to ensure that it contains a description of when user permission is required for Bluetooth pairing, and that this description mandates explicit user authorization via manual input for all Bluetooth pairing, including application use of the Bluetooth trusted channel and situations where temporary (non-bonded) connections are formed. The evaluator shall examine the API documentation provided according to Section 5.2.2 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.

Section 6.4 of the ST states the TOE requires explicit user authorization before it will pair with a remote Bluetooth device. When pairing with another device, the TOE requires that the user either confirm that a displayed numeric passcode matches between the two devices or that the user enter (or choose) a numeric passcode that the peer device generates (or must enter). The TOE requires this authorization (via manual input) for mobile application use of the Bluetooth trusted channel and in situations where temporary (non-bonded) connections are formed.

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

Section 4.1 of the guidance details Bluetooth pairing and clearly identifies the user authorization screens and instructions.

**Component Testing Assurance Activities**: Test 1: The evaluator shall perform the following steps:

Step 1: Initiate pairing with the TOE from a remote Bluetooth device that requests no man-in-the-middle protection, no bonding, and claims to have NoInputNoOutput input-output (IO) capability. (Such a device will attempt to evoke behavior from the TOE that represents the minimal level of user interaction that the TOE supports during pairing.)

Step 2: Verify that the TOE does not permit any Bluetooth pairing without explicit authorization from the user (e.g. the user must have to minimally answer 'yes' or 'allow' in a prompt).


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 1 – The evaluator requested Bluetooth pairing with a device that requests no man-in-the-middle protection, no bonding, and claims to have NoInputNoOutput input-output (IO) capability. The evaluator observed that the user had to explicitly authorize the pairing.

### 2.4.3 Bluetooth Mutual Authentication (BT10:FIA_BLT_EXT.2)

#### 2.4.3.1 BT10:FIA_BLT_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states the TOE prevents data transfer of any type until Bluetooth pairing has completed. Additionally, the TOE supports OBEX (OBject Exchange) through L2CAP (Logical Link Control and Adaptation Protocol). Any new connection or data transfer attempts must first be accepted by the user, else the connection/data is rejected. In the event that there is no user input, the request times out and defaults to rejecting the attempt

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The evaluator shall perform the following test:

Test 1: The evaluator shall use a Bluetooth tool to attempt to access TOE files using the OBEX Object Push service and verify that pairing and mutual authentication are required by the TOE before allowing access. (If the OBEX Object Push service is unsupported on the TOE, a different service that transfers data over Bluetooth L2CAP and/or RFCOMM may be used in this test.)

Test 1: The evaluator turned on Bluetooth sniffing on a debug device and then paired the TOE with a second device and transferred a screenshot. A packet capture confirmed mutual authentication.

### 2.4.4 Rejection of Duplicate Bluetooth Connections (BT10:FIA_BLT_EXT.3)

#### 2.4.4.1 BT10:FIA_BLT_EXT.3.1

**TSS Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall ensure that the TSS describes how Bluetooth connections are maintained such that two devices with the same Bluetooth device address are not simultaneously connected and such that the initial session is not superseded by any following session initialization attempts.

Section 6.4 of the ST states the TOE rejects duplicate Bluetooth connections by only allowing a single session per paired device. This ensures that when the TOE receives a duplicate session attempt while the TOE already has an active session with that device, then the TOE ignores the duplicate session.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test 1 - The evaluator configured two remote devices to share the same Bluetooth address. The evaluator then enabled Bluetooth packet sniffing and initiated a file transfer from the first remote device to the TOE. While the file was being transferred from the first remote device, the evaluator configured a second remote device with the same address to send a file to the TOE. The evaluator confirmed that the TOE was not prompted about a second transfer and that the first transfer completed. The evaluator analyzed the packet capture to ensure that the TOE received packets from the second transfer but ignored the second connection.

Next, the evaluator paired a device (Peer-1) with the TOE. Upon pairing, the evaluator then reconnected Peer-1 with the TOE and immediately tried to connect another device sharing Peer-1's Bluetooth address (Peer-2) with the TOE. The packet capture from the TOE and UI feedback on Peer-2 indicate that the duplicate connection was rejected while the packet capture and UI feedback on Peer-1 and the TOE indicate that the initial connection was successful.

### 2.4.5 Secure Simple Pairing (BT10:FIA_BLT_EXT.4)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.4.5.1 BT10:FIA_BLT_EXT.4.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.5.2 BT10:FIA_BLT_EXT.4.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall verify that the TSS describes the secure simple pairing process.

Section 6.4 of the ST states the TOE'S Bluetooth host and controller supports Bluetooth Secure Simple Pairing and the TOE utilizes this pairing method when the remote host also supports it.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: Test 1: 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.

Test 1 – The evaluator paired the TOE with another android device that supported Secure Simple Pairing. The evaluator checked the packet capture to verify that Secure Simple Pairing was used.

### 2.4.6 Trusted Bluetooth Device User Authorization (BT10:FIA_BLT_EXT.6)

#### 2.4.6.1 BT10:FIA_BLT_EXT.6.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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'.

Section 6.4 of the states the TOE requires explicit user authorization before granting trusted remote devices access to services associated with the OPP and MAP Bluetooth profiles. Additionally, the TOE requires explicit user authorization before granting untrusted remote devices access to services associated with all following Bluetooth profiles.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The evaluator shall perform the following tests for each service protected according to this requirement:

Test 1: While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a 'protected' Bluetooth service (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.

Test 1 – The evaluator started the test by enabling Bluetooth on the TOE device. A file transfer was attempted by Bluetooth OPP File Transfer, which was rejected on the TOE device. The file transfer did not succeed.

Test 2 -- The evaluator started the test by enabling Bluetooth on the TOE device. A file transfer was attempted by Bluetooth OPP File Transfer, which was approved on the TOE device. The file transfer did succeed.

### 2.4.7 Untrusted Bluetooth Device User Authorization (BT10:FIA_BLT_EXT.7)

#### 2.4.7.1 BT10:FIA_BLT_EXT.7.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The TSS evaluation activities for this component are addressed by FIA_BLT_EXT.6.

See MDFPP32: FIA_BLT_EXT.6.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test 1 – Not applicable, the TOE does not differentiate between trusted and untrusted devices for granting access to services (Both FIA_BLT_EXT.6 and FIA_BLT_EXT.7 claim OPP and MAP profiles)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 2 – Not applicable, the TOE does not differentiate between trusted and untrusted devices for granting access to services (Both FIA_BLT_EXT.6 and FIA_BLT_EXT.7 claim OPP and MAP profiles)

Test 3 – Not applicable – the same authorization is required per OBEX tests.

Test 4 -. Not applicable – test 3 does not apply.

Test 5 - Not applicable – test 3 does not apply.

### 2.4.8 Port Access Entity Authentication (WLANCEP10:FIA_PAE_EXT.1)

#### 2.4.8.1 WLANCEP10:FIA_PAE_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The evaluator shall perform the following tests:

- Test 1: The evaluator shall demonstrate that the TOE has no access to the test network. After successfully authenticating with an authentication server through a wireless access system, the evaluator shall demonstrate that the TOE does have access to the test network.

- Test 2: The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid client certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.

- Test 3: The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid authentication server certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.

Test 1 - The evaluator loaded the applicable root CA on the TOE and then attempted a connection and the evaluator ensured the connection was established. The evaluator then repeated the test first loading a bad (expired) client certificate and then a bad (expired) server certificate and confirmed that the connections could not be made with bad certificates.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 2 – See Test 1. This test has been performed where good certificates, bad client certificate, and bad server certificate variations are all tested.

Test 3 -- See Test 1. This test has been performed where good certificates, bad client certificate, and bad server certificate variations are all tested.

### 2.4.9 Password Management (MDFPP32:FIA_PMG_EXT.1)

#### 2.4.9.1 MDFPP32:FIA_PMG_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: 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.

The "Password Management" table entry in Section 3.6 of the Admin-Guide provides the API for setting the minimum passwords length.

Section 3.7 (Password Recommendations) in the Admin Guide provides guidance to administrators on the composition of strong passwords.

**Component Testing Assurance Activities**: 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.

Test 1 – The evaluator set the minimum password length to 16 characters (with Android's built-in 16 character maximum, this requires a 16 character password exactly). The evaluator then went through all the possible characters claimed in the ST, 16 at a time. The evaluator was able to verify that all claimed characters were valid. The evaluator attempted to set a password of 15 and 17 characters and was denied as expected.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

## 2.4.10 Authentication Throttling (MDFPP32:FIA_TRT_EXT.1)

### 2.4.10.1 MDFPP32:FIA_TRT_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states that Android's GateKeeper throttling is used to prevent brute-force attacks. After a user enters an incorrect password, GateKeeper APIs return a value in milliseconds (500ms default) in which the caller must wait before attempting to validate another password. Any attempts before the defined amount of time has passed will be ignored by GateKeeper. Gatekeeper also keeps a count of the number of failed validation attempts since the last successful attempt. These two values together are used to prevent brute-force attacks of the TOE's password.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.4.11 Multiple Authentication Mechanisms (MDFPP32:FIA_UAU.5)

### 2.4.11.1 MDFPP32:FIA_UAU.5.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.4.11.2 MDFPP32:FIA_UAU.5.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states that the TOE, in its evaluated configuration, allows the user to authenticate using a password. Upon boot, the first unlock screen presented requires the user to enter their password to unlock the device.

Upon device lock during normal use of the device, the user has the ability to unlock the phone by entering their password. Throttling of this input can be read about in the FIA_AFL_EXT.1 section. The entered password is compared to a value derived as described in the key hierarchy and key table above (FCS_STG_EXT.2 and FCS_CKM_EXT.4, respectively).

Some security related user settings (e.g. changing the password, setting up SmartLock, etc.) and actions (e.g. factory reset) require the user to enter their password before modifying these settings or executing these actions.

The TOE's evaluated configuration disallows other authentication mechanisms, such as pattern, PIN, or Smart Lock mechanisms (on-body detection, trusted places, trusted devices, trusted face, trusted voice).

**Component Guidance Assurance Activities**: The evaluator shall verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.

The TOE, in its evaluated configuration, allows the user to authenticate using a password.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

The “Password Management” table entry in Section 3.6 (Common Criteria Related Settings) of the Admin Guide provides the APIs for setting the minimum passwords length, password complexity, password expiration and maximum number of authentication failures.

Section 9 (FDP_DAR_EXT.2 & FCS_CKM.2(2) – Sensitive Data Protection Overview) in the Admin Guide states that sensitive data protection including biometric protections (while not relevant for this product) are enabled by default by using the Strong configuration via the NIAPSEC library. The library provides APIs via SecureContextCompat to write files when the device is either locked or unlocked. Reading an encrypted file is only possible when the device is unlocked and authenticated.

Section 10.1 (Cryptographic APIs) in the Admin Guide provides the APIs for creating a BiometricSupport object to handle key authentication (while named “biometric”, it does not require biometrics – it is a code reference).

**Component Testing Assurance Activities**: 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.

Test 1 & 2 –

The evaluator configured the TOE with a password that could be used as an authentication factor. The evaluator then locked the device and used the authentication method to unlock the phone. The password was able to be used to transition the TOE from the locked state to the unlocked state.

