KeyW Protect for Samsung, Version 1.2.1.0 (PP_APP_V1.3/MOD_FE_V1.0) Security Target

Version 0.4
June 4, 2020

Prepared for:

KeyW Corporation

7740 Milestone Parkway
Suite 500
Hanover, MD 21076

www.keywcorp.com

Prepared By:

www.gossamersec.com

List of Tables

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is KeyW Protect for Samsung provided by KeyW Corporation. The TOE is being evaluated as a file 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 – KeyW Protect for Samsung, Version 1.2.1.0 (PP_APP_V1.3/MOD_FE_V1.0) Security Target

ST Version – Version 0.4

ST Date – June 4, 2020

1.2 TOE Reference

TOE Identification – KeyW Protect for Samsung, Version 1.2.1.0

TOE Developer – KeyW Corporation

Evaluation Sponsor – United States Special Operations Command (USSOCOM)

1.3 TOE Overview

The Target of Evaluation (TOE) is KeyW Protect for Samsung.

The KeyW Protect for Samsung TOE is also known as KEYWprotect. The TOE provides an AES-based Data at Rest (DAR) encryption model that is used to encrypt the Android Enterprise workspace data when the workspace is unlocked, and enables the protection of workspace data when the workspace is locked and when the Samsung mobile device is powered off. The TOE is an application on the Samsung mobile device. The TOE was evaluated on the following Samsung mobile devices.

Device Name	Model Number	Processor	Operating System
Samsung Galaxy S10e	SM-G970	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy S10e	SM-G970	Samsung Exynos 9820	Android 9.0

In addition to the evaluated devices, the following Samsung mobile devices are claimed as equivalent.

Device Name	Model Number	Processor	Operating System
Samsung Galaxy S10	SM-G973	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy S10	SM-G973	Samsung Exynos 9820	Android 9.0
Samsung Galaxy S10+	SM-G975	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy S10+	SM-G975	Samsung Exynos 9820	Android 9.0
Samsung Galaxy S10 5G	SM-G977	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy S10 5G	SM-G977	Samsung Exynos 9820	Android 9.0
Samsung Galaxy Note10	SM-N970	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy Note10 5G	SM-N971	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy Note10+	SM-N975	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy Note10+ 5G	SM-N976	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy A90 5G	SM-A908	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy Fold	SM-F900	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Galaxy Fold 5G	SM-F907	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Tab S6 (Wi-Fi)	SM-T860	Qualcomm Snapdragon 855 (SDM855)	Android 9.0
Samsung Tab S6 (LTE)	SM-T865	Qualcomm Snapdragon 855 (SDM855)	Android 9.0

The TOE utilizes Samsung Knox DualDAR Application Programming Interface (API) to enable DAR by relaying all file operations within the Android Enterprise workspace to itself for encryption/decryption. When the Android Enterprise workspace is unlocked, all workspace data created by other applications is automatically encrypted by the TOE and stored in the workspace file system via the Samsung Knox DualDAR API. Therefore, no clear text is ever written to the Android Enterprise workspace file system and workspace data may only be decrypted as needed after a user presents valid authentication factors.

1.4 TOE Description

The KeyW Protect for Samsung application (i.e., the TOE) is a file encryption tool that runs on a Samsung mobile device with Android OS 9.0, Knox 3.3 and DualDAR 1.0. The Samsung Knox DualDAR API is used to relay all file operations within the Android Enterprise workspace to the TOE, which encrypts or decrypts the file contents automatically. The TOE is a headless Android OS Suite B Data-At-Rest (DAR) encryption application on the Samsung mobile device, which is designed to auto-run and register at mobile device startup for protection of files that reside within the Android Enterprise workspace. Therefore, the Android operating system will ensure that the mobile device features are available only when the TOE is running.

The TOE utilizes the Android OS 9.0 for storage of keys while the TOE includes the *Suite B Cryptographic Algorithms* library, which implements its own random bit generation, AES encryption/decryption, AES key wrapping, keyed-hashing functions, password-based key derivation, key pair generation, key establishment and cryptographic hashing, which have been certified through CAVP.

1.4.1 TOE Architecture

The TOE is an application that is installed as a required headless application when the Samsung mobile device is activated for Android Enterprise for use. Being a required application, the Android OS 9.0 platform with Knox 3.3 and DualDAR 1.0 ensures that the TOE is running prior to presenting the home screen to the user, and prevents all actions, which could uninstall the application. The TOE provides its own cryptographic functionality via the existing *Suite B Cryptographic Module*, which has been FIPS-validated through CAVP and CMVP certifications (details of the relevant CAVP certificates are provided later in this document).

When the Android Enterprise workspace is unlocked, all workspace data created by other applications is automatically encrypted by the TOE and stored in the workspace file system via the Samsung Knox DualDAR API. Therefore, no clear text version of the file is ever created on the workspace file system.

The TOE generates and stores the following keys and Critical Security Parameters (CSPs):

- Generates and stores a static 521-bit ECC Client keypair encrypted by a non-exportable 256-bit Key Encryption Key (KEK) in the private shared preferences file using AES/CBC-256 with no padding and a 128-bit Initialization Vector (IV).
    - Stores the 128-bit IV in the private shared preferences file.
    - Stores the compressed 521-bit ECC Client Public Key in the Secrets metadata contained in volatile memory.
- Creates a private Native State file to store the Native State File Descriptor.
- Stores a Prepare Flag in the Native State contained in volatile memory.
- Generates and stores an ephemeral 521-bit ECC Native keypair in the Native State contained in volatile memory.
    - Stores the compressed ephemeral 521-bit ECC Native Public Key in the Version metadata contained in volatile memory.
    - Evaluates the Prepare Flag to determine whether to zeroize the ephemeral 521-bit ECC Native keypair upon data lock.
- Computes a Shared Secret (SS) using ECC CDH Primitive from the static 521-bit ECC Client keypair and ephemeral 521-bit ECC Native keypair.
- Derives a 256-bit Secret KEK from the Shared Secret (SS) using SHA.
- Generates and stores the following keys and CSPs when initializing the user's workspace without a password:
    - Generates and stores a 256-bit Temporary HASH KEK encrypted by a non-exportable 256-bit KEK in the private shared preferences file using AES/CBC-256 with no padding and a 128-bit IV.
        - Stores the 128-bit IV in the private shared preferences file.
        - