Cyber Reliant Mobile Data Defender for Android SDK version 4.0 Security Target

Version 0.6
03/16/2023

Prepared for:

Cyber Reliant Corp

National Security & Justice Group

180 Admiral Cochrane Drive Suite 310
Annapolis, MD 21401 U.S.A.

Prepared By:

Gossamer Laboratories

www.gossamersec.com

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE), including ST conventions, ST conformance claims, and the ST organization. The TOE is the Cyber Reliant Mobile Data Defender for Android SDK. The TOE is being evaluated as a file level encryption software application.

The Security Target contains the following additional sections:

- Conformance Claims (Section 2)
- Security Objectives (Section 3)
- Extended Components Definition (Section 4)
- Security Requirements (Section 5)
- TOE Summary Specification (Section 6)

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In the ST, iteration is indicated by a parenthetical number placed at the end of the component. For example FDP_ACC.1(1) and FDP_ACC.1(2) indicate that the ST includes two iterations of the FDP_ACC.1 requirement.
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").
- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.1 Security Target Reference

ST Title – Cyber Reliant Mobile Data Defender for Android SDK Version 4.0 Security Target

ST Version – Version 0.6

ST Date – 03/16/2023

1.2 TOE Reference

TOE Identification – Cyber Reliant Mobile Data Defender for Android SDK Version 4.0

TOE Developer – Cyber Reliant Corp

Evaluation Sponsor – Cyber Reliant Corp

1.3 TOE Overview

The Target of Evaluation (TOE) is the Cyber Reliant Mobile Data Defender for Android SDK Version 4.0 software application package residing on evaluated mobile devices running Android 11. The TOE is a software solution providing the capability to handle file encryption on mobile devices. The TOE is capable of running on Android 11 devices under VID11160, VID11211, or 11315 the same application supporting any of the devices. The TOE was tested on the following mobile devices representing one device from each VID.

Device Name	Chipset/CPU	Architecture	Android Version
Samsung S20 (VID11160)	Snapdragon 865	A64	11
Samsung Tab Active 3 (VID11211)	Exynos 9810	A64	11
Panasonic ToughBook FZN1 (VID11315)	Snapdragon 660 (SDM660)	A64	11

The devices above were identified based on availability for testing, however these devices were previously evaluated to exhibit the same behavior from a security function standpoint and any device could have been used in place for testing. Since the TOE is the same for the remaining devices under these evaluations, all other devices from these evaluations are claimed as equivalent.

VID11160

- Samsung S21 (S21 5G / S21+ 5G / S21 Ultra 5G)
- Samsung S21 (S21 5G / S21+ 5G / S21 Ultra 5G / S21 5G FE / Z Fold3 5G / Z Flip3 5G)
- Samsung S20 (S20 5G / S20+ 5G / S20 Ultra 5G / S20 LTE 5G / S20 5G FE / Note20 5G / Note20 LTE / Note20 Ultra LTE / Note20 Ultra 5G)
- Samsung S20 (S20+ 5G / S20 FE / S20 Ultra 5G / Z Flip 5G / Tab S7 / Tab S7+ / Note20 5G / Note20 Ultra 5G / Z Fold2 5G)
- Samsung XCover Pro / Samsung A51
- Samsung Note10 (Note10+ 5G / Note10+ / Note 10 5G / Note 10)
- Samsung S10e (S10+ / S10 5G / S10)
- Samsung S10+_(Note10+ 5G / Note10+ / Note 10 / Tab S6 / S10 5G / S10 / S10e / Fold 5G / Fold / Z Flip)

VID11211

- Samsung A52 5G (A52 5G / A42 5G)
- Samsung A71 5G (A71 5G / A51 5G)
- Samsung Tab Active 3

VID11315

- Panasonic ToughBook FZN1 (FZN1 / FZS1 / FZA3)

1.4 TOE Description

The Cyber Reliant Mobile Data Defender for Android SDK Version 4.0 provides file level encryption through an Android Package Kit (APK) and a library implementation. The library contains both Java and native (C/C++) interfaces in order to support the majority of Android application storage requirements. The same implementation and functionality for both java and C/C++ are provided by the TOE. The library offers two groups of Application Programming Interface (API): one set to manipulate files and one set to manipulate SQLite databases. These libraries are:

AuthenticationLib	Used by the library to talk to the Management service for key management.
CRCsqlite3	Our new SQLite library that is completely file based and calls the FileEncryptionLib-arm library directly
FileEncryptionLib-arm	This is the actual File encryption library which is called for file I/O by the application. It manages all interface with the application.

DataEncryptionLib-arm	This is our library that does the actual data encryption and splitting. It is our core library.

While the API groups provide different abstractions for the read and write operations, they are ultimately simply reading and writing a single file. The library is providing file level encryption.

The Management Service application is a straight Java Data Protection SDK APK, while the Library is intended to be included into a mobile application (and then the mobile application can use the API libraries). The Management Service Application runs in the background and uses WolfCrypt keystores to provide the File Encryption Key Encryption Key (FEKEK) to each of the applications. The Data Protection SDK uses the Android keystore to generate and store an RSA key pair used by the Management Service. On a per application basis the Android keystore is leveraged to store each application's RSA keypair to double wrap the AES-wrapped FEKEK. The single and double wrapped FEKEKs are then stored in WolfCrypt secure keystores. The Management Service application handles necessary authentication and key management. The file level encryption suite is an API designed to support the use of specialized file level encryption for Android applications. Encryption is provided by the Cyber Reliant Mobile Data Defender for Android SDK with WolfCrypt.

1.4.1 TOE Architecture

The TOE is software installed on an evaluated mobile device running Android 11. The TOE software is installed as a Management Service as well as the security functionality (TSF) interface library that is compiled into other applications. References to applications noted in this Security Target are regarded as applications that are compiled with the TSF interface API library. The Management Service is responsible for handling the File Encryption Key Encryption Keys (FEKEKs) necessary to unwrap the FEK. The Management Service obtains the Cyber Reliant Mobile Data Defender for Android SDK password (hereafter referred to as the DaR password) from the user and double wraps the FEKEK by using RSA-2048 first and then wrapping it again using AES-256. Note that the Management Service itself is not responsible for encryption.

The TOE's interface library is compiled into another application's package which allows the other application to invoke the TOE's services (i.e. allows the application to call the TOE's file encryption services once the application is registered with the Management Service). Applications registered to the TOE have a unique RSA public/private keypair to enable the applications to pass their RSA public keys to the Management Service along with a Cyber Reliant Mobile Data Defender for Android SDK's certificate fingerprint; the application uses this as the password to the application's key store. Android's keystore protects keys by storing them in a container with limited access to the keys through Android's keystore API. The TOE allows only a single user at a time.

The TOE stores the double wrapped FEKEKs in the Management Service's WolfCrypt keystore and the single wrapped FEKEKs in the application's specific WolfCrypt keystore. The keys are protected by requiring a password to load both the Management Service and application's keystore. In order for other applications to access its FEK, the application must use the TOE's interface library API to request Management Service functions. The Management Service uses the application's public key to wrap the FEKEK (via RSA-OAEP) so that it can be passed to the application by placing the single wrapped FEKEK into the application's WolfCrypt keystore. The wrapped FEKEKs in each application's WolfCrypt keystore are ephemeral to enable a safeguard as a configurable threshold for consecutive incorrect password attempts. When the password attempt threshold is met, the Cyber Reliant system automatically initiates an authentication timeout.

The TOE uses the WolfCrypt and WolfSSL modules in conjunction with our CRC Encryption and Shredding engine for cryptographic services.

During evaluation testing, Gossamer tested the Cyber Reliant Mobile Data Defender for Android SDK on the Samsung S20, Samsung Tab Active 3, and Panasonic ToughBook FZN1.

1.4.1.1 Physical Boundaries

The physical boundary of the TOE is the physical perimeter of the evaluated device (Samsung S20, Samsung Tab Active 3, Panasonic ToughBook FZN1, or equivalent device as identified in section 1.3) on which the TOE resides.

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by the Cyber Reliant Mobile Data Defender for Android SDK:

- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Privacy
- Protection of the TSF
- Trusted path/channels

1.4.1.2.1 Cryptographic support

The evaluated platform runs on Android 11 operating system. Android APIs allow generation of keys through Key Generator, and random numbers are generated using Java SecureRandom (256 bits). Keys are used to protect data belonging to the applications that use the TOE.