Next, the evaluator rebooted the device and confirmed that a password was required to unlock and decrypt the phone. The evaluator also confirmed that the TOE requires entry of the password before allowing the password authentication factor to be changed. The evaluator also tested wiping the device and confirmed that the password was required as a final step.

### 2.4.12 Re-authenticating (Credential Change) - per TD0706 (MDFPP32:FIA_UAU.6/CREDENTIAL)

#### 2.4.12.1 MDFPP32:FIA_UAU.6.1/CREDENTIAL

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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 configurating 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.

Test 1 – The evaluator used the TOE's Lock Screen settings under the Settings Application to change the screen lock password. Before the evaluator could choose the type of screen lock, the evaluator was prompted for the old password.

Test 2 – Not applicable, the TOE does not claim any BAF as a part of FIA_UAU.5.1.

Test 3 – Not applicable, the TOE does not claim any HAF as a part of FIA_UAU.5.1.

## 2.4.13 Re-authenticating (TSF Lock) - per TD0706 (MDFPP32:FIA_UAU.6/LOCKED)

### 2.4.13.1 MDFPP32:FIA_UAU.6.1/LOCKED

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: Test 1: The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF.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.

See FTA_SSL.1 where the evaluator transitions to a locked state (including after the inactivity time threshold has been met as demonstrated in FMT_SMF). The evaluator then verified that the TOE required the entry of the Password Authentication Factor before transitioning to the unlocked state.

#### 2.4.13.2 MDFPP32:FIA_UAU.6.2/LOCKED

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 – The evaluator configured the TOE to transition to the locked state after a time of inactivity. In each case after the session locks a password dialog is presented and a valid password must be entered to unlock the TOE.

Test 2 – Not applicable, the TOE does not claim any BAF as a part of FIA_UAU.5.1.

Test 3 – Not applicable, the TOE does not claim any HAF as a part of FIA_UAU.5.1.

Test 4 – The evaluator transitioned the TOE from a locked state (by pressing the power button on the TOE) to the unlocked state using the configured password.

Test 5 – Not applicable, the TOE does not claim any BAF as a part of FIA_UAU.5.1.

Test 6 – Not applicable, the TOE does not claim any HAF as a part of FIA_UAU.5.1.

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.4.14 Protected Authentication Feedback (MDFPP32:FIA_UAU.7)

#### 2.4.14.1 MDFPP32:FIA_UAU.7.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states the TOE allows the user to enter the user's password from the lock screen. The TOE will, by default, display the most recently entered character of the password briefly or until the user enters the next character in the password, at which point the TOE obscures the character by replacing the character with a dot symbol.

**Component Guidance Assurance Activities**: The evaluator shall verify that any configuration of this requirement is addressed in the AGD guidance and that the password is obscured by default.

No configuration is required nor available for this requirement, thus it is not present in the AGD.

**Component Testing Assurance Activities**: 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.

Test 1 –The evaluator observed that at the lockscreen and change password screen no characters were displayed to the user.

Test 2 – Not applicable, the TOE does not claim any BAF as a part of FIA_UAU.5.1.

### 2.4.15 Authentication for Cryptographic Operation (MDFPP32:FIA_UAU_EXT.1)

#### 2.4.15.1 MDFPP32:FIA_UAU_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states the TOE's key hierarchy requires the user's password in order to derive the KEK_* keys in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized for Data-At-Rest encryption, and thus cannot decrypt the user's protected data.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The following tests may be performed in conjunction with FDP_DAR_EXT.1 and FDP_DAR_EXT.2.

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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

See the test cases in FDP_DAR_EXT.1 and FDP_DAR_EXT.2.

## 2.4.16 Timing of Authentication (MDFPP32:FIA_UAU_EXT.2)

### 2.4.16.1 MDFPP32:FIA_UAU_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.4.16.2 MDFPP32:FIA_UAU_EXT.2.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall verify that the TSS describes the actions allowed by unauthorized users in the locked state.

Section 6.2 of the ST states that the TOE, when configured to require a user password, allows a user to perform the actions assigned in FIA_UAU_EXT.2.1 (see selections in section 5 for FIA_UAU_EXT.2) without first successfully authenticating. Choosing the input method allows the user to select between different keyboard devices (say, for example, if the user has installed additional keyboards). Note that the TOE automatically names and saves (to the internal Flash) any screen shots or photos taken from the lock screen, and the TOE provides the user no opportunity to name them or change where they are stored.

When configured, the user can also launch Google Assistant to initiate some features of the phone. However, if the command requires access to the user's data (e.g. contacts for calls or messages), the phone requires the user to manually unlock the phone before the action can be completed.

Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating. Additionally, the TOE provides the user the ability to hide the contents of notifications once a password (or any other locking authentication method) is enabled.

**Component Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: 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.

The evaluator locked the phone and attempted to perform actions not listed in the selection. The evaluator confirmed that authentication was required to proceed with all attempted actions not listed in the selection.

## 2.4.17 X.509 Validation of Certificates - per TD0603 (MDFPP32:FIA_X509_EXT.1)

### 2.4.17.1 MDFPP32:FIA_X509_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.4.17.2 MDFPP32:FIA_X509_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states the TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate. Additionally, the TOE verifies the extendedKeyUsage Server Authentication purpose during WPA2/EAP-TLS negotiation. The TOE'S certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired; or if not yet valid, whether the certificate contains the appropriate X.509 extensions [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the Server Authentication purpose in the ExtendedKeyUsagefield]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung "up" in the chain and that the chain ends in a self-signed certificate present in


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

either the TOE'S trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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 certificatesconditional 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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 8.1: (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 8.2: (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.

Test 1 – Part A: The evaluator loaded the applicable root CA on the TOE and then attempted a connection and confirmed that the connection succeeded with a complete certificate chain. Part B: The evaluator then configured the server to utilize an incorrect CA (ecdsa based instead of rsa, simulating removing the root-ca) and verified the connection failed. Part C-1: The evaluator then configured the server to utilize a certificate that omitted the basicConstraints field and verified that the connection failed. Part C-2: The evaluator then configured the server to utilize a certificate that had the basicConstraints cA Flag field set to false and verified that the connection failed. Part C-3: the evaluator configured the server to have a certificate that omitted the CA Signing bit of the key usage field and verified that the connection failed. Part C-4: The evaluator configured the server to have a certificate that indicates an incorrect (too short) certificate path length and verified that the connection failed. This test was repeated for both TLS and HTTPS.

Test 2 - For this test, the evaluator alternately configured freeradius on a test server to send an authentication certificate 1) that is valid, 2) that is expired, and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server and confirmed that the connection


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

succeeded in the first case, but failed in the next two cases with the expired certificates. This test was performed for 2 variations as follows: TLS/HTTPS and TLS.

Test 3- For this test, the evaluator alternately configured stunnel on a test server to send an authentication certificate 1) that is valid, 2) that is revoked, and 3) issued by an intermediate CA that is revoked. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server and confirmed that the connection succeeded in the first case, but failed in the next two cases with the revoked certificates. This test was performed for 2 variations as follows: TLS/HTTPS and TLS

Test 4 - For this test, the evaluator alternately configured stunnel on a test server to send an authentication certificate 1) that is valid, 2) that has a root that refers to an OCSP revocation server where the signer lacks OCSPSigning, 3) issued by an intermediate CA whose issuer CA refers to an OCSP revocation server where the signer lacks OCSPSigning, and 4) issued by an intermediate CA referring to an OCSP revocation server where the signer lacks OCSPSigning. The connection was rejected in each invalid case. This test was performed for 2 variations as follows: TLS/HTTPS and TLS.

Test 5- For this test, the evaluator alternately configured freeradius on a test server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server. As expected, in the first case, the connection was accepted. In the following three cases, the connection was rejected due to the certificate being modified/corrupted. This test was performed for 2 variations as follows: TLS/HTTPS and TLS.

Test 6 – This test has been performed in conjunction with test 5.

Test 7 – This test has been performed in conjunction with test 5.

Test 8- For this test, the evaluator alternatively configured a valid EC certificate chain and one with a modified intermediate EC certificate and verified that the connection succeeded and failed respectively. This test was performed for 2 variations as follows: TLS/HTTPS and TLS.

### 2.4.18 X.509 Certificate Validation (WLANCEP10:FIA_X509_EXT.1/WLAN)

#### 2.4.18.1 WLANCEP10:FIA_X509_EXT.1.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.4.18.2 WLANCEP10:FIA_X509_EXT.1.2/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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. (TD0439 applied)

Section 6.4 of the ST states the TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate. Additionally, the TOE verifies the extendedKeyUsage Server Authentication purpose during WPA2/EAP-TLS negotiation. The TOE'S certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired; or if not yet valid, whether the certificate contains the appropriate X.509 extensions [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the Server Authentication purpose in the ExtendedKeyUsagefield]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung "up" in the chain and that the chain ends in a self-signed certificate present in either the TOE'S trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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).

(TD0439 applied)

Test 1: The evaluator first loaded a valid RSA certificate chain and verified the connection succeeded. Next, the evaluator configured the TOE client to use an ECDSA root certificate with the RSA certificate chain (simulating deleting the root-ca, which must have a value selected) and verified the connection failed.

Test 2: the evaluator alternately configured freeradius on a test server to send an authentication certificate 1) that is valid, 2) that is expired, and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server and confirmed the connection succeeded only if there are no expired certificates.

Test 3: For this test, the evaluator alternately configured freeradius on a test server to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false. In both cases, the evaluator then attempted to connect the TLSC TOE client to the test server and confirmed the connection was rejected in each case.

Test 4: Completed in conjunction with Test 3.

Test 5: For this test, the evaluator alternately configured freeradius on a test server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server expecting the connection to succeed only if the certificate is not modified/corrupted.

Test 6: Completed in conjunction with Test 5.

Test 7: Completed in conjunction with Test 5.

### 2.4.19 X.509 Certificate Authentication - per TD0623 (MDFPP32:FIA_X509_EXT.2)

#### 2.4.19.1 MDFPP32:FIA_X509_EXT.2.1

**TSS Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

### 2.4.19.2 MDFPP32:FIA_X509_EXT.2.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.4 of the ST states the TOE uses X.509v3 certificates during EAP-TLS, TLS, and HTTPS. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates), and while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CA's cannot validate successfully. In addition, a user and an administrator/MDM can import a new trusted CA certificate into the Trust Anchor Database (the TOE stores the new CA certificate in the Security Key Store). Users and administrators/MDMs can also import new client certificates as well via the settings UI and the TOE's MDM APIs, respectively. Users then select which client certificate to present during configuration of the connection while administrators configure it while creating Wi-Fi connection profiles.

The TOE does not establish TLS connections itself (beyond EAP-TLS used for WPA2 Wi-Fi connections), but provides a series of APIs that mobile applications can use to check the validity of a peer certificate. The mobile application, after correctly using the specified APIs, can be assured as to the validity of the peer certificate and be assured that the TOE will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation [through OCSP]). If, during the process of certificate verification, the TOE cannot establish a connection with the server acting as the OCSP Responder, the TOE will not deem the server's certificate as valid and will not establish a TLS connection with the server.

The user or administrator explicitly specifies the trusted CA that the TOE will use for EAP-TLS authentication of the server's certificate. For mobile applications, the application developer will specify whether the TOE should use the Android system Trusted CAs, use application-specified trusted CAs, or a combination of the two. In this way, the TOE always knows which trusted CAs to use.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

The TOE, when acting as a WPA2 supplicant uses X.509 certificates for EAP-TLS authentication. Because the TOE may not have network connectivity to a revocation server prior to being admitted to the WPA2 network and because the TOE cannot determine the IP address or hostname of the authentication server (the Wi-Fi access point proxies the supplicant's authentication request to the server), the TOE will accept the certificate of the server.

**Component Guidance Assurance Activities**: 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.

Not applicable. The administrator specifying the default action is not selected in the ST.

**Component Testing Assurance Activities**: 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.

For this test, the evaluator alternately configured stunnel on a test server to send an authentication certificate with valid/accessible revocation servers and an authentication certificate with revocation information referring to an inaccessible revocation server. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server expecting the connection to be successful when the revocation server is accessible and when the revocation server is not accessible only if that behavior is claimed for the TOE. In this test, the test server uses a server certificate containing CDP/CRL and AIA/OCSP extensions that point to a revocation server at an empty address, and because no such server exists (this script actively pings that address to ensure there is no response), the TOE's revocation requests will time out. The results are iterated for 2 variations as follows: HTTPS and TLS.

## 2.4.20 X.509 Certificate Authentication (EAP-TLS) (WLANCEP10:FIA_X509_EXT.2/WLAN)

### 2.4.20.1 WLANCEP10:FIA_X509_EXT.2.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.4.20.2 WLANCEP10:FIA_X509_EXT.2.2/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.4 of the TSS states that the TOE uses X.509v3 certificates during EAP-TLS, TLS, and HTTPS. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates), and while the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CA's cannot validate successfully. In addition, a user and an administrator/MDM can import a new trusted CA certificate into the Trust Anchor Database (the TOE stores the new CA certificate in the Security Key Store). Users and administrators/MDMs can also import new client certificates as well via the settings UI and the TOE's MDM APIs, respectively. Users then select which client certificate to present during configuration of the connection while administrators configure it while creating Wi-Fi connection profiles.

The TOE does not establish TLS connections itself (beyond EAP-TLS used for WPA2 Wi-Fi connections), but provides a series of APIs that mobile applications can use to check the validity of a peer certificate. The mobile application, after correctly using the specified APIs, can be assured as to the validity of the peer certificate and be assured that the TOE will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation [through OCSP]). If, during the process of certificate verification, the TOE cannot establish a connection with the server acting as the OCSP Responder, the TOE will not deem the server's certificate as valid and will not establish a TLS connection with the server.

The user or administrator explicitly specifies the trusted CA that the TOE will use for EAP-TLS authentication of the server's certificate. For mobile applications, the application developer will specify whether the TOE should use the Android system Trusted CAs, use application-specified trusted CAs, or a combination of the two. In this way, the TOE always knows which trusted CAs to use.

The TOE, when acting as a WPA2 supplicant uses X.509 certificates for EAP-TLS authentication. Because the TOE may not have network connectivity to a revocation server prior to being admitted to the WPA2 network and because the


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

TOE cannot determine the IP address or hostname of the authentication server (the Wi-Fi access point proxies the supplicant's authentication request to the server), the TOE will accept the certificate of the server.

**Component Guidance Assurance Activities**: 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.

Section 3.6 of the Admin Guide describes the common criteria related settings. This section details the certificate management configuration and API.

Section 10.3 of the Admin Guide details certificate validation for TLS and HTTPS. SecureURL is included in the NIAPSEC library. SecureURL automatically configures TLS and can perform certificate and host validation checking. At construction, SecureURL requires a reference identifier. The Admin Guide also details all associated public methods and APIs for certificate use, validation, and revocation.

**Component Testing Assurance Activities**: None Defined

### 2.4.21 Request Validation of Certificates (MDFPP32:FIA_X509_EXT.3)

#### 2.4.21.1 MDFPP32:FIA_X509_EXT.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.4.21.2 MDFPP32:FIA_X509_EXT.3.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Guidance Assurance Activities**: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security function (certificate validation) described in this requirement. This documentation shall be clear as to which results indicate success and failure.

Section 10 API specification of the Admin Guide details API's used for configuring reference identifiers and validation checks.

**Component Testing Assurance Activities**: 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.

Test 1: The TOE provides two APIs to verify a certificate or a chain of certificates. The evaluator utilized these APIs in the Gossamer test applications used during FIA_X509_EXT.1 and FTP_ITC_EXT.1 testing. Those tests demonstrated that the TOE's certificate validation APIs correctly detected invalid certificate chains.

## 2.5 Security Management (FMT)

### 2.5.1 Management of Security Functions Behavior - per TD0658 (MDFPP32:FMT_MOF_EXT.1)

#### 2.5.1.1 MDFPP32:FMT_MOF_EXT.1.1

**TSS Assurance Activities**: 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.

Section 6.5 of the ST states that the TOE provides the management functions as detailed in the ST's Table 3 Security Management Functions. This includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE'S ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.5.1.2 MDFPP32:FMT_MOF_EXT.1.2

**TSS Assurance Activities**: 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.

The TOE provides the management functions described in section 5.1.5.2 of the ST. The section includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE'S ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test 1 - The developer provided a sample MDM application that was copied into local device storage and installed. The application is an MDM agent that allows the MDM APIs and functions to be performed through the application UI. Once installed and enabled, the TOE device is effectively enrolled and policies can be defined and applied to the device.

Test 2 – The evaluator used the MDM application to deploy policies and attempt to override settings both as a user (if setting was available) and as an application (if API was available). In each case, the evaluator attempted to


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

bypass the policy set by the evaluator and was unable to circumvent the MDM application policy. Policies were deployed in order to restrict access to the following options:

1. Configure password policy:
   a. minimum password length
   b. minimum password complexity
   c. maximum password lifetime
2. Configure session locking policy
   a. screen-lock enabled/disabled
   b. screen lock timeout
   c. number of authentication failures
3. Enable/disable the VPN protection:
   a. across device
   c. on a per-group of applications processes basis
4. Enable/disable [Bluetooth, Wi-Fi, Cellular, NFC]
5. Enable/disable [microphone, camera]:
   a. across device
   c. no other method
8. Configure application installation policy by:
   c. denying installation of applications
18. configure the Bluetooth trusted channel:
   a. disable/enable the Discoverable mode (for BR/EDR)
   b. change the Bluetooth device name
   k. no other Bluetooth configuration
21. Enable/disable location services:
   a. across device
25. Enable/disable [Wi-Fi tethering, USB tethering, and Bluetooth tethering]
26. Enable/disable developer modes
40. enable/disable backup of [***all applications***] to [***locally connected system***]
41. Enable/disable [a. Hotspot functionality authenticated by [pre-shared key], b. USB tethering authenticated by [no authentication]]
42. approve exceptions for sharing data between [***groups of application***]
44. Unenroll the TOE from management
45. Enable/disable the Always-On VPN protection
57. (WLAN1) configure security policy for each wireless network [a. [specify the CA(s) from which the TSF will accept WLAN authentication server certificate(s)], b. security type, c. authentication protocol, d. client credentials to be used for authentication]
58. (WLAN2) specify wireless networks (SSIDs) to which the TSF may connect

All other restrictions are tested with FMT_SMF_EXT.1.

Test 3 – See FMT_SMF_EXT.1.1

### 2.5.2 Specification of Management Functions - per TD0645 & TD0646 & TD0658 (MDFPP32:FMT_SMF_EXT.1)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.5.2.1 MDFPP32:FMT_SMF_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Test activities specified below shall take place in the test environment described in the evaluation activity for FPT_TUD_EXT.1.

Section 6.5 of the ST states that the TOE provides the management functions described in the table in FMT_SMF_EXT.1. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE's ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).

**Component Guidance Assurance Activities**: 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.

The table in Section 3.6 (Common Criteria Related Settings) of the Admin Guide provides the Common Criteria Related Settings.

- Function 4 is described in the Radio Control entry in the table and includes the APIs and settings for controlling access to Wi-Fi, GPS, NFC and Bluetooth.
- Function 5 is described in the Hardware Control entry in the table and includes the API and settings for controlling access to the microphone and camera.
- Function 11 is described in the Certificate Management entry in the table and includes the APIs and settings for importing and removing CA certificates to and from the Trust Anchor Database.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

- Function 13 is described in the TOE Management entry in the table and includes the settings instructions for enrolling the TOE in management.
- Function 14 is described in the Application Control entry in the table and includes the API for uninstalling applications.
- Function 19 is described in the Lockscreen entry in the table and includes the API for configuring whether notifications are displayed on the lockscreen.
- Function 27 – not applicable/not claimed in ST.
- Function 29 – not applicable/not claimed in ST.

**Component Testing Assurance 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).

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

- minimum password length

- minimum password complexity

- maximum password lifetime

Function 2

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

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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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 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. The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function for each radio.

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 â€“ 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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

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

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

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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

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.

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

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

Test 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

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

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

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

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

Test 13: The evaluator shall verify that user approval is required to enroll the device into management.

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. 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 email client. It does not, on the other hand, refer to data such as word processing documents (for a word processing app) or photos (for a photo or camera app).

Test 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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

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.

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-


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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]

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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 while the TOE may continue to receive data on the RX pins, it is not responding on TX 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]

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

Test 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]

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

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

Function 28 [conditional]

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

Function 29 [conditional]

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator shall also verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes any security functions (import, modification, or destruction of the Trust Anchor Database) allowed by applications.

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

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]

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

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

Function 32 [conditional]


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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]

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

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

Function 35 [conditional]

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

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

Function 36 [conditional]

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

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

Function 37 [conditional]

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

Function 38 [conditional]

Test 38: The test of this function is performed in conjunction with FPT_NOT_EXT.2.1.

Function 39 [conditional]

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

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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]

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.

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]

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

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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Function 47

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

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.

Test 1 - See the test case for FIA_PMG_EXT.1 where the evaluator tested with the password length configured to 16 characters and verified it was enforced. The evaluator also configured password complexity and ensured it was enforced. The evaluator configured a password expiration time to ensure the password expired after the configured timeframe.

Test 2 - The TOE doesn't allow screen lock to be disabled; only the timeout can be configured. The evaluator restricted the maximum screen timeout to 2 minutes and then verified that the screen lock and display timeout limits were restricted to no more than 2 minutes. The evaluator then disallowed screen timeout config and then confirmed that the action is no longer allowed to the user. See also test case for FTA_SSL_EXT.1, Test 1 where session timeout limits of 2 and 4 minutes are tested, showing that the TOE does in fact lock after those periods of inactivity. The evaluator then configured a maximum login failure value to 5 and confirmed that after 5 incorrect passwords the device began the process to factory reset and wipe the device. The evaluator then failed to type the correct password until the device wiped itself. See also Test Case for FIA_AFL_EXT.1, Test 1 where login failures are tested to be handled properly at each authentication interface.