The TOE uses the Cyber Reliant Mobile Data Defender for Android SDK with the WolfCrypt Module for cryptographic algorithms. The module supports encryption via AES and random number generation via an SP 800-90 AES-256 DRBG. The TOE also performs AES key wrapping and keyed hashing via HMAC.

1.4.1.2.2 User data protection

The TOE protects user data by providing encryption services for applications to encrypt their data. The TOE allows encryption of data using AES-256 bit keys.

1.4.1.2.3 Identification and authentication

The TOE authenticates applications by requiring a PIN/passphrase to unlock the application's file encryption key. A wrong password results in the unsuccessful loading of the application's WolfCrypt keystore. Without the correct keystore, the application cannot load the keys necessary for file encryption/decryption.

1.4.1.2.4 Security management

The TOE's services/options are inaccessible until a configuration has been created. The TOE does not allow invocation of its services without configuration of the TOE's settings upon first start up. The TOE allows password changes for management purposes.

1.4.1.2.5 Privacy

The TOE does not transmit Personally Identifiable Information (PII) over any network.

1.4.1.2.6 Protection of the TSF

The TOE uses the physical boundary of the evaluated platform as well as the Android operating system for the protection of the TOE's application components.

The TOE checks for updates by selecting the check current version option on its menu. If an update is needed, Cyber Reliant shall deliver, via email or other agreed upon method, an updated application. The TOE's software is digitally signed by Cyber Reliant. Each update is accompanied by documentation outlining changes to the overall service, as well as compatible versions of the Cyber Reliant API.

1.4.1.2.7 Trusted path/channels

The TOE does not transmit any data between itself and another product. All TOE-managed data resides on the evaluated platform.

1.4.2 TOE Documentation

Cyber Reliant offers documents that describe the operation and maintenance for the TOE. The following list of documents was examined as part of the evaluation.

[AGD] User Guide Cyber Reliant Defender Installation and Use, March 2023

2. Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017.
    - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017.
    - Part 3 Extended
- PP-Configuration for Application Software and File Encryption, Version 1.1, 7 April 2019 (CFG_APP-FE_V1.1)
    - The PP-Configuration includes the following components:
        - Base-PP: Protection Profile for Application Software, Version 1.4 (PP_APP_V1.4)
        - PP-Module: PP-Module for File Encryption, Version 1.0 (MOD_FE_V1.0)
- Technical Decisions

Package	Technical Decision	Applied	Notes
PP_APP_V1.4	TD0719 – ECD for PP APP V1.3 and 1.4	Yes	
PP_APP_V1.4	TD0717 – Format changes for PP_APP_V1.4	Yes	
PP_APP_V1.4	TD0709 – Number of elements for iterations of FCS_HTTPS_EXT.1	No	Requirement not claimed
PP_APP_V1.4	TD0669 - FIA_X509_EXT.1 Test 4 Interpretation	No	Requirement not claimed
PP_APP_V1.4	TD0664 - Testing activity for FPT_TUD_EXT.2.2	Yes	
PP_APP_V1.4	TD0655 - Mutual authentication in FTP_DIT_EXT.1 for SW App	Yes	
PP_APP_V1.4/ MOD_FE_V1.0	TD0650 - Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4	No	Not a VPN Client
MOD_FE_V1.0	TD0644 - FCS_CKM_EXT.4 test applicability	Yes	
PP_APP_V1.4	TD0628 - Addition of Container Image to Package Format	Yes	
PP_APP_V1.4	TD0624 - Addition of DataStore for Storing and Setting Configuration Options	Yes	
MOD_FE_V1.0	TD0600 - Conformance claim sections updated to allow for MOD_VPNC_V2.3	No	Not a VPN Client
MOD_FE_V1.0	TD0472 - File Encryption SFR Rationale and Consistency of TOE Type added	Yes	
MOD_FE_V1.0	TD0455 - NIST SP800-133 keygen methods for FAK/FEK Generation	Yes	

2.1 Conformance Rationale

The ST conforms to the PP_APP_V1.4/MOD_FE_V1.0. As explained previously, the security problem definition, security objectives, and security requirements have been drawn from the PP.

3. Security Objectives

The Security Problem Definition may be found in the PP_APP_V1.4/MOD_FE_V1.0 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The PP_APP_V1.4/MOD_FE_V1.0 offers additional information about the identified security objectives, but that has not been reproduced here and the PP_APP_V1.4/MOD_FE_V1.0 should be consulted if there is interest in that material.

In general, the PP_APP_V1.4/MOD_FE_V1.0 has defined Security Objectives appropriate for a file encryption application and as such are applicable to the Cyber Reliant Mobile Data Defender for Android SDK TOE.

3.1 Security Objectives for the Operational Environment

OE.AUTHORIZATION_FACTOR_STRENGTH An authorized user will be responsible for ensuring that all externally derived authorization factors have sufficient strength and entropy to reflect the sensitivity of the data being protected. This can apply to password- or passphrase-based, ECC CDH, and RSA authorization factors.

OE.PLATFORM The TOE relies upon a trustworthy computing platform for its execution. This includes the underlying operating system and any discrete execution environment provided to the TOE.

OE.POWER_SAVE The non-mobile operational environment must be configurable so that there exists at least one mechanism that will cause the system to enter a safe power state (A.SHUTDOWN). Any such mechanism (e.g., sleep, hibernate) that does not conform to this requirement must be capable of being disabled. The mobile operational environment must be configurable such that there exist at least one mechanism that will cause the system to lock upon a period of time

OE.PROPER_ADMIN The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

OE.PROPER_USER The user of the application software is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.

OE.STRONG_ENVIRONMENT_CRYPTO The Operating environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE.

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the PP_APP_V1.4/MOD_FE_V1.0. The PP_APP_V1.4/MOD_FE_V1.0 defines the following extended requirements and since they are not redefined in this ST the PP_APP_V1.4/MOD_FE_V1.0 should be consulted for more information in regard to those CC extensions.

Extended SFRs:

- PP_APP_V1.4:FCS_CKM_EXT.1: Cryptographic Key Generation Services
- MOD_FE_V1.0:FCS_CKM_EXT.2: File Encryption Key (FEK) Generation
- MOD_FE_V1.0:FCS_CKM_EXT.3: Key Encrypting Key (KEK) Support
- MOD_FE_V1.0:FCS_CKM_EXT.4: Cryptographic Key Destruction
- MOD_FE_V1.0:FCS_CKM_EXT.6: Cryptographic Password/Passphrase Conditioning
- MOD_FE_V1.0:FCS_IV_EXT.1: Initialization Vector Generation
- MOD_FE_V1.0:FCS_KYC_EXT.1: Key Chaining and Key Storage
- PP_APP_V1.4:FCS_RBG_EXT.1: Random Bit Generation Services
- PP_APP_V1.4:FCS_RBG_EXT.2: Random Bit Generation from Application
- PP_APP_V1.4:FCS_STO_EXT.1: Storage of Credentials
- MOD_FE_V1.0:FCS_VAL_EXT.1: Validation
- MOD_FE_V1.0:FCS_VAL_EXT.2: Validation Remediation
- PP_APP_V1.4:FDP_DAR_EXT.1: Encryption Of Sensitive Application Data
- PP_APP_V1.4:FDP_DEC_EXT.1: Access to Platform Resources
- PP_APP_V1.4:FDP_NET_EXT.1: Network Communications
- MOD_FE_V1.0:FDP_PM_EXT.1: Protection of Data in Power Managed States
- MOD_FE_V1.0:FDP_PRT_EXT.1: Protection of Selected User Data
- MOD_FE_V1.0:FDP_PRT_EXT.2: Destruction of Plaintext Data
- MOD_FE_V1.0:FIA_AUT_EXT.1: Subject Authorization
- PP_APP_V1.4:FMT_CFG_EXT.1: Secure by Default Configuration
- PP_APP_V1.4:FMT_MEC_EXT.1: Supported Configuration Mechanism
- PP_APP_V1.4:FPR_ANO_EXT.1: User Consent for Transmission of Personally Identifiable
- PP_APP_V1.4:FPT_AEX_EXT.1: Anti-Exploitation Capabilities
- PP_APP_V1.4:FPT_API_EXT.1: Use of Supported Services and APIs
- PP_APP_V1.4:FPT_IDV_EXT.1: Software Identification and Versions
- MOD_FE_V1.0:FPT_KYP_EXT.1: Protection of Keys and Key Material
- PP_APP_V1.4:FPT_LIB_EXT.1: Use of Third Party Libraries
- PP_APP_V1.4:FPT_TUD_EXT.1: Integrity for Installation and Update
- PP_APP_V1.4:FPT_TUD_EXT.2: Integrity for Installation and Update
- PP_APP_V1.4:FTP_DIT_EXT.1: Protection of Data in Transit

Extended SARs:

- ALC_TSU_EXT.1: Timely Security Updates

5. Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

The SFRs have all been drawn from the PP_APP_V1.4/MOD_FE_V1.0. The refinements and operations already performed in the PP_APP_V1.4/MOD_FE_V1.0 are not identified (e.g., highlighted) here, rather the requirements have been copied from the PP_APP_V1.4/MOD_FE_V1.0 and any residual operations have been completed herein. Of particular note, the PP_APP_V1.4/MOD_FE_V1.0 made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that PP should be consulted to identify those changes if necessary.