Test 3a – Configure VPN protection and verify it can only be modified when allowed by the MDM agent.

Test 3b – Not claimed.

Test 3c – Configure and enable a VPN connection on the base profile and verify that the VPN is not available to the secondary, work profile.

Test 4 – The evaluators used a spectrum analyzer to look for Wifi, Bluetooth, NFC, and cellular signals when the radios were turned off. The evaluator then turned off each protocol and ensured the signals no longer existed. The evaluator observed signals with radios enabled and then observed (with the same configuration) the signals are absent when the radio is disabled.

Note: For the Cellular (1700-2100 Mhz) and Wi-Fi (2.4-5GHz) ranges, the evaluator observed calibration signals broadcast during the first few seconds of TOE boot/initialization, the evaluator investigated these signals and based upon the information learned, the implementation was determined to be acceptable. A NIAP decision was made to accept this implementation.

Test 5 – With the microphone enabled and usable, the evaluator used an MDM test application to disallow the microphone access across the device as the administrator. The evaluator confirmed that the microphone was disabled across the device and no longer usable. Next the evaluator confirmed that the Camera app was given


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

permission to use the phone's camera and the Camera was enabled and usable. The evaluator then denied the Camera app permission to use the phone's camera in the Settings UI and confirmed that the app was no longer allowed to use the phone's camera and was in fact disabled. Then the evaluator used the Settings UI to display all the Microphone permissions and confirmed that the Chrome app had permission to use the microphone and that it was usable. The evaluator then denied the Chrome app permission to use the microphone and confirmed that the microphone was no longer allowed and was in fact disabled.

Test 6 – The evaluator used the MDM test application to lock the TOE as administrator. See Test Case FTA_SSL_EXT.1, Test 2 where the TOE was locked by the user.

Test 7- The evaluator wiped the device using the MDM application. See Test Case FCS_CKM_EXT.5, Test 2 where the evaluator wiped the TOE as a device user and confirmed the TOE was actually wiped.

Test 8, a, c – The evaluator confirmed using the Settings UI on the TOE that applications (Chrome, Drive, and Gmail) were "Not allowed" to install unknown apps. The evaluator then used the MDM test application "Allow installs from unknown sources" function to allow installs from unknown sources, and then confirmed using the Settings UI on the TOE that they have actually been allowed. The evaluator then used the MDM test application "Disallow install unknown sources" function to disallow installs from unknown sources, and then confirmed using the Setting UI on the TOE that they have actually been disabled. Next the evaluator confirmed that the user has access to the Playstore and that it was usable to install applications. The evaluator confirmed that the Playstore was working properly by installing an application (WhatsApp Messenger) from the Playstore and confirming that the application actually got installed on the TOE. The evaluator then used the MDM test application "Disallow install apps" function to disallow installs from the Playstore, and then confirmed that no applications could be installed using the Playstore.

Next the evaluator used the Settings UI on the TOE and confirmed an application could be installed by the user. The evaluator then used the MDM test application "Disallow install apps" function to disallow installation, uninstalled the application, and attempted to reinstall the same application. Permission was denied.

Test 9 – See Test Case FCS_STG_EXT.1, Test 1.

Test 10 – See Test Case FCS_STG_EXT.1, Test 1.

Test 11 – The evaluator imported a certificate into the Trust Anchor database and removed a certificate from the database as a user and also using the MDM application. Certificates are repeatedly installed and used over the course of testing and the evaluator never encountered any issues using a configured certificate.

Test 12 - See Test 11 – the evaluator removed imported X509 certificates both as an administrator and as a user, and then confirmed that the certificates had actually been removed from the TOE.

Test 13 – The evaluator installed the MDM test application and enabled administration. This effectively enrolls the TOE. Note that the MDM test application is essentially an MDM agent application that provides direct access to the MDM accessible functions rather than requiring some remote communication protocols.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 14 - See Test FMT_MOF_EXT.1 Test 2-8 where applications are installed and removed repeatedly by the user. The evaluator identified an application and removed it using the administrator interface. The evaluator checked the Application Manager to ensure it was no longer evident on the TOE.

Test 15 - See the test cases for FPT_TUD_EXT.1.

Test 16 - See Test Case FMT_MOF_EXT.1 Test 2-8. The evaluator installed an application using the MDM test application and confirmed in the Application Manager that it was installed.

Test 17 - See Test 14 where applications are removed both as an administrator and as a user and shown to be no longer accessible. There is no distinction between mobile and enterprise applications.

Test case 18 - The evaluator locked the TOE and observed a number of notifications from previous device usage. The evaluator then configured the TOE to display no notifications when locked. The evaluator locked the device once again and found no evidence of any notifications. The evaluator then used the MDM to enable and disable notifications.

Test case 19 – The current version of Android is always encrypted, although it can be configured whether to require a password upon rebooting.

Test 20 – Not Applicable as the TOE device does not have removable media.

Test 21 – The evaluator ensured location services were on and could be used. The evaluator then turned off location services to see they could not be used - the application prompted the user to turn the services back on so they could be used. The evaluator turned the location services back on and then disabled them as the administrator - disabled all location services. The evaluator then observed that location services were disabled and could not be turned on. The location services were not usable and even after selecting OK to turn the services back on using the app, the services remained off and inaccessible.

Test 23 – Not applicable

Test 23 – Not applicable

Test 24 – Not applicable

Test 25 - The TOE supports three types of Tethering: Wi-Fi, blue tooth, and USB. The evaluator turned on each type of tethering and ensured that a connection could be made (Wi-Fi, USB, or a Bluetooth pairing).

Test 26- See Test FMT_MOF_EXT.1 Test 2-26 where developer mode is tested to show it can be enabled/disabled by the user and restricted/unrestricted by the administrator. The evaluator disabled developer mode, restarted the device and confirmed that developer mode was still not available.

Test 27 – Not applicable



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 28 – The evaluator wiped the Enterprise data from the device and then confirmed via the display that it was wiped.

Test 29 – Not applicable

Test 30 – Not applicable

Test 31 – Not applicable

Test 32 –The evaluator used TestDPC to ensure that the MDM Agent could read the security logs.

Test 33 – Not applicable

Test 34 – Not applicable

Test 35 – Not applicable

Test 36 – See the testing activity for FTA_TAB.1.

Test 37 – Not applicable

Test 38 – Not applicable

Test 39 – Not applicable

Test case 40 – The evaluator enabled Backup services and verified it was available. The evaluator then used TestDPC to disable Backup services and verified that mass storage was not available.

Test case 41a & 41b- See Test FMT_SMF_EXT.1 Test 25 where all three tethering features are tested to show they work, can be enabled/disabled by the user, and can be restricted/unrestricted by the administrator.

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

Test 43 - Not applicable

Test 44– See test case FMT_SMF_EXT.2.1 where the TOE is unenrolled from management

Test 45 – a) The evaluator configured the VPN as Always-On function. This test is performed in conjunction with FDP_IFC_EXT.1.1. See Test FDP_IFC_EXT.1, Test 1. b) The evaluator verified that, when the VPN is not established but Always-on-VPN is enabled, no traffic is able to leave the device. Next, the evaluator established the VPN connection and verified all traffic was encrypted and websites could be reached. Then, the evaluator disabled the VPN from the server side and verified no traffic left the device. c) The evaluator established a VPN connection (and verified it), then enabled airplane mode and verified the VPN connection disconnected. Finally, Airplane mode was disabled and the VPN connection was reestablished.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 46 - Not applicable

Test 47 - Not applicable

### 2.5.3 Specification of Management Functions (BT10:FMT_SMF_EXT.1/BT)

#### 2.5.3.1 BT10:FMT_SMF_EXT.1/BT.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.5 of the ST states that The TOE provides the management function described in Table 4 Bluetooth Management functions in section 5. As with Table 3 in the ST, the table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted. It is worth noting that the TOE'S ability to specify authorized application repositories takes the form of allowing enterprise applications (i.e., restricting applications to only those applications installed by an MDM Agent).


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Guidance Assurance Activities**: 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.

Section 4 of the Admin Guide explains how to use Bluetooth. It is the same level of detail as other management functions.

**Component Testing Assurance Activities**: 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.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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

Test 1 – The evaluator first turned-on discoverable mode and demonstrated the device could be seen by other devices. The evaluator then disabled discovery mode and attempted to scan once again and found that no TOE devices could be discovered and the UI reported that discoverable mode was not allowed.

Test 2 – Not applicable.

Test 3 – Not applicable

Test 4 – Not applicable

Test 5 – Not applicable

Test 6 – Not applicable


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 7 – Not applicable

Test 8 – Not applicable

Test 9 – Not applicable

### 2.5.4 Specification of Management Functions (Wireless LAN) (WLANCEP10:FMT_SMF_EXT.1/WLAN)

#### 2.5.4.1 WLANCEP10:FMT_SMF_EXT.1.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: 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.

Table 2 in Section 3.6 of the Admin Guide details the Common Criteria Related Settings. The table contains descriptions, APIs, and configuration/use of the Common Criteria security functions.

**Component Testing Assurance Activities**: 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.

The evaluator tested this as part of FMT_MOF_EXT.1 where both pre-shared key and EAP-TLS WLAN connections are tested in terms of administrator restrictions and capabilities. Those results show that WLAN client connections support the configuration of CA, security type, authentication protocol, and client credential for specific SSIDs.

### 2.5.5 Specification of Remediation Actions (MDFPP32:FMT_SMF_EXT.2)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.5.5.1 MDFPP32:FMT_SMF_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.5 of the ST states that the TOE offers MDM agents the ability to wipe protected data, wipe sensitive data, remove Enterprise applications, and remove all device stored Enterprise resource data upon un-enrollment. The TOE offers MDM agents the ability to wipe protected data (effectively wiping the device) at any time. Similarly, the TOE also offers the ability to remove Enterprise applications and a full wipe of managed profile data of the TOE'S Enterprise data/applications at any time.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test – The evaluator used the MDM test application to unenroll the device. For wipe upon unenrollment, see test FMT_SMF_EXT.1, Test 7 where the evaluator shows the device can be wiped by and administrator. See FCS_CKM_EXT.5 which shows that the user can wipe the device and FIA_AFL_EXT.1 which shows that the device will be wiped as a result of reaching the maximum number of authentication failures limit. In the case of wiping sensitive data, there is no specific function to do that, but performing a wipe of protected data also results in a wipe of sensitive data. In the cases of both wiping the device and removing Enterprise apps, if an MDM were to unenroll a mobile device it can certainly instruct its agent(s) to perform any available administrator functions immediately prior to the final unenrollment or incidentally resulting in unenrollment (in the case of instructing a wipe). As such Test FMT_SMF_EXT.1, Test 7 and Test FMT_SMF_EXT.1, Test 28 serve to show that the device (protected and sensitive data) could be wiped and enterprise applications can be removed as a result of unenrollment as claimed.