The SARs are also drawn from the PP_APP_V1.4/MOD_FE_V1.0 The PP_APP_V1.4/MOD_FE_V1.0 should be consulted for the assurance activity definitions.

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by Mobile Data Defender for Android SDK TOE.

Requirement Class	Requirement Component
FCS: Cryptographic support	PP_APP_V1.4:FCS_CKM_EXT.1: Cryptographic Key Generation Services
	PP_APP_V1.4:FCS_CKM.1/AK: Cryptographic Asymmetric Key Generation - per TD0659
	PP_APP_V1.4: FCS_CKM.1.1/SK: Cryptographic Symmetric Key Generation
	MOD_FE_V1.0:FCS_CKM_EXT.2: File Encryption Key (FEK) Generation
	MOD_FE_V1.0:FCS_CKM_EXT.3: Key Encrypting Key (KEK) Support
	MOD_FE_V1.0:FCS_CKM_EXT.4: Cryptographic Key Destruction
	MOD_FE_V1.0:FCS_CKM_EXT.6: Cryptographic Password/Passphrase Conditioning
	PP_APP_V1.4:FCS_COP.1/SKC: Cryptographic Operation - Encryption/Decryption
	PP_APP_V1.4:FCS_COP.1/Hash: Cryptographic Operation - Hashing
	PP_APP_V1.4:FCS_COP.1/KeyedHash: Cryptographic Operation - Keyed-Hash Message Authentication - per TD0626
	MOD_FE_V1.0:FCS_COP.1(5): Cryptographic operation (Key Wrapping)
	MOD_FE_V1.0:FCS_COP.1(6): Cryptographic operation (Key Transport)
	MOD_FE_V1.0:FCS_IV_EXT.1: Initialization Vector Generation
	MOD_FE_V1.0:FCS_KYC_EXT.1: Key Chaining and Key Storage
	PP_APP_V1.3:FCS_RBG_EXT.1: Random Bit Generation Services
	PP_APP_V1.4:FCS_RBG_EXT.2: Random Bit Generation from Application
	PP_APP_V1.4:FCS_STO_EXT.1: Storage of Credentials
	MOD_FE_V1.0:FCS_VAL_EXT.1: Validation
	MOD_FE_V1.0:FCS_VAL_EXT.2: Validation Remediation
FDP: User data protection	PP_APP_V1.4:FDP_DAR_EXT.1: Encryption Of Sensitive Application Data
	PP_APP_V1.4:FDP_DEC_EXT.1: Access to Platform Resources
	PP_APP_V1.4:FDP_NET_EXT.1: Network Communications

	MOD_FE_V1.0:FDP_PM_EXT.1: Protection of Data in Power Managed States
	MOD_FE_V1.0:FDP_PRT_EXT.1: Protection of Selected User Data
	MOD_FE_V1.0:FDP_PRT_EXT.2: Destruction of Plaintext Data
	MOD_FE_V1.0:FDP_PRT_EXT.3: Protection of Third-Party Data
FIA: Identification and authentication	MOD_FE_V1.0:FIA_AUT_EXT.1: Subject Authorization
FMT: Security management	PP_APP_V1.4:FMT_CFG_EXT.1: Secure by Default Configuration
	PP_APP_V1.4:FMT_MEC_EXT.1: Supported Configuration Mechanism - per TD0624
	PP_APP_V1.4:FMT_SMF.1: Specification of Management Functions
	MOD_FE_V1.0:FMT_SMF.1(2): Specification of File Encryption Management Functions
FPR: Privacy	PP_APP_V1.4:FPR_ANO_EXT.1: User Consent for Transmission of Personally Identifiable
FPT: Protection of the TSF	PP_APP_V1.4:FPT_AEX_EXT.1: Anti-Exploitation Capabilities
	PP_APP_V1.4:FPT_API_EXT.1: Use of Supported Services and APIs
	PP_APP_V1.4:FPT_IDV_EXT.1: Software Identification and Versions
	MOD_FE_V1.0:FPT_KYP_EXT.1: Protection of Keys and Key Material
	PP_APP_V1.4:FPT_LIB_EXT.1: Use of Third Party Libraries
	PP_APP_V1.4:FPT_TUD_EXT.1: Integrity for Installation and Update
	PP_APP_V1.4:FPT_TUD_EXT.2: Integrity for Installation and Update - per TD0664
FTP: Trusted path/channels	PP_APP_V1.4:FTP_DIT_EXT.1: Protection of Data in Transit - per TD0655

Table 1 TOE Security Functional Components

5.1.1 Cryptographic support (FCS)

5.1.1.1 Cryptographic Key Generation Services (PP_APP_V1.4:FCS_CKM.1)

PP_APP_V1.4:FCS_CKM_EXT.1.1

The application shall [***invoke platform-provided functionality for asymmetric key generation***].

5.1.1.2 Cryptographic Asymmetric Key Generation - per TD0659 (PP_APP_V1.4:FCS_CKM.1/AK)

PP_APP_V1.4: FCS_CKM.1.1/AK

The application shall [***invoke platform-provided functionality***] to generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [***RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.3***].

5.1.1.3 Cryptographic Symmetric Key Generation (PP_APP_V1.4:FCS_CKM.1/SK)

PP_APP_V1.4: FCS_CKM.1.1/SK

The application shall generate symmetric cryptographic keys using a Random Bit Generator as specified in FCS_RBG_EXT.1 and specified cryptographic key sizes [***256 bit***].

5.1.1.4 File Encryption Key (FEK) Generation (MOD_FE_V1.0:FCS_CKM_EXT.2)

MOD_FE_V1.0:FCS_CKM_EXT.2.1
The TSF shall [***generate FEK cryptographic keys [using a Random Bit Generator as specified in FCS_RBG_EXT.1 (from AppPP) and with entropy corresponding to the security strength of AES key sizes of [256 bit]]***]. (TD0455 applied)

MOD_FE_V1.0:FCS_CKM_EXT.2.2
The TSF shall use a unique FEK for each file (or set of files) using the mechanism on the client as specified in FCS_CKM_EXT.2.1.

5.1.1.5 Key Encrypting Key (KEK) Support (MOD_FE_V1.0:FCS_CKM_EXT.3)

MOD_FE_V1.0:FCS_CKM_EXT.3.1
The TSF shall [***generate KEK cryptographic keys [using a Random Bit Generator as specified in FCS_RBG_EXT.1 (from AppPP) and with entropy corresponding to the security strength of AES key sizes of [256 bit], derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6]***].

5.1.1.6 Cryptographic Key Destruction (MOD_FE_V1.0:FCS_CKM_EXT.4)

MOD_FE_V1.0:FCS_CKM_EXT.4.1
The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [
For volatile memory, the destruction shall be executed by a [destruction of reference to the key directly followed by a request for garbage collection]].

MOD_FE_V1.0:FCS_CKM_EXT.4.2
The TSF shall destroy all keys and key material when no longer needed.

5.1.1.7 Cryptographic Password/Passphrase Conditioning (MOD_FE_V1.0:FCS_CKM_EXT.6)

MOD_FE_V1.0:FCS_CKM_EXT.6.1
The TSF shall support a password/passphrase of up to [***[128]***] characters used to generate a password authorization factor.

MOD_FE_V1.0:FCS_CKM_EXT.6.2
The TSF shall allow passwords to be composed of any combination of upper case characters, lower case characters, numbers, and the following special characters: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')', and [***no other characters***].