### 2.5.6 Current Administrator (MDFPP32:FMT_SMF_EXT.3)

#### 2.5.6.1 MDFPP32:FMT_SMF_EXT.3.1


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The evaluator shall cause the TOE to be enrolled into management. The evaluator shall then invoke this mechanism and verify the ability to view that the device has been enrolled, view the management functions that the administrator is authorized to perform.

Test 1 – The evaluator enrolled the TOE into management. The evaluator examined the device settings for Device Admin Apps to first identify the enrolled application and then to review the permissions assigned to that app.

## 2.6 Protection of the TSF (FPT)

### 2.6.1 Application Address Space Layout Randomization (MDFPP32:FPT_AEX_EXT.1)

#### 2.6.1.1 MDFPP32:FPT_AEX_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.6.1.2 MDFPP32:FPT_AEX_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states that the Linux kernel of the TOE'S Android operating system provides address space layout randomization utilizing the get_random_int(void) kernel random function to provide eight unpredictable bits to the base address of any user-space memory mapping. The random function, though not cryptographic, ensures that one cannot predict the value of the bits.

**Component Guidance Assurance Activities**: None Defined

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

Test 1: The evaluator shall select 3 apps included with the TSF. These must include any web browser or mail client included with the TSF. For each of these apps, the evaluator 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.

Test 1 – The evaluator had 2 devices of each type to perform this test. The evaluator chose the email, settings, and chrome applications to perform this test. The evaluator started the matching apps on each device and performed a memory map of each device. The evaluator then used a test program to compare the memory locations of each app on each device. The mapping for each app was different on each device.

### 2.6.2 Memory Page Permissions (MDFPP32:FPT_AEX_EXT.2)

#### 2.6.2.1 MDFPP32:FPT_AEX_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states that the TOE utilizes a 5.4 Linux kernel (https://source.android.com/devices/architecture/kernel/modular-kernels#core-kernel-requirements), whose


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory and ensures that write and execute permissions are not simultaneously granted on all memory. The Android operating system (as of Android 2.3) sets the ARM No eXecute (XN) bit on memory pages and the TOE'S ARMv8 Application Processor's Memory Management Unit (MMU) circuitry enforces the XN bits. From Android's documentation (https://source.android.com/devices/tech/security/index.html), Android 2.3 forward supports 'Hardware-based No eXecute (NX) to prevent code execution on the stack and heap'. Section D.5 of the ARMv8 Architecture Reference Manual contains additional details about the MMU of ARM-based processors: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.f/index.html.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.6.3 Stack Overflow Protection (MDFPP32:FPT_AEX_EXT.3)

#### 2.6.3.1 MDFPP32:FPT_AEX_EXT.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall determine that the TSS contains a description of stackbased 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.

Section 6.6 of the ST states that the TOE's Android operating system provides explicit mechanisms to prevent stack buffer overruns in addition to taking advantage of hardware-based No eXecute to prevent code execution on the stack and heap. Specifically, the vendor builds the TOE (Android and support libraries) using gcc-fstack-protector compile option to enable stack overflow protection and Android takes advantage of hardware-based eXecute-Never to make the stack and heap non-executable. The vendor applies these protections to all TSF executable binaries and libraries.

**Component Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: None Defined

### 2.6.4 Domain Isolation (MDFPP32:FPT_AEX_EXT.4)

#### 2.6.4.1 MDFPP32:FPT_AEX_EXT.4.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.6.4.2 MDFPP32:FPT_AEX_EXT.4.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Section 6.6 of the ST states the TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the TOE is a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader and kernels using data fused into the device processor.

The TOE protects its REK by limiting access to only trusted applications within the TEE (Trusted Execution Environment). The TOE key manager includes a TEE module which utilizes the REK to protect all other keys in the key hierarchy. All TEE applications are cryptographically signed, and when invoked at runtime (at the behest of an untrusted application), the TEE will only load the trusted application after successfully verifying its cryptographic signature.

Additionally, the TOE'S Android operating system provides 'sandboxing' that ensures that each third-party mobile application executes with the file permissions of a unique Linux user ID, in a different virtual memory space. This ensures that applications cannot access each other's memory space or files and cannot access the memory space or files of other applications (notwithstanding access between applications with a common application developer).

The TOE has a locked bootloader, which restricts a user to installing a new software image through the Zebra's proscribed OTA (Over The Air) methods. The TOE allows an operator to download and install an OTA update through the system settings (Settings->System->Advanced->System update->Check for update) while the phone is fully booted, or by separately downloading an OTA image, and then "sideloading via ADB" the OTA update from Android's recovery mode. In both cases, the TOE will verify the digital signature of the new OTA before applying the new firmware.

No USSD nor MMI codes are available to be used while the phone is in the locked state. The user can only be presented with a dialer from the lock screen by selecting the "Emergency" button. From this dialer, the user is only allowed to dial a specific set of emergency phone numbers; any attempts to enter a USSD or MMI code results in a pop-up message stating "Can't call. <Phone number> is not an emergency number." and the call is not made/the USSD or MMI code is not submitted.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 – The evaluator traversed all files and directories on the device starting at / and identified those with write permission for the world. The evaluator examined the list and did not find any TSF files in the list. The evaluator used the list in the ST to identify the TSF files.

The evaluator attempted to modify a protected file to show that the TOE file permissions could not be subverted. The evaluator attempted to change the permissions on a *.jar file in /system/framework and was denied. The evaluator attempted to modify the file using vi and was unable to do so (no editor is on the device). The evaluator then tried to overwrite the file and was denied since the evaluator did not have permission.

Test 2 - The TOE does not allow access to system files in any of its boot modes, including auxiliary boot modes, to normal users. The evaluator used an application in the previous test the access while the TOE was fully booted. The TOE does not allow applications to run in any of its other auxiliary boot modes. Although USB debugging is disabled under the normal CC-evaluated configuration, the evaluator re-enabled it to show what is presented to a normal shell on the device while fully booted. The evaluator used ADB to show that system files could not be modified while the device was fully booted. The evaluator then attempted to modify files in the bootloader auxiliary mode, fastboot mode, and recovery mode and was unable to do so. The evaluator concluded that there was no way of modifying the system files while the TOE was in an auxiliary boot mode.

### 2.6.5 Kernel Address Space Layout Randomization (MDFPP32:FPT_AEX_EXT.5)

#### 2.6.5.1 MDFPP32:FPT_AEX_EXT.5.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.6.5.2 MDFPP32:FPT_AEX_EXT.5.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall ensure that the TSS section of the ST describes how the 4 bits are generated and provides a justification as to why those bits are unpredictable.

Section 6.6 of the ST states the TOE models provide Kernel Address Space Layout Randomization (KASLR) as a hardening feature to randomize the location of kernel data structures at each boot, including the core kernel as a random physical address, mapping the core kernel at a random virtual address in the vmalloc area, loading kernel modules at a random virtual address in the vmalloc area, and mapping system memory at a random virtual address in the linear area. The entropy used to dictate the randomization is based on the hardware present within the phone. For ARM devices, such as the TOE, 13–25 bits of entropy are generated on boot, from which the starting memory address is generated.

**Component Guidance Assurance Activities**: None Defined

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

Test 1: The evaluator shall reboot the TOE six times. For each of these reboots, the evaluator shall examine memory mapping locations of the kernel. The evaluator must ensure that for at least five reboots the memory mappings are not placed in the same location on both devices.

The evaluator rebooted the device 6 times, examined the memory mapping locations of the kernel, and verified that for at least 5 of the reboots the memory mappings were not in the same location on both devices.

### 2.6.6 Application Processor Mediation (MDFPP32:FPT_BBD_EXT.1)

#### 2.6.6.1 MDFPP32:FPT_BBD_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: The evaluator shall ensure that the TSS section of the ST describes at a high level how the processors on the Mobile Device interact, including which bus protocols they use to communicate, any other devices operating on that bus (peripherals and sensors), and identification of any shared resources. The evaluator shall verify that the design described in the TSS does not permit any BPs from accessing any of the peripherals and sensors or from accessing main memory (volatile and non-volatile) used by the AP. In particular, the evaluator shall ensure that the design prevents modification of executable memory of the AP by the BP.

Section 6.6 of the ST states the TOE'S hardware and software architecture ensures separation of the application processor (AP) from the baseband or communications processor (CP) through internal controls of the TOE'S SoC, which contains both the AP and the CP. The AP restricts hardware access control through a protection unit that restricts software access from the baseband processor through a dedicated 'modem interface'. The protection unit combines the functionality of the Memory Protection Unit (MPU), the Register Protection Unit (RPU), and the Address Protection Unit (APU) into a single function that conditionally grants access by a master to a software defined area of memory, to registers, or to a pre-decoded address region, respectively. The modem interface provides a set of APIs (grouped into five categories) to enable a high-level OS to send messages to a service defined on the modem/baseband processor. The combination of hardware and software restrictions ensures that the TOE'S AP prevents software executing on the modem or baseband processor from accessing the resources of the application processor (outside of the defined methods, mediated by the application processor).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.6.7 JTAG Disablement (MDFPP32:FPT_JTA_EXT.1)

### 2.6.7.1 MDFPP32:FPT_JTA_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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:


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Section 6.6 of the ST states the TOE'S prevents access to its processor's JTAG interface by requiring use of a signing key to authenticate prior to gaining JTAG access. Only a JTAG image with the accompanying device serial number (which is different for each mobile device) that has been signed by the vendor's private key can be used to access a device's JTAG interface. The private key corresponds to the vendor's RSA 2048-bit public key (a SHA-256 hash of which is fused into the TOE'S application processor).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Section 6.6 of the ST states the TOE'S prevents access to its processor's JTAG interface by requiring use of a signing key to authenticate prior to gaining JTAG access. Only a JTAG image with the accompanying device serial number (which is different for each mobile device) that has been signed by the vendor's private key can be used to access a device's JTAG interface. The private key corresponds to the vendor's RSA 2048-bit public key (a SHA-256 hash of which is fused into the TOE'S application processor).

Test – The test is not applicable as the ST selected 'control access by a signing key' in the requirement.

### 2.6.8 Key Storage (MDFPP32:FPT_KST_EXT.1)

#### 2.6.8.1 MDFPP32:FPT_KST_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Section 6.6 of the ST states The TOE does not store any plaintext key material in its internal Flash; the TOE encrypts all keys before storing them. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery depletes or is removed), all keys stored in the internal Flash are wrapped with a KEK. Please refer to section 6.2 of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash. As the TOE encrypts all keys stored in Flash, upon boot-up, the TOE presents a password authentication screen before any functionality is unlocked. Prior to the user authenticating with the password, all DEKs, stored keys, and data remain encrypted. Upon user authentication, the password is used in conjunction to the REK to decrypt all DEKs, stored keys, and data and they become available for use. Further information about this process can be seen in the FDE Key Hierarchy diagram in the KMD.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

## 2.6.9 No Key Transmission (MDFPP32:FPT_KST_EXT.2)

### 2.6.9.1 MDFPP32:FPT_KST_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states the TOE itself (i.e., the mobile device) comprises a cryptographic module that utilizes cryptographic libraries including BoringSSL, application processor cryptography (which leverages AP hardware), and the following system-level executables that utilize KEKs: vold, wpa_supplicant, and the Android Key Store.