MOD_FE_V1.0:FCS_CKM_EXT.6.3
The TSF shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC- [***SHA-512***] , with [***[4096] iterations***] , and output cryptographic key sizes [***256***] that meet the following: NIST SP 800-132.

MOD_FE_V1.0:FCS_CKM_EXT.6.4
The TSF shall not accept passwords less than [***a value settable by the administrator***] and greater than the maximum password length defined in FCS_CKM_EXT.6.1.

MOD_FE_V1.0:FCS_CKM_EXT.6.5
The TSF shall generate all salts using an RBG that meets FCS_RBG_EXT.1 (from AppPP) and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.6.3.

5.1.1.8 Cryptographic Operation - Encryption/Decryption (PP_APP_V1.4:FCS_COP.1/SKC)

PP_APP_V1.4:FCS_COP.1.1/SKC
The application shall perform encryption/decryption in accordance with a specified cryptographic algorithm [***AES-CBC (as defined in NIST SP 800-38A) mode***] and cryptographic key sizes [***256-bit***].

5.1.1.9 Cryptographic Operation - Hashing (PP_APP_V1.4:FCS_COP.1/Hash)

PP_APP_V1.4:FCS_COP.1.1/Hash

The application shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-256, SHA-384, SHA-512***] and message digest sizes [***256***, ***384, 512***] bits that meet the following: FIPS Pub 180-4.

5.1.1.10 Cryptographic Operation - Keyed-Hash Message Authentication - per TD0626 (PP_APP_V1.4:FCS_COP.1/KeyedHash)

PP_APP_V1.4:FCS_COP.1.1/KeyedHash

The application shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm [***SHA-512***] with key sizes [***256***] and message digest sizes [***512***] and [***no other size***] bits that meet the following: FIPS PUB 198-1 The Keyed-Hash Message Authentication Code and FIPS PUB 180-4 Secure Hash Standard.

5.1.1.11 Cryptographic operation (Key Wrapping) (MOD_FE_V1.0:FCS_COP.1(5))

MOD_FE_V1.0:FCS_COP.1.1(5)

The TSF shall [***implement functionality to perform Key Wrapping***] in accordance with a specified cryptographic algorithm AES in the following modes [***Key Wrap***] and cryptographic key sizes [***256 bits (AES)***] that meet the following: [***'NIST SP 800-38F'***] and no other standards.

5.1.1.12 Cryptographic operation (Key Transport) (MOD_FE_V1.0:FCS_COP.1(6))

MOD_FE_V1.0:FCS_COP.1(6).1

The TSF shall perform key transport in accordance with a specified cryptographic algorithm RSA in the following modes [***KTS-OAEP***] and cryptographic key sizes [***3072***] bits that meet the following: NIST SP 800-56B, Revision 1.

5.1.1.13 Initialization Vector Generation (MOD_FE_V1.0:FCS_IV_EXT.1)

MOD_FE_V1.0:FCS_IV_EXT.1.1

The TSF shall [***generate IVs with the following properties [CBC: IVs shall be non-repeating and unpredictable]***].

5.1.1.14 Key Chaining and Key Storage (MOD_FE_V1.0:FCS_KYC_EXT.1)

MOD_FE_V1.0:FCS_KYC_EXT.1.1

The TSF shall maintain a key chain of: [
KEKs originating from [one or more authorization factors(s)] to [the FEK(s)] using the following method(s):
[implementation of key wrapping as specified in FCS_COP.1(5),
implementation of key transport as specified in FCS_COP.1(6)]
while maintaining an effective strength of
[[256 bits] for symmetric keys,
[256 bits] for asymmetric keys]
commensurate with the strength of the FEK] and [***no supplemental key chains***].

5.1.1.15 Random Bit Generation Services (PP_APP_V1.4:FCS_RBG_EXT.1)

PP_APP_V1.4:FCS_RBG_EXT.1.1

The application shall [***invoke platform-provided DRBG functionality, implement DRBG functionality***] for its cryptographic operations.

5.1.1.16 Random Bit Generation from Application (PP_APP_V1.4:FCS_RBG_EXT.2)

PP_APP_V1.4:FCS_RBG_EXT.2.1
The application shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [***HASH_DRBG (any)***].

PP_APP_V1.4:FCS_RBG_EXT.2.2
The deterministic RBG shall be seeded by an entropy source that accumulates entropy from a platform-based DRBG and [***a software-based noise source***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

5.1.1.17 Storage of Credentials (PP_APP_V1.4:FCS_STO_EXT.1)

PP_APP_V1.4:FCS_STO_EXT.1.1
The application shall [***invoke the functionality provided by the platform to securely store [keys]***] to non-volatile memory.

5.1.1.18 Validation (MOD_FE_V1.0:FCS_VAL_EXT.1)

MOD_FE_V1.0:FCS_VAL_EXT.1.1
The TSF shall perform validation of the user by [***validating the [submask] using the following methods: [key wrap as specified in FCS_COP.1(5)]***].

MOD_FE_V1.0:FCS_VAL_EXT.1.2
The TSF shall require validation of the user prior to decrypting any FEK.

5.1.1.19 Validation Remediation (MOD_FE_V1.0:FCS_VAL_EXT.2)

MOD_FE_V1.0:FCS_VAL_EXT.2.1
The TSF shall [***institute a delay such that only [120 attempts] can be made within a 24 hour period***].

5.1.2 User data protection (FDP)

5.1.2.1 Encryption Of Sensitive Application Data (PP_APP_V1.4:FDP_DAR_EXT.1)

PP_APP_V1.4:FDP_DAR_EXT.1.1
The application shall [***implement functionality to encrypt sensitive data as defined in the PP-module for File Encryption)***] in non-volatile memory.

5.1.2.2 Access to Platform Resources (PP_APP_V1.4:FDP_DEC_EXT.1)

PP_APP_V1.4:FDP_DEC_EXT.1.1
The application shall restrict its access to [***network connectivity, USB, Bluetooth, [SD card]***].

PP_APP_V1.4:FDP_DEC_EXT.1.2
The application shall restrict its access to [***no sensitive information repositories***].

5.1.2.3 Network Communications (PP_APP_V1.4:FDP_NET_EXT.1)

PP_APP_V1.4:FDP_NET_EXT.1.1
The application shall restrict network communication to [***[application checking for updates]***].

5.1.2.4 Protection of Data in Power Managed States (MOD_FE_V1.0:FDP_PM_EXT.1)

MOD_FE_V1.0:FDP_PM_EXT.1.1
The TSF shall protect all data selected for encryption during the transition to the [***Data Locked***] state as per FDP_PRT_EXT.1.1.

MOD_FE_V1.0:FDP_PM_EXT.1.2

On the return to a powered-on state from the state(s) indicated in FDP_PM_EXT.1.1, the TSF shall authorize the user in the manner specified in FIA_AUT_EXT.1.1 once before any protected data are decrypted.

MOD_FE_V1.0:FDP_PM_EXT.1.3

The TSF shall destroy all key material and authentication factors stored in plaintext when transitioning to a protected state as defined by FDP_PM_EXT.1.1.

5.1.2.5 Protection of Selected User Data (MOD_FE_V1.0:FDP_PRT_EXT.1)

MOD_FE_V1.0:FDP_PRT_EXT.1.1

The TSF shall perform encryption and decryption of the user-selected file (or set of files) in accordance with FCS_COP.1(1) (from AppPP).

MOD_FE_V1.0:FDP_PRT_EXT.1.2

The TSF shall [***implement functionality***] to ensure that all sensitive data created by the TOE when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory when the data is no longer needed according to FCS_CKM_EXT.4.

5.1.2.6 Destruction of Plaintext Data (MOD_FE_V1.0:FDP_PRT_EXT.2)

MOD_FE_V1.0:FDP_PRT_EXT.2.1

The TSF shall [***implement functionality***] to ensure that all original plaintext data created when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory according to FCS_CKM_EXT.4 upon completion of the decryption/encryption operation.

5.1.3 Identification and authentication (FIA)

5.1.3.1 Subject Authorization (MOD_FE_V1.0:FIA_AUT_EXT.1)

MOD_FE_V1.0:FIA_AUT_EXT.1.1

The TSF shall [***provide user authorization***] based on [***a password authorization factor conditioned as defined in FCS_CKM_EXT.6***].

5.1.4 Security management (FMT)

5.1.4.1 Secure by Default Configuration (PP_APP_V1.4:FMT_CFG_EXT.1)

PP_APP_V1.4:FMT_CFG_EXT.1.1

The application shall provide only enough functionality to set new credentials when configured with default credentials or no credentials.

PP_APP_V1.4:FMT_CFG_EXT.1.2

The application shall be configured by default with file permissions which protect the application's binaries and data files from modification by normal unprivileged user.

5.1.4.2 Supported Configuration Mechanism (PP_APP_V1.4:FMT_MEC_EXT.1)

PP_APP_V1.4:FMT_MEC_EXT.1.1

The application shall [***invoke the mechanisms recommended by the platform vendor for storing and setting configuration options***]

5.1.4.3 Specification of Management Functions (PP_APP_V1.4:FMT_SMF.1)

PP_APP_V1.4:FMT_SMF.1.1

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

Change password/passphrase authentication credential
Lock/unlock the TOE management service,
Check for updates,
Configure encryption/decryption file status]].

5.1.4.4 Specification of File Encryption Management Functions (MOD_FE_V1.0:FMT_SMF.1(2))

MOD_FE_V1.0:FMT_SMF.1.1(2)
The TSF shall be capable of performing the following management functions: [***[***

change password/passphrase authentication credential]].

5.1.5 Privacy (FPR)

5.1.5.1 User Consent for Transmission of Personally Identifiable (PP_APP_V1.4:FPR_ANO_EXT.1)

PP_APP_V1.4:FPR_ANO_EXT.1.1
The application shall [***not transmit PII over a network***].

5.1.6 Protection of the TSF (FPT)

5.1.6.1 Anti-Exploitation Capabilities (PP_APP_V1.4:FPT_AEX_EXT.1)

PP_APP_V1.4:FPT_AEX_EXT.1.1
The application shall not request to map memory at an explicit address except for [***none***].
PP_APP_V1.4:FPT_AEX_EXT.1.2
The application shall [***not allocate any memory region with both write and execute permissions***].
PP_APP_V1.4:FPT_AEX_EXT.1.3
The application shall be compatible with security features provided by the platform vendor.
PP_APP_V1.4:FPT_AEX_EXT.1.4
The application shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.
PP_APP_V1.4:FPT_AEX_EXT.1.5
The application shall be built with stack-based buffer overflow protection enabled.

5.1.6.2 Use of Supported Services and APIs (PP_APP_V1.4:FPT_API_EXT.1)

PP_APP_V1.4:FPT_API_EXT.1.1
The application shall use only documented platform APIs.

5.1.6.3 Software Identification and Versions (PP_APP_V1.4:FPT_IDV_EXT.1)

PP_APP_V1.4:FPT_IDV_EXT.1.1
The application shall be versioned with [***[a unique release number]*****].**

5.1.6.4 Protection of Keys and Key Material (MOD_FE_V1.0:FPT_KYP_EXT.1)

MOD_FE_V1.0:FPT_KYP_EXT.1.1
The TSF shall [***store keys in non-volatile memory only when [wrapped, as specified in FCS_COP.1(5)]), the plaintext key is stored in the underlying platform's keystore as specified by FCS_STO_EXT.1.1 (from AppPP)***].

5.1.6.5 Use of Third Party Libraries (PP_APP_V1.4:FPT_LIB_EXT.1)

PP_APP_V1.4:FPT_LIB_EXT.1.1
The application shall be packaged with only [***WolfCrypt***].

5.1.6.6 Integrity for Installation and Update (PP_APP_V1.4:FPT_TUD_EXT.1)

PP_APP_V1.4:FPT_TUD_EXT.1.1
The application shall [***provide the ability***] to check for updates and patches to the application software.

PP_APP_V1.4:FPT_TUD_EXT.1.2
The application shall [***provide the ability***] to query the current version of the application software.

PP_APP_V1.4:FPT_TUD_EXT.1.3
The application shall not download, modify, replace or update its own binary code.

PP_APP_V1.4:FPT_TUD_EXT.1.4
Application updates shall be digitally signed such that the application platform can cryptographically verify them prior to installation.

PP_APP_V1.4:FPT_TUD_EXT.1.5
The application is distributed [***as an additional software package to the platform OS***].

5.1.6.7 Integrity for Installation and Update - per TD0664 (PP_APP_V1.4:FPT_TUD_EXT.2)

PP_APP_V1.4:FPT_TUD_EXT.2.1
The application shall be distributed using ***[the format of the platform-supported package manager***] (TD0628 applied).

PP_APP_V1.4:FPT_TUD_EXT.2.2
The application shall be packaged such that its removal results in the deletion of all traces of the application, with the exception of configuration settings, output files, and audit/log events.

PP_APP_V1.4:FPT_TUD_EXT.2.3
The application installation package shall be digitally signed such that its platform can cryptographically verify them prior to installation.

5.1.7 Trusted path/channels (FTP)

5.1.7.1 Protection of Data in Transit (PP_APP_V1.4:FTP_DIT_EXT.1)

PP_APP_V1.4:FTP_DIT_EXT.1.1
The application shall [***not transmit any [sensitive data]***] between itself and another trusted IT product.

5.2 TOE Security Assurance Requirements

The SARs for the TOE are the components as specified in Part 3 of the Common Criteria. Note that the SARs have effectively been refined with the assurance activities explicitly defined in association with both the SFRs and SARs.

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic Functional Specification
AGD: Guidance documents	AGD_OPE.1: Operational User Guidance
	AGD_PRE.1: Preparative Procedures
ALC: Life-cycle support	ALC_CMC.1: Labeling of the TOE
	ALC_CMS.1: TOE CM Coverage
	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent Testing - Conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability Survey

Table 2 Assurance Components

5.2.1 Development (ADV)

5.2.1.1 Basic Functional Specification (ADV_FSP.1)

ADV_FSP.1.1d
The developer shall provide a functional specification.

ADV_FSP.1.2d
The developer shall provide a tracing from the functional specification to the SFRs.

ADV_FSP.1.1c
The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.2c
The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.3c
The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.4c
The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ADV_FSP.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2e
The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational User Guidance (AGD_OPE.1)

AGD_OPE.1.1d
The developer shall provide operational user guidance.

AGD_OPE.1.1c
The operational user guidance shall describe, for each user role, the user accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c
The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3c
The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4c
The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5c
The operational user guidance shall identify all possible modes of operation of the TOE (including

operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AGD_OPE.1.6c

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

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

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

5.2.2.2 Preparative Procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE, including its preparative procedures.

AGD_PRE.1.1c

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

AGD_PRE.1.2c

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

AGD_PRE.1.1e

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

AGD_PRE.1.2e

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

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d

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

ALC_CMC.1.1c

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1e

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

5.2.3.2 TOE CM Coverage (ALC_CMS.1)

ALC_CMS.1.1d

The developer shall provide a configuration list for the TOE.

ALC_CMS.1.1c

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

ALC_CMS.1.2c

The configuration list shall uniquely identify the configuration items.

ALC_CMS.1.1e

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

5.2.3.3 Timely Security Updates (ALC_TSU_EXT.1)

ALC_TSU_EXT.1.1d
The developer shall provide a description in the TSS of how timely security updates are made to the TOE. Note: Application developers must support updates to their products for purposes of fixing security vulnerabilities.
ALC_TSU_EXT.1.2d
The developer shall provide a description in the TSS of how users are notified when updates change security properties or the configuration of the product.
ALC_TSU_EXT.1.1c
The description shall include the process for creating and deploying security updates for the TOE software.
ALC_TSU_EXT.1.2c
The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.
ALC_TSU_EXT.1.3c
The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.
Note: The reporting mechanism could include web sites, email addresses, as well as a means to protect the sensitive nature of the report (e.g., public keys that could be used to encrypt the details of a proof-of-concept exploit).
ALC_TSU_EXT.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Tests (ATE)

5.2.4.1 Independent Testing - Conformance (ATE_IND.1)

ATE_IND.1.1d
The developer shall provide the TOE for testing.
ATE_IND.1.1c
The TOE shall be suitable for testing.
ATE_IND.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.1.2e
The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.2.5 Vulnerability assessment (AVA)

5.2.5.1 Vulnerability Survey (AVA_VAN.1)

AVA_VAN.1.1d
The developer shall provide the TOE for testing.
AVA_VAN.1.1c
The TOE shall be suitable for testing.
AVA_VAN.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.1.2e
The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.
AVA_VAN.1.3e
The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to

determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6. TOE Summary Specification

This chapter describes the security functions:

- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Privacy
- Protection of the TSF
- Trusted path/channels

6.1 Cryptographic support

The TOE operates on evaluated or Android 11 devices, and uses the Cyber Reliant Mobile Data Defender for Android SDK with the WolfCrypt module version 4.7 for encryption services and random number generation. The TOE utilizes cryptographic functions provided by the TOE platform.