1. vold and QCT's application processor hardware provides Data-At-Rest encryption of the user data partition in Flash
2. wpa_supplicant provides 802.11-2014/WPA2 services
3. the Android Key Store application provides key generation, storage, deletion services to mobile applications and to user through the UI

The TOE ensures that plaintext key material is not exported by not allowing the REK to be exported and by ensuring that only authenticated entities can request utilization of the REK. Furthermore, the TOE only allows the system-level executables access to plaintext DEK values needed for their operation. The TSF software (the system-level executables) protects those plaintext DEK values in memory both by not providing any access to these values and by clearing them when no longer needed (in compliance with FCS_CKM_EXT.4).

**Component Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: None Defined

### 2.6.10 No Plaintext Key Export (MDFPP32:FPT_KST_EXT.3)

#### 2.6.10.1 MDFPP32:FPT_KST_EXT.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states the TOE does not provide any way to export plaintext DEKs or KEKs (including all keys stored in the Android Key Store) as the TOE chains or directly encrypts all KEKs to the REK.

Furthermore, the components of the device are designed to prevent transmission of key material outside the device. Each internal system component requiring access to a plaintext key (for example the Wi-Fi driver) must have the necessary precursor(s), whether that be a password from the user or file access to key in Flash (for example the encrypted AES key used for encryption of the Flash data partition). With those appropriate precursors, the internal system-level component may call directly to the system-level library to obtain the plaintext key value. The system library in turn requests decryption from a component executing inside the trusted execution environment and then directly returns the plaintext key value (assuming that it can successfully decrypt the requested key, as confirmed by the CCM/GCM verification) to the calling system component. That system component will then utilize that key (in the example, the kernel which holds the key in order to encrypt and decrypt reads and writes to the encrypted user data partition files in Flash). In this way, only the internal system components responsible for a given activity have access to the plaintext key needed for the activity, and that component receives the plaintext key value directly from the system library.

For a user's mobile applications, those applications do not have any access to any system-level components and only have access to keys that the application has imported into the Android Key Store. Upon requesting access to a key, the mobile application receives the plaintext key value back from the system library through the Android API. Mobile applications do not have access to the memory space of any other mobile application so it is not possible for a malicious application to intercept the plaintext key value to then log or transmit the value off the device.

**Component Guidance Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: None Defined

### 2.6.11 Self-Test Notification (MDFPP32:FPT_NOT_EXT.1)

#### 2.6.11.1 MDFPP32:FPT_NOT_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

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

Section 6.6 of the ST states that when the TOE encounters a critical failure (either a self-test failure or TOE software integrity verification failure), a failure is message is displayed to the screen, and the TOE attempts to reboot. If the failure persists between boots, the user may attempt to boot to the recovery mode/kernel to wipe data and perform a factory reset in order to recover the device.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test 1 - The evaluator first flashed the TOE with the latest image that was used for testing. When the TOE boots up, it starts the initialization wizard. The evaluator took screenshots to compare the results with results from when the TOE enters a non-operational mode.

The vendor provided a special software image designed to force a power-up self-test error in the TOE's BoringSSL cryptographic library. The evaluator loaded this image onto the TOE, booted the phone, and waited. The phone booted up to the "Powered by Android" screen and stayed there indefinitely.

Next the evaluator used a hex editor to modify the image of one of the phone's partitions (the boot partition). The evaluator then attempted to boot the phone with the corrupted image and found the phone detected the error and refused to boot.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

## 2.6.12 RELIABLE TIME STAMPS (MDFPP32:FPT_STM.1)

### 2.6.12.1 MDFPP32:FPT_STM.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states the TOE requires time for the Package Manager (which installs and verifies APK signatures and certificates), image verifier, wpa_supplicant, and Android Key Store applications. These TOE components obtain time from the TOE using system API calls [e.g., time() or gettimeofday()]. An application (unless a system application is residing in /system/priv-app or signed by the vendor) cannot modify the system time as mobile applications need the Android 'SET_TIME' permission to do so. Likewise, only a process with root privileges can directly modify the system time using system-level APIs. The TOE uses the Cellular Carrier time (obtained through the Carrier's network time server) as a trusted source; however, the user can also manually set the time through the TOE'S user interface. Further, this stored time is used both for the time/date tags in audit logs and is used to track inactivity timeouts that force the TOE into a locked state.

**Component Guidance Assurance Activities**: The evaluator examines the operational guidance to ensure it describes how to set the time.

The "Hardware Control" table entry in Section 3.6 (Common Criteria Related Settings) of the Admin Guide provides the API and settings for automatic time which can be enabled or disabled and allows the device to get time from the network.

**Component Testing Assurance Activities**: 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.

Test 1 - The evaluator turned off Wi-Fi to ensure the clock would not automatically update and examined the current time on the TOE by pressing down on the time in order to also see the date. The evaluator then changed the time and date by unchecking the automatic date/time setting and manually changing the time. The evaluator


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

then re-enabled the automatic data and time setting and the Wi-Fi to cause the time to be reset. The evaluator observed the restored date and time.

### 2.6.13 TSF Cryptographic Functionality Testing (MDFPP32:FPT_TST_EXT.1)

#### 2.6.13.1 MDFPP32:FPT_TST_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

The TOE automatically performs known answer power on self-tests (POST) on its cryptographic algorithms to ensure that they are functioning correctly. Each component providing cryptography (application processor, and BoringSSL) performs known answer tests on their cryptographic algorithms to ensure they are working correctly. Should any of the tests fail, the TOE displays an error message stating “Boot Failure” and halts the boot process, displays an error to the screen, and forces a reboot of the device.

| Algorithm | Implemented in | Description |
|---|---|---|
| AES encryption/decryption | BoringSSL | Comparison of known answer to calculated value |
| ECDH key agreement | BoringSSL | Comparison of known answer to calculated value |
| DRBG random bit generation | BoringSSL | Comparison of known answer to calculated value |
| ECDSA sign/verify | BoringSSL | Comparison of known answer to calculated value |
| HMAC-SHA | BoringSSL | Comparison of known answer to calculated value |
| RSA sign/verify | BoringSSL | Comparison of known answer to calculated value |
| SHA hashing | BoringSSL | Comparison of known answer to calculated value |
| AES encryption/decryption | Application Processor | Comparison of known answer to calculated value |
| HMAC-SHA | Application Processor | Comparison of known answer to calculated value |
| DRBG random bit generation | Application Processor | Comparison of known answer to calculated value |
| SHA hashing | Application Processor | Comparison of known answer to calculated value |
| AES-XTS encrypt/decrypt | Application Processor | Comparison of known answer to calculated value |


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: None Defined

### 2.6.14 TSF Cryptographic Functionality Testing (Wireless LAN) (WLANCEP10:FPT_TST_EXT.1/WLAN)

#### 2.6.14.1 WLANCEP10:FPT_TST_EXT.1.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.6.14.2 WLANCEP10:FPT_TST_EXT.1.2/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

See MDFPP32:FPT_TST_EXT.1.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Guidance Assurance Activities**: 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.

See MDFPP32:FPT_TST_EXT.1 for a description of the actions when a self-test fails.

**Component Testing Assurance Activities**: 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.

Test 1 – Many test results in this report serve to demonstrate that the unmodified TSF software can load without integrity violations and the TOE is fully operational. See Test FPT_NOT_EXT.1 where the evaluator tested integrity.

Test 2 – See the test FPT_NOT_EXT.1 where a corrupted image is used to ensure the corruption is detected.

## 2.6.15 TSF Integrity Checking (Post-Kernel) (MDFPP32:FPT_TST_EXT.2/POSTKERNEL)

### 2.6.15.1 MDFPP32:FPT_TST_EXT.2/POSTKERNEL.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluation activity shall be completed in conjunction with FPT_TST_EXT.2/PREKERNEL for all executable code specified.

See MDFPP32:FPT_TST_EXT.2/PREKERNEL

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: The evaluation activity shall be completed in conjunction with FPT_TST_EXT.2/PREKERNEL for all executable code specified.

See MDFPP32:FPT_TST_EXT.2/PREKERNEL


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

### 2.6.16 TSF Integrity Checking (Pre-Kernel) (MDFPP32:FPT_TST_EXT.2/PREKERNEL)

#### 2.6.16.1 MDFPP32:FPT_TST_EXT.2/PREKERNEL.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states the TOE ensures a secure boot process in which the TOE verifies the digital signature of the bootloader software for the Application Processor (using a public key whose hash resides in the processor's internal fuses) before transferring control. The bootloader, in turn, verifies the signature of the Linux kernel it loads. The TOE performs checking of the entire /system and /vendor partition through use of Android's dm_verity mechanism (and while the TOE will still operate, it will log any blocks/executables that have been modified).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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:


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Tests 1 & 2 – See MDFPP32:FPT_NOT_EXT.1 where screenshots for a correctly booted phone are provide and where the evaluator tested integrity violations and demonstrated the opposite of this behavior.

Test 3 – Not applicable.

### 2.6.17 Trusted Update: TSF Version Query (MDFPP32:FPT_TUD_EXT.1)

#### 2.6.17.1 MDFPP32:FPT_TUD_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.6.17.2 MDFPP32:FPT_TUD_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.6.17.3 MDFPP32:FPT_TUD_EXT.1.3


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: None Defined

**Component Guidance Assurance Activities**: None Defined

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

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

- the current version of the TSF operating system and any firmware that can be updated separately

- the hardware model of the TSF

- the current version of all installed mobile applications

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

Section 6.6 of the ST states the TOE'S user interface provides a method to query the current version of the TOE software/firmware (Android version, baseband version, kernel version, build number, and software version) and hardware (model and version). Additionally, the TOE provides users the ability to review the currently installed apps (including 3rd party 'built-in' applications) and their version.

## 2.6.18 TSF Update Verification (MDFPP32:FPT_TUD_EXT.2)

### 2.6.18.1 MDFPP32:FPT_TUD_EXT.2.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.6.18.2 MDFPP32:FPT_TUD_EXT.2.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.6.18.3 MDFPP32:FPT_TUD_EXT.2.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states the TOE verifies all OTA (Over The Air) updates to the TOE software (which includes baseband processor updates) using a public key chaining ultimately to the Root Public Key, a hardware protected key whose SHA-256 hash resides inside the application processor. Should this verification fail, the software update will fail and the update will not be installed.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

The application processor verifies the bootloader's authenticity and integrity (thus tying the bootloader and subsequent stages to a hardware root of trust: the SHA-256 hash of the Root Public Key, which cannot be reprogrammed after the "write-enable" fuse has been blown).

The Android OS on the TOE requires that all applications bear a valid signature before Android will install the application.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test 1 – The developer provided three updates – one signed, one unsigned, and one signed by a disallowed key. The evaluator loaded the updates on the device and attempted to install each device. The evaluator attempted to install the unsigned update and failed with a signature verification error. The evaluator then attempted to install the disallowed key update and failed. The evaluator attempted to install the signed update and succeeded.

Test 2 – This was tested as part of test 1.

Test 3 – This test is not applicable.