The following table denotes the CAVP certificates for the WolfCrypt module version 4.7 used by the TOE for cryptographic operations.

Functions	Standards	CAVP Certificates
Encryption/Decryption		
AES-CBC (256 bits)	FIPS PUB 197, NIST SP 800-38E	A3240
Cryptographic Hashing		
SHA-256 SHA-384 SHA-512	FIPS Pub 180-4	A3240
Keyed-Hash Message Authentication		
HMAC-SHA-512	FIPS Pub 198-1, FIPS Pub 180-4	A3240
Key Wrapping		
AES Key Wrap (256-bit)	NIST SP 800-38F	A3240
Key Transport		
RSA KTS-OAEP	NIST SP 800-56B, Revision 1	Vendor-Affirmed
Random Bit Generation		
AES-HASH-DRBG (256 bits)	NIST SP 800-90A	A3240

Table 6-1 CAVP Certificates

The Cryptographic support function satisfies the following security functional requirements:

- **PP_APP_V1.4:FCS_CKM.1/AK/PP_APP_V1.4:FCS_CKM_EXT.1/ MOD_FE_V1.0:FCS_CKM_EXT.2**

  The TOE generates keys using both the CRC Encryption and Shredding engine cryptographic module and the platform's API (KeyGenerator) as described here. The CRC Encryption and Shredding Engine Cryptographic module API uses an SP 800-90A HASH- DRBG and the platform uses an SP 800-90A AES-256 CTR DRBG. Both DRBGs are seeded with sufficient entropy from the platform itself.

  The platform DRBG is used to generate 256-bit AES FEKEKs. The file encryption key encryption key (FEKEK) is generated using the KeyGenerator API. The FEKEK is stored in the Management Service's WolfCrypt keystore. The FEKEK is wrapped twice, once using RSA and one more time using AES. The AES key used to wrap the FEKEK is derived from the DaR password using PBKDF2. Both the Management Service and Application's 2048-bit RSA keys used to wrap the FEKEK are generated using the KeyGenerator API. If the DaR password provided by the user is not correct, then the Management Service's WolfCrypt keystore will not properly load, preventing the Management Service from accessing its keystore. Furthermore, an incorrect DaR password results in the incorrect derivation of the PBKDF2 derived AES key, therefore the FEKEK will not be unwrapped properly. The CRC Encryption and Shredding Engine Cryptographic module DRBG is used to generate the FEKs. The FEK is wrapped by the FEKEK before being stored

- **PP_APP_V1.4:FCS_CKM.1/SK/MOD_FE_V1.0:FCS_CKM_EXT.2**

The TOE generates file encryption keys using the CRC Encryption and Shredding engine cryptographic module which implements an SP 800-90A HASH DRBG. The DRBG is seeded with sufficient entropy to generate keys with 256 bits of security strength by using seeding material collected by the evaluated platform. The FEKs are generated every time a new file is going to be encrypted. The TOE associates a FEK with an individual file that is being encrypted by storing the wrapped FEK in the same hidden directory as the encrypted file. The TOE automatically generates a FEK (without user action) whenever the application attempts to encrypt a new file

- **MOD_FE_V1.0:FCS_CKM_EXT.3**

The platform DRBG is used to generate 256-bit AES FEKEKs. The file encryption key encryption key (FEKEK) is generated using the KeyGenerator API. The FEKEK is stored in the Management Service's WolfCrypt keystore. The FEKEK is wrapped twice, once using RSA and one more time using AES. The AES key used to wrap the FEKEK is derived from the DaR password using PBKDF2. Both the Management Service and Application's RSA keys used to wrap the FEKEK are generated using the KeyGenerator API. If the DaR password provided by the user is not correct, then the Management Service's keystore will not properly load, preventing the Management Service from accessing its keystore. Furthermore, an incorrect DaR password results in the incorrect derivation of the PBKDF2 derived AES key, therefore the FEKEK will not be unwrapped properly. The CRC Encryption and Shredding Engine Cryptographic module DRBG is used to generate the FEKs and requires no user action other than file access. The FEK is wrapped by the FEKEK before being stored.

- **MOD_FE_V1.0:FCS_CKM_EXT.4**

The TOE relies on the platform and CRC Encryption and Shredding Engine Cryptographic module for destroying keys. The platform utilizes Java Garbage Collection in order to clear memory. The TOE releases all references to objects (e.g. keys) when they are no longer needed, and the Java Garbage Collection clears out the memory that is no longer in use. The CRC Encryption and Shredding Engine Cryptographic module also utilizes platform functions to destroy FEKs. The CRC Encryption and Shredding Engine Cryptographic module destroys API overwrites the reference to the key directly and initiates a request for garbage collection for the destruction of non-persistent CSPs from volatile memory.

Key Name	Cleartext Storage Location	Destruction	Entity responsible for Destruction	When it is destroyed
FEK	RAM	Native memory destruction	CRC Encryption and Shredding Engine Cryptographic module	After file encryption/decryption
Management Service unwrapped FEKEK	RAM	Java Garbage Collection	Platform key destruction API	After wrapping the FEKEK with the Application's RSA key
Application unwrapped FEKEK	RAM	Java Garbage Collection	CRC Encryption and Shredding Engine Cryptographic module	After wrapping/unwrapping the FEK
PBKDF2 derived key	RAM	Java Garbage Collection	Platform key destruction API	After AES unwrapping double wrapped FEKEK
Management Service's private RSA key	TrustZone	TrustZone	TrustZone	After RSA unwrapping of double-wrapped FEKEK
Application's private RSA key	TrustZone	TrustZone	TrustZone	After RSA unwrapping of single-wrapped FEKEK

Table 6-2 Key Destruction

- **MOD_FE_V1.0:FCS_CKM_EXT.6**

The TOE allows the use of DaR passwords that support all special characters mentioned in FCS_CKM_EXT.6. The TOE encodes the DaR password using UTF-8 before the DaR password is passed into the evaluated Android platform's cryptographic APIs to perform Password-Based key derivation (SP 800-132 PBKDF2) using HMAC-SHA-512 pseudo-random function. Note that this DaR password does not derive the FEK. The password input into the PBKDF2 function results in a HMAC-SHA-512 output that is 512 bits long. The TOE truncates this output to a length of 256 bits without any further manipulation and uses that value as a derived key to perform a secondary AES wrap on the FEKEK in conjunction with an asymmetric key wrap using an RSA key pair stored in the Android keystore. The TOE enforces a minimum password length of 6 characters and can support a maximum password of 128 characters.

- **PP_APP_V1.4:FCS_COP.1/SKC**

The TOE invokes the WolfSSL Module to perform AES-256-CBC encryption. The TOE invokes the CRC Encryption and Shredding Engine Cryptographic module's AES implementation to perform AES 256-CBC encryption when encrypting files using the 256-bit FEK.

- **PP_APP_V1.4:FCS_COP.1/HASH**

The TOE performs SHA hashing using its WolfSSL Module. The TOE supports hash sizes of SHA-256, SHA-384, SHA-512 and message digest sizes 256, 384, and 512 bits. The TOE uses its SHA_256 function to support the HASH-DRBG. The TOE uses its WolfSSL Module SHA-384 to hash the DaR password with a salt generated by the platform's DRBG. The TOE uses the 512 hash function to support the keyed hash function.

- **PP_APP_V1.4:FCS_COP.1/KeyedHash**:

The TOE uses the platform HMAC-SHA-512 to hash a PBKDF2 derived key with a second salt to produce a key used to further unwrap the FEKEK using AES key wrap, after the FEKEK has been unwrapped using RSA-OAEP.