Test 4 – See test case 1. The developer provided updates, including updated files for each processor: AP, Bootloader, Communication Processor, and Carrier Specific Configuration. The integrity check is on the entire update package and not on specific files in the package.

### 2.6.19 Application Signing (MDFPP32:FPT_TUD_EXT.3)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.6.19.1 MDFPP32:FPT_TUD_EXT.3.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.6 of the ST states that android verifies the authenticity of applications by verifying the Android APK signature prior to installing the file (additionally, Android ensures that updates to existing applications use the same signing certificate).

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test 1 – The developer provided the evaluator 2 applications – one signed and one not signed. The evaluator attempted to install the signed application and was successful. The evaluator attempted to install the unsigned application and it failed with a “no certificate” message.

## 2.7 TOE ACCESS (FTA)

### 2.7.1 TSF- AND USER-INITIATED LOCKED STATE (MDFPP32:FTA_SSL_EXT.1)

#### 2.7.1.1 MDFPP32:FTA_SSL_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.7.1.2 MDFPP32:FTA_SSL_EXT.1.2


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.7.1.3 MDFPP32:FTA_SSL_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall verify the TSS describes the actions performed upon transitioning to the locked state.

Section 6.7 of the ST states the TOE transitions to its locked state either immediately after a User initiates a lock by pressing the power button (if configured) or after a (also configurable) period of inactivity, and as part of that transition, the TOE will display a lock screen to obscure the previous contents and play a “lock sound” to indicate the phone’s transition; however, the TOE’S lock screen still displays email notifications, calendar appointments, user configured widgets, text message notifications, the time, date, call notifications, battery life, signal strength, and carrier network. But without authenticating first, a user cannot perform any related actions based upon these notifications (they cannot respond to emails, calendar appointments, or text messages) other than the actions assigned in FIA_UAU_EXT.2.1 (see selections in section 5).

Note that during power up, the TOE presents the user with an unlock screen stating “unlock for all features and data”. While at this screen, the TOE has already decrypted Device Encrypted (DE) files within the user’s data partition, but cannot yet decrypt the user’s Credential Encrypted (CE) files. The user can only access a subset of device functionality before authenticating (e.g. the user can making an emergency call, receive incoming calls, receiving alarms, and any other “direct boot” functionality). After the user enters their password, the TOE decrypts the user’s CE files within the user data partition and the user has unlocked the full functionality of the phone. After this initial authentication, upon (re)locking the phone, the TOE presents the user with the previously mentioned KeyGuard lock screen. While locked, the actions described in FIA_UAU_EXT.2.1 are available for the user to utilize.

**Component Guidance Assurance Activities**: 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.

Section 3.6 (Common Criteria Related Settings) table 2 “Lockscreen” section provides administrators with the APIs and available values that can be used to set the inactivity timeout (Inactivity to lockout) and issue a remote lock (Remote Lock).


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Component Testing Assurance Activities**: 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.

Test 1 – The TOE was alternately configured with 2 and 4 minute session timeout settings and after showing the configured time, the TOE was left inactive for the configured period of time to demonstrate that it locked at that time as expected.

The lockscreen login display shows number of device status indicators and allows the following list of functions to be performed:

- Take screen shots (stored internally)
- Enter password to unlock
- Make/receive emergency calls
- Take pictures (stored internally) - unless the camera was disabled
- Turn the TOE off
- Restart the TOE
- Enable Airplane mode
- See notifications (note that some notifications identify actions, for example to view a screenshot; however, selecting those notifications highlights the password prompt and require the password to access that data)
- Configure sound, vibrate, or mute
- Set the volume (up and down) for ringtone
- Access notification widgets (without authentication):
    - Flashlight toggle
    - Do not disturb toggle
    - Auto rotate toggle
    - Sound (on, mute, vibrate)
    - Night light filter toggle

Test 2 – The evaluator configured the TOE to lock immediately when the display is turned off. The evaluator turned off the display (pressed the power button briefly) to demonstrate that it locked immediately as expected.

### 2.7.2 Default TOE Access Banners (MDFPP32:FTA_TAB.1)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.7.2.1 MDFPP32:FTA_TAB.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The TSS shall describe when the banner is displayed.

Section 6.7 of the ST states the TOE can be configured to display a user-specified message on the Lock screen, and additionally an administrator can also set a Lock screen message using an MDM.

**Component Guidance Assurance Activities**: None Defined

**Component Testing Assurance Activities**: 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.

Test 1 - The evaluator used the MDM application to configure a banner message. The evaluator then pressed the power button to lock the TOE and then pressed it again to initiate an unlock action. The banner was found on the display. The evaluator also configured a custom message and demonstrated it was displayed as a banner.

### 2.7.3 Wireless Network Access (WLANCEP10:FTA_WSE_EXT.1)

#### 2.7.3.1 WLANCEP10:FTA_WSE_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall examine the TSS to determine that it defines SSIDs as the attribute to specify acceptable networks. (TD0470 applied)

Section 6.7 of the ST states that the TOE allows an administrator to specify (through the use of an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect to, the security type, authentication protocol, and the client credentials to be used for authentication. When not enrolled with an MDM, the TOE allows the user to control to which wireless networks the TOE should connect, but does not provide an explicit list of such


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

networks, rather the user may scan for available wireless network (or directly enter a specific wireless network), and then connect. Once a user has connected to a wireless network, the TOE will automatically reconnect to that network when in range and the user has enabled the TOE'S Wi-Fi radio.

**Component Guidance Assurance Activities**: 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. (TD0470 applied)

The table in Section 3.6 of the Admin Guide provides the Common Criteria Related Settings.

- The Wi-Fi Settings entry in the table includes the API for configuring the security policy for each wireless network including setting the WLAN CA certificate, specifying the security type, selecting the authentication protocol and selecting the client credentials

Additionally, section 5 of the Admin Guide describes the restrictions that an administrator can create regarding the list of acceptable Wi-Fi networks.

**Component Testing Assurance Activities**: 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. (TD0470 applied)

During testing for FMT_MOF_EXT.1 the evaluator verified the TOE shows the ability for the administrator to configure Wi-Fi connections and the TOE also shows the ability for the administrator to: 1) restrict the user's ability to modify Wi-Fi configurations and choose Wi-Fi networks, and 2) the administrator's ability to view and remove available Wi-Fi configurations.

These settings together show that the administrator can create and delete (both admin and user created) Wi-Fi configurations (thus specifying allowed connections) and restrict the user's ability to control these connections, preventing connection to unwanted Wi-Fi networks.

## 2.8 TRUSTED PATH/CHANNELS (FTP)

### 2.8.1 BLUETOOTH ENCRYPTION (BT10:FTP_BLT_EXT.1)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

#### 2.8.1.1 BT10:FTP_BLT_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.8.1.2 BT10:FTP_BLT_EXT.1.2

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.8 of the ST states that The TOE provides support for both Bluetooth BR/EDR and Bluetooth LE connections. The TSF uses 128-bit keys to encrypt Bluetooth connections (BR/EDR and LE) and provides no method to configure alternate key sizes and Bluetooth encryption is enabled by default.

**Component Guidance Assurance Activities**: 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).

Bluetooth encryption is enabled by default. No configuration is required by the administrator

**Component Testing Assurance Activities**: 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.

See FTP_BLT_EXT.3/BR and FTP_BLT_EXT.3/LE.

### 2.8.2 Persistence of Bluetooth Encryption (BT10:FTP_BLT_EXT.2)

#### 2.8.2.1 BT10:FTP_BLT_EXT.2.1


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: The evaluator shall verify that the TSS describes the TSF's behavior if a remote device stops encryption while connected to the TOE.

Section 6.8 of the ST states The TOE requires an encrypted connection between itself and another Bluetooth device, and should a remote device stop encryption, the TSF will terminate the connection. The remote device can only attempt to re-establish a new, encrypted channel (and if the connection were no encrypted, the TOE would refuse the connection).

**Component Guidance Assurance Activities**: The evaluator shall verify that the operational guidance describes how to enable/disable encryption (if configurable).

Bluetooth encryption is enabled by default including the key size. No configuration is required by the administrator.

**Component Testing Assurance Activities**: 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 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.

The evaluator established a Bluetooth connection between two devices and observed that encryption was enabled in the packet capture. The evaluator then disabled encryption on one device and demonstrated via a packet capture that the TOE re-established encryption.

### 2.8.3 Bluetooth Encryption Parameters (BR/EDR) - per TD0640 (BT10:FTP_BLT_EXT.3/BR)

#### 2.8.3.1 BT10:FTP_BLT_EXT.3/BR.1


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.8 of the ST states the TOE provides support for both Bluetooth BR/EDR and Bluetooth LE connections. The TSF uses 128-bit keys to encrypt Bluetooth connections (BR/EDR and LE) and provides no method to configure alternate key sizes and Bluetooth encryption is enabled by default.

**Component Guidance Assurance Activities**: The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for BR/EDR encryption, if configurable.

Bluetooth encryption is enabled by default including the key size. No configuration is required by the administrator.

**Component Testing Assurance Activities**: 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 – The developer provided a tool to enable BTLE advertising on the test devices (since that is not normally possible). The evaluator enabled Bluetooth snooping on the TOE device and also enabled BTLE advertising using the provided tool. With the devices advertising BTLE, the evaluator used a non-test phone (which has a default max key size of 16 for both BTLE and BT/BR) to pair with each of the test devices in turn for both BTLE and BT/BR. In each case, the evaluator was able to identify the negotiated Linkkey was 16 bytes in length as expected.

Test 2 - Not applicable - the TOE does not support changing the Bluetooth/BR key size.

Test 3 – The developer provided a test device that always requires a 1-byte link key. The TOE failed on the attempt to connect with a 1-byte key.

### 2.8.4 Bluetooth Encryption Parameters (LE) (BT10:FTP_BLT_EXT.3/LE)

#### 2.8.4.1 BT10:FTP_BLT_EXT.3/LE.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.8 of the ST states the TOE provides support for both Bluetooth BR/EDR and Bluetooth LE connections. The TSF uses 128-bit keys to encrypt Bluetooth connections (BR/EDR and LE) and provides no method to configure alternate key sizes.

**Component Guidance Assurance Activities**: The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for LE encryption, if configurable.

Bluetooth encryption is enabled by default including the key size. No configuration is required by the administrator.

**Component Testing Assurance Activities**: 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:


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Test 1 - See BT10:FTP_BLT_EXT/BR where both Bluetooth BLE and BT/BR were tested together. The Bluetooth packet captures in that test were examined to see that the BTLE long term keys were 16 bytes as expected in each case.

Test 2 - Not applicable - the TOE does not support changing the Bluetooth/LE key size.

Test 3 - The developer provided a test device that always requires a 1-byte link key and advertises Bluetooth/LE. The TOE failed on the attempt to connect with a 1-byte key.

### 2.8.5 Trusted Channel Communication (MDFPP32:FTP_ITC_EXT.1)

#### 2.8.5.1 MDFPP32:FTP_ITC_EXT.1.1

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.8.5.2 MDFPP32:FTP_ITC_EXT.1.2


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.8.5.3 MDFPP32:FTP_ITC_EXT.1.3

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

Section 6.8 in the ST states that the TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of IEEE 802.11-2012, 802.1X, and EAP-TLS and TLS, HTTPS. The TOE permits itself and applications to initiate communicate via the trusted channel, and the TOE initiates communications via the WPA2 (IEEE 802.11-2012, 802.1X with EAP-TLS) trusted channel for connection to a wireless access point. The TOE provides mobile applications and MDM agents access to HTTPS and TLS via published APIs, thus facilitating administrative communication and configured enterprise connections. These APIs are accessible to any application that needs an encrypted end-to-end trusted channel. The evaluator confirmed that all the protocols listed in the TSS are specified and included in the FTP_ITC_EXT.1 and FTP_ITC_EXT.1/WLAN requirements in the ST.

**Component Guidance Assurance Activities**: The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to access points, VPN Gateways, and other trusted IT products.

Section 2.5 of the Admin Guide details VPN connectivity. Section 5 and 6 describe Wi-Fi and VPN configuration.

**Component Testing Assurance Activities**: The evaluator shall also perform the following tests for each protocol listed:


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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 (and the TSF includes a native VPN client), the evaluator shall ensure that the TOE is able to initiate communications with a VPN Gateway, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 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.

Test 1 - See the Test Cases associated with FCS_TLSC_EXT.1 where many TLS/HTTPS connections are made. See Test Case FCS_CKM.1/WLAN test case 2 where Wi-Fi encryption is demonstrated. For all protocol-based tests, packet captures were collected and no plaintext was observed in the traffic.

Test 2 – Not applicable.

Test 3 – Not applicable. OTA updates are not claimed.

Test 4 – See Test Case FTP_ITC_EXT.1/WLAN test case 1 for 802.11-2012 PSK and EAP-TLS connections. See Test Case FCS_TLSC_EXT.1.1 test case 1 where TLS/HTTPS connections are made.

### 2.8.6 Trusted Channel Communication (Wireless LAN) (WLANCEP10:FTP_ITC_EXT.1/WLAN)

#### 2.8.6.1 WLANCEP10:FTP_ITC_EXT.1.1/WLAN

**TSS Assurance Activities**: None Defined

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

#### 2.8.6.2 WLANCEP10:FTP_ITC_EXT.1.2/WLAN

**TSS Assurance Activities**: None Defined


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Guidance Assurance Activities**: None Defined

**Testing Assurance Activities**: None Defined

**Component TSS Assurance Activities**: 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.

This was addressed in the description for MDFPP32:FTP_ITC_EXT.1.

**Component Guidance Assurance Activities**: 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.

As indicated previously in this report, the Admin Guide specifies the path for configuration of Wi-Fi, CA certificates, EAP-TLS, and client credentials. The Admin Guide also contains detailed information about the corresponding APIs. Section 5 (Wi-Fi Configuration) in the Admin Guide states that if a Wi-Fi connection unintentionally terminates, the end user will need to reconnect to re-establish the session.

**Component Testing Assurance Activities**: 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.

Test 1 - See Test Case FTP_ITC_EXT.1/WLAN test case 4 where Access Point connections are made with a pre-shared key. See also WLANCEP10:FCS_TLSC_EXT.1/WLAN test case 1 where several Access Point connections are made using EAP-TLS.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

Test 2 – See FCS_CKM.1/WLAN test case 2 where it can be seen that packets are protected using 802.11 once a connection is made with EAPOL based on a pre-shared key. Additionally, a wireless packet capture was provided showing the 802.11 packets in the air are encrypted once a connection is made with EAPOL based on an EAP-TLS.

Test 3 – The evaluator modified traffic to the access point and observed in the packet capture that the traffic was dropped.

Test 4 – The evaluator connected to an access point and then interrupted power to the access point. When the power was restored to the access point, the TOE and access point re-created a secure channel.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

# 3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

## 3.1 Development (ADV)

### 3.1.1 Basic Functional Specification (ADV_FSP.1)

**Assurance Activities**: 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, 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.

## 3.2 Guidance documents (AGD)

### 3.2.1 Operational User Guidance (AGD_OPE.1)

**Assurance Activities**: Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements 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.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

Section 3.4 (Cryptographic Module Identification) in the Admin Guide identifies the cryptographic components of the TOE which provide the CAVP certified algorithms. This section states that the use of other cryptographic components beyond those listed was neither evaluated nor tested during the TOE's Common Criteria evaluation. No additional configuration is needed for the cryptographic modules in order to be compliant.

Section 7 (Secure Update Process) in the Admin Guide states that Over the Air (OTA) updates (which includes baseband processor updates) use a public key chaining ultimately to the Root Public Key, a hardware protected key whose SHA-256 hash resides inside the application processor. Should this verification fail, the software update will fail and the update will not be installed. Additionally, the devices also provide roll-back protection for OTA updates to prevent a user from installing a prior/previous version of software by check. The user will get a notification when an update is made available. No special configuration is required to ensure a secure update process.

Section 3.1 (Entering into Common Criteria State) in the Admin Guide provides the settings which must be configured to put the device into Common Criteria Mode. This includes enabling certain features and disabling other features that are not include in the evaluated configuration.

Section 3.6 (Common Criteria Related Settings) in the Admin Guide describes the security settings available in the evaluated configuration to the user and/or administrator.

### 3.2.2 Preparative Procedures (AGD_PRE.1)

**Assurance Activities**: As indicated in the introduction above, there are significant expectations with respect to the documentationâ€"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.

Section 1.1 (Equivalent Devices) in the Admin Guide identifies the evaluated devices included in the evaluated configuration. Section 3.1 explains how to configure the devices into Common Criteria Mode.

## 3.3 Life-cycle support (ALC)

### 3.3.1 Labeling of the TOE (ALC_CMC.1)

**Assurance Activities**: The evaluator shall check the ST to ensure that it contains an identifier 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same versions. The evaluator checked the TOE version during testing by examining the actual devices used for testing.

### 3.3.2 TOE CM Coverage (ALC_CMS.1)

**Assurance Activities**: 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.

See section 3.3.1 above for an explanation of how all CM items are identified. In regards to development environments, developers who wish to develop apps for the devices can go to http://developer.android.com/index.html

### 3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

**Assurance Activities**: The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall verify that this description addresses the TOE OS, the firmware, and bundled applications, each. The evaluator shall also verify that, in addition to the TOE developer's process, any carrier or other third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days.

The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator shall verify that the description includes where users can seek information about the availability of new security updates including details of the specific public vulnerabilities corrected by each update. The evaluator shall verify that the description includes the minimum amount of time that the TOE is expected to be supported with security updates, and the process by which users can seek information about when the TOE is no longer expected to receive security updates.

Section 6.6 of the ST states that Google (the parent developer of Android) supports a bug filing system for the Android OS outlined here: https://source.android.com/setup/contribute/report-bugs. This allows developers or users to search for, file, and vote on bugs that need to be fixed. This helps to ensure that all bugs that affect large numbers of people get pushed up in priority to be fixed.

The vendor also supports their own form of bug reporting, via their website: zebra.com/us/en/about-zebra/contact-zebra/contact-tech-support.html.

Google publishes monthly security updates which the vendor reviews and implements on their devices, releasing as a part of their own monthly security update cycle. Once updates are available, they are immediately made available on Zebra's website here: https://www.zebra.com/us/en/support-downloads.html.

## 3.4 Tests (ATE)

### 3.4.1 Independent Testing - Conformance (ATE_IND.1)

**Assurance Activities**: 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.

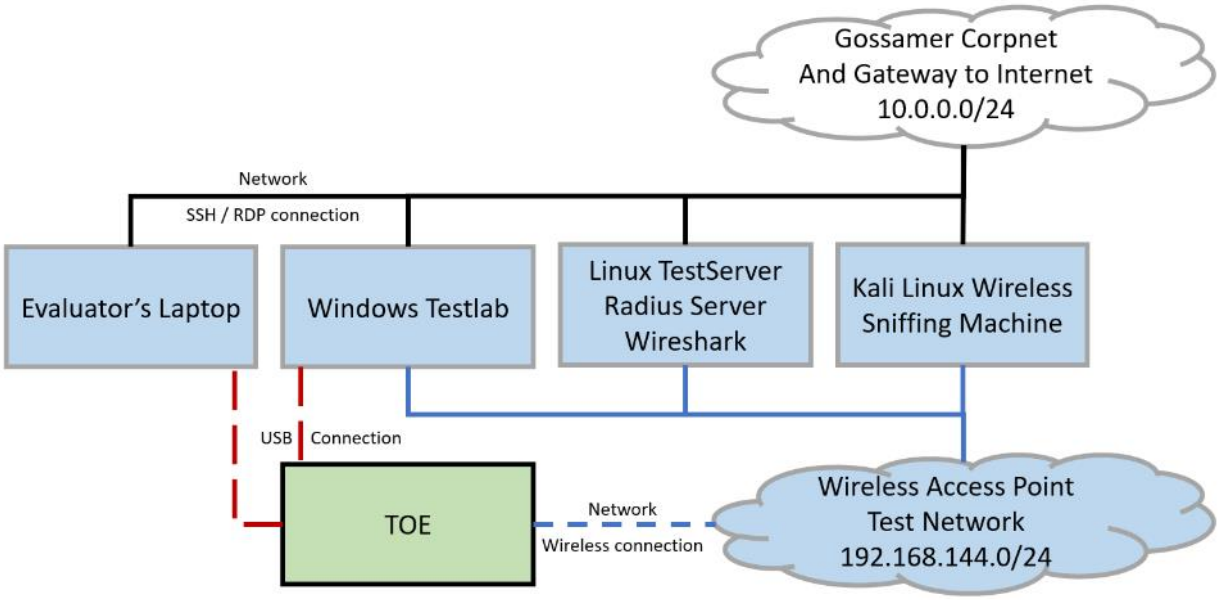
The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test 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


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

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.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The following diagram indicates the test environment.




The evaluator used the following test tools: Windows, Linux, Putty, Wireshark, Freeradius, OpenSSL, web server, adb, HxD, strongswan, tcpdump, micro-httpd, and Gossamer and Samsung developed test programs.

## 3.5 Vulnerability assessment (AVA)

### 3.5.1 Vulnerability Survey (AVA_VAN.1)


© 2021 Gossamer Security Solutions, Inc.
All rights reserved.

**Assurance Activities**: 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.

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities. None of the public search for vulnerabilities uncovered any residual vulnerability.

The evaluator searched the National Vulnerability Database (https://web.nvd.nist.gov/vuln/search), Vulnerability Notes Database (http://www.kb.cert.org/vuls/) on 04/26/2023 with the following search terms: "Android", "Android 11", "BoringSSL", "System Call Policy Engine", "Android Locksettings service KBKDF", "QTI Crypto Engine Core", "QTI Inline Crypto Engine", "QTI Random Number Generator", "Zebra", "ET40", "ET45", "ET40HC", "ET45HC", None of the public search for vulnerabilities uncovered any residual vulnerability and all reported vulnerabilities have been patched.



© 2021 Gossamer Security Solutions, Inc.
All rights reserved.