- **MOD_FE_V1.0:FCS_COP.1(5)/MOD_FE_V1.0:FCS_COP.1(6)**

The TOE uses its WolfSSL Module 256 bit AES key wrapping to wrap the FEK with the FEKEK. The TOE performs AES key wrapping in accordance with SP 800-38F and RSA key wrapping in accordance with SP 800-56B. The TOE's [AGD] contains the full list of APIs used by the TOE. The TOE leverages platform APIs to perform RSA-OAEP key transport using 3072-bit RSA/ECB/NoPadding, SHA-512 for hashing and mask generation, and the platform's invoked DRBG source though Java's SecureRandom API's.

- **MOD_FE_V1.0:FCS_IV_EXT.1**

The TOE calls the CRC Encryption and Shredding Engine Cryptographic module's API to access that module's Hash_DRBG to generate IV values. By using a Hash_DRBG to generate its IVs, the TOE guarantees those IVs to be non-repeating and unpredictable. The TOE's [AGD] contains the full list of APIs used by the TOE. The TOE only uses IVs when performing AES-256 encryption/decryption of user data using the FEK (IVs are not used as part of any key wrap/unwrap process).

- **MOD_FE_V1.0:FCS_KYC_EXT.1**

The DaR password is used in two places. First, it is hashed using SHA-384 with a salt (generated using the platform's DRBG) and used to load the Management Service's keystore. The DaR password is then used as input to PBKDF2 with 4096 iterations and HMAC-SHA-512 PRF along with a second salt value (also generated using the platform's DRBG).

The platform retrieves the double-wrapped FEKEK from the Management Service's keystore. The double wrapped FEKEK is AES unwrapped using the 256-bit PBKDF2 derived key, resulting in an RSA wrapped FEKEK.

The RSA wrapped FEKEK is unwrapped using the Management Service's RSA keypair, resulting in a cleartext FEKEK that is re-encrypted using the Application's RSA public key. The Management Service passes this single wrapped FEKEK to the application. The Management Service obtains the Application's RSA public key as well as the Application's certificate fingerprint when the Application registers itself to the Management Service (performed once). The Application's RSA public key is kept within Android's SharedPreferences under MODE_PRIVATE.

The Management Service uses the Application's certificate fingerprint to read/write to the Application's keystore

- **PP_APP_V1.4:FCS_RBG_EXT.1**

TOE uses the wolfCrypt Module for random number generation, which utilizes an SP 800-90A HASH DRBG. This DRBG is used to generate the File Encryption Keys (FEKs). The TOE generates the FEKEK using the platform's AES-256 AES CTR DRBG

- **PP_APP_V1.4:FCS_RBG_EXT.2**

The TOE uses the wolfCrypt Module to generate random data. This Module always creates a global HASH_DRBG upon load, seeds the global DRBG with 384-bits taken from /dev/urandom, uses that global DRBG elsewhere within the module when it requires random data, and reseeds the global DRBG (again drawing from /dev/urandom) after generating 1,000 blocks of random output.

When the TOE must generate an AES-256 bit FEK, the wolfCrypt Module creates a separate HASH_DRBG context that it uses solely for generation of keys. The module will seed this AES-256 DRBG context using 384-bits of data drawn from the global DRBG.

- **PP_APP_V1.4:FCS_STO_EXT.1**

All claimed keystores live in Android's protected directory /data/misc/keystore, which is also on the platform's flash storage. The TOE uses the evaluated Android platform's Android keystore to store all RSA keypairs.

Upon registration, each Application passes its public key to the Management Service, which is stored in Android's SharedPreferences under MODE_PRIVATE. The TOE uses the Application's public key to wrap the FEKEK for unwrapping and use by the application.

- The TOE uses the Management Service's keypair to keywrap the FEKEK, which is again wrapped by the PBKDF-derived key to form the double wrapped FEKEK for storage.
- The wrapped FEK and IV are stored in the same hidden directory as the encrypted file, which is all stored on flash

- **MOD_FE_V1.0:FCS_VAL_EXT.1**

See MOD_FE_V1.0:FCS_KYC_EXT.1 where Validation is described in detail.

- **MOD_FE_V1.0:FCS_VAL_EXT.2**

The TOE only allows for a single password to be used for authentication. After 5 attempts at authenticating with the wrong password, the TOE will lock out the service for 3600 seconds (60 minutes) before allowing the user to attempt a password authentication again. This effectively sets a limit at a total of 120 attempts per 24 hour period.

6.2 User data protection

The TOE protects user data by providing encryption of user selected data. The TOE uses 256-bit AES keys to encrypt the files stored to flash. These keys are derived from DaR passwords through a key derivation and key wrapping process. If an application does not enter the right DaR password, the file encryption key will never be derived correctly, thus preventing the application from decrypting its files. The AES keys used as a File Encryption Key (FEK) are generated using the wolfCrypt Module. The random data collected to seed the CRC Encryption and Shredding Engine Cryptographic module DRBG comes from the evaluated device's /dev/urandom

The User data protection function satisfies the following security functional requirements:

- **PP_APP_V1.4:FDP_DAR_EXT.1**

The TOE offers two groups of API: one set to manipulate files and one set to manipulate SQLite databases. While the API groups provide different abstractions for the read and write operations, they both are ultimately simply reading and writing a single file. The TOE implements functionality to encrypt data and store it securely on the evaluated platform. The TOE uses the CRC Encryption and Shredding engine cryptographic module to generate AES-256 bit keys for file encryption (a.k.a., FEK).

- **PP_APP_V1.4:FDP_DEC_EXT.1**

The TOE only uses the READ_EXTERNAL_STORAGE to write to an external SD Card. The TOE does not access any of the listed sensitive information repositories. The TOE communicates over networks only for the purpose of checking for TOE updates and does not transmit PII data over a network. According to the TOE's [AGD], Cyber Reliant Mobile Data Defender for Android SDK can read/write data to any directory based on the permissions of the integrated application. These directories can reside either in the internal Android SD Card or in expandable memory.

The TOE requires permissions in order to perform the following:

- Modify or delete the contents of USB storage
- Read the contents of USB storage
- Full network access in order to check for product updates
- Reorder running apps
- Retrieve running apps
- Run at startup
- Prevent Phone from Sleeping

- **PP_APP_V1.4:FDP_NET_EXT.1**

The TOE communicates over networks only for the purpose of checking for TOE updates and does not transmit PII data over a network.

- **MOD_FE_V1.0:FDP_PM_EXT.1**

The TOE has 3 states: Power-off, Data Locked, and Data Unlocked. In the Power-off state, the device is off and TOE is not available.

The TOE spends most of its time between the Data Lock and Data Unlock states. It is not possible to move directly from Power-off to Data Unlock; the TOE always starts in the Data Lock state on startup.

During the Power-off state, all keys that may be in use are removed from volatile storage automatically, regardless of whether this state was by user shutdown/restart or by power failure.

On startup, the device places the TOE into the Data Lock state. From the Data Lock state, the user must authenticate successfully to the CRC Encryption and Shredding Engine Cryptographic module to transition to the Data Unlock state. In the Data Unlock phase, the passphrase is hashed using SHA-384 with a salt (generated using the platform's DRBG) and used to load the Management Service's keystore. The DaR password is then used as input to PBKDF2 with 4096 iterations and HMAC-SHA-512 PRF along with a second salt value (also generated using the platform's DRBG).

The "Data Locked" state is when the screen is locked. In this state the TOE deletes the 256-bit HASH KEK and encrypted 256-bit Master KEK from volatile memory, leaving only the encrypted files. A screen can be screen locked, password locked, or data locked after a period of inactivity. The TOE requires the user to re-authenticate (i.e., provide the DaR password) to decrypt files following a mobile device entering the "Data Locked" state.

- **MOD_FE_V1.0:FDP_PRT_EXT.1\MOD_FE_V1.0:FDP_PRT_EXT.2**

The TOE encrypts each individual file separately. The encrypted files are stored in a hidden directory at the same directory level as the original files. Before encryption, the TOE splits a file into chunks to more easily access specific parts within an encrypted file. Using the CRC Encryption and Shredding Engine Cryptographic Module's API, each chunk is then encrypted separately. The original file is replaced with a directory structure of different files after chunking and encryption. The directory structure of the encrypted file includes a metadata file that describes the chunking structure, a hidden folder for every chunk that includes a header file, and the encrypted file chunks split into encrypted pieces. The TOE implements functionality to ensure that sensitive data is destroyed in volatile and non-volatile memory upon completion of either a decrypt or encrypt operation of the sensitive files. The Cyber Reliant Mobile Data Defender for Android SDK returns the plaintext data returned by the API to the application for processing. The Cyber Reliant Mobile Data Defender for Android SDK clears all internal buffers of the data. The CRC Encryption and Shredding Engine Cryptographic Module also has routines to destroy keys stored inside. The TOE does not create any temporary resources.

Each chunk is decrypted separately (using the CRC Encryption and Shredding Engine Cryptographic module's decryption API). This allows much faster access to read/write encrypted data to the encrypted file (e.g. random access files). For example, if data is added to the middle of an encrypted file that has not been chunked, the entire file needs to be decrypted, and the data needs to be inserted before the file is re-encrypted. In the same scenario, if the encrypted file was chunked, then the first half of the chunks can be skipped before reaching the point at which the data needs to be encrypted and inserted. The Cyber Reliant Mobile Data Defender for Android SDK chunks the data set on 10MB boundaries, so if only a subset of the data is needed, the system will only encrypt the 10MB chunk that contains the desired data set. If the data in the file wishes to be changed, then the whole file must be re-encrypted. Decrypted pieces are retained for caching purposes for up to 30 seconds, before they are purged and the memory is wiped. Each file has a unique FEK and IV, which is used to encrypt/decrypt each chunk. The wrapped FEK and IV are stored in a hidden directory that resides in the same directory as the encrypted file.

The TOE programmatically destroys keys in volatile memory per Table 6-2 Key Destruction (such as keys stored in RAM and used by the TOE) by calling Android's Arrays.fill method in order to zero out the key array.

The CRC Encryption and Shredding Engine Cryptographic Module also has its own zeroization. The module_destroy API zeroes the FEK and FEKEK from volatile memory per Table 6-2 Key Destruction.

6.3 Identification and authentication

The TOE maintains identification and authentication by using DaR passwords. In order for an application to unlock its files, the application must provide the correct DaR password. The DaR password is used to derive the necessary keys in order to obtain the file encryption key, which is used to decrypt the files.

The Identification and authentication function satisfies the following security functional requirements:

- **MOD_FE_V1.0:FIA_AUT_EXT.1**

The TOE provides a DaR password based authorization factor in order to authenticate to the Cyber Reliant Mobile Data Defender for Android SDK service

6.4 Security management

The TOE does not allow invocation of its services until a configuration file has been created. The configuration options are stored in the evaluated or Android 11 operating system's defined private area on flash memory. The TOE allows users to change DaR passwords as part of its security management.

The Security management function satisfies the following security functional requirements:

- **PP_APP_V1.4:FMT_CFG_EXT.1**

The TOE has a default password but provides only enough functionality to set new credentials when first installed. The encryption services are greyed out until configuration file is created.

- **PP_APP_V1.4:FMT_MEC_EXT.1**

The TOE's evaluated Android platform automatically uses /data/data/package/shared_prefs/ to store configuration options and settings. The evaluated TOE is distributed with an embedded configuration which includes a list of apps to encypt, shredding configurations, IP address of the TCM (N/A), password requirements and settings, and authentication lockout settings, however these are not configurable post-installation. The Time-outs, Lock-outs, and SALTs for PBKDFv2 are stored in private files in the file directory.

- **PP_APP_V1.4:FMT_SMF.1/ MOD_FE_V1.0:FMT_SMF.1(2)**

The TOE provides the ability to change DaR passwords/passphrases, lock/unlock the TOE management service, check for updates, and configure encryption/decryption file status. Upon a password/passphrase change, the TOE will load the double-wrapped FEKEK, rederive the old PBKDF-derived key to singularly unwrap the double-wrapped FEKEK, and then use the new password to rewrap this and store this into the embedded WolfCrypt keystore. During this process, the FEKEK is never unwrapped with the TOE management service's RSA keypair, meaning the cleartext FEKEK and any resulting FEK plaintext values are never exposed.

6.5 Privacy

The Privacy function satisfies the following security functional requirements:

- **PP_APP_V1.4:FPR_ANO_EXT.1**

The TOE does not transmit any PII over a network. The only use of a network is to determine the currently available product version for the purpose of detecting available updates.

6.6 Protection of the TSF

The TOE is physically protected by the boundary of the evaluated device. The TOE is executed on an evaluated Android 11 OS. The TOE utilizes the evaluated platform APIs only. Android's application management requires application updates to be signed with an Android key, thus allowing the secure updates of its applications. The Android OS Linux kernel is capable of ASLR (address space layout randomization), ensuring that no application uses the same address layout on two different devices. Keys are also stored in memory, which can be wiped by rebooting the device.

The Protection of the TSF function satisfies the following security functional requirements:

- **PP_APP_V1.4:FPT_AEX_EXT.1**

The application is compiled with "-v -DBUILD_JNI -DANDROID -DCyber -O2 -fstack-protector-all -fexceptions" in order to enable ASLR and stack-based buffer overflow protection. The Linux kernel of the TOE's Android OS also 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.

- **PP_APP_V1.4:FPT_API_EXT.1**

The TOE uses only platform provided APIs and does not import any third-party APIs. The TOE's [AGD, section 5.5] contains the full list of APIs used by the TOE.

- **PP_APP_V1.4:FPT_IDV_EXT.1**

The TOE labels its software using a unique release number that can be queried by users.

- **MOD_FE_V1.0:FPT_KYP_EXT.1**

The TOE stores plaintext keys in non-volatile memory by relying on the Android platform keystores. The TOE uses the Management Service's WolfCrypt keystore to store the double wrapped FEKEK. The double-wrapped FEKEK is unwrapped using AES key wrap and the PBKDF derived value from the user's password. The TOE uses the Android keystore to store all RSA public/private keys, which are used to unwrap the single-wrapped FEKEK. In order to pass the plaintext FEKEK between the Management service and the registered application, the plaintext FEKEK is wrapped using an RSA public/private keypair stored in the application's platform keystore. The TOE wraps the file encryption keys (FEK, used for encrypting files) with another AES key, and then stores the wrapped FEK on the evaluated device's flash memory. The TOE utilizes the evaluated Android platform's AES key wrap and RSA OAEP key wrap functions to protect keys.

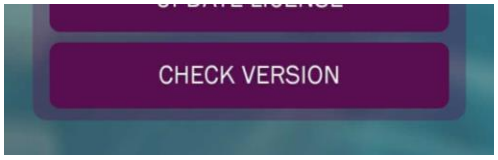
- **PP_APP_V1.4:FPT_LIB_EXT.1**

The TOE only uses the third-party WolfCrypt v4.3 library that is compiled by the vendor and is CAVP-certified for its cryptographic use as a part of this evaluation.

- **PP_APP_V1.4:FPT_TUD_EXT.1/ PP_APP_V1.4:FPT_TUD_EXT.2/ALC_TSU_EXT.1**

The TOE's software is distributed as APK files that are digitally signed by Cyber Reliant. Each update is accompanied by documentation outlining changes to the overall service, as well as compatible versions of the Cyber Reliant API.

Checking for updates can be done in the app by selecting the following button in the Management Service:



A popup will appear indicating whether an update is necessary and instructions on how to retrieve it. Updates are distributed through email from the vendor which contains a download link to the software patch hosted by CRC's ShareFile portal.

If a security vulnerability was found by a user, then the user must report it to Cyber Reliant via email at support@cyberreliant.com. In the case that the vulnerability impacts the Cyber Reliant Mobile Data Defender for Android SDK: Cyber Reliant will deliver updated API Code, as well as additional developer documentation outlining any potential changes in the implementation, within a maximum of 30 days from time of vulnerability disclosure. In order to deliver the final resolution to the end-user, the partner developer or customer will need to implement the updated code into their target applications. The time for final delivery will be dependent on their ability to update the end-user application, and to distribute to users via Mobile Device Management Service, application store, or other delivery mechanism.

In case the vulnerability directly relates to the Management Service APK: Cyber Reliant shall deliver, via email or other agreed upon method, an updated application with security vulnerabilities addressed. The delivered software shall be accompanied by documentation outlining changes to the overall service, as well as compatible versions of the API. Once delivered to the customer or partner, the application can be delivered to end-users via Internal MDM instances, Internal 'App Stores' or other agreed upon methodologies.

6.7 Trusted path/channels

The TOE does not transmit sensitive data to any other products.

The Trusted path/channels function satisfies the following security functional requirements:

- **PP_APP_V1.4:FTP_DIT_EXT.1**

The TOE does not transmit any sensitive data between itself and other products.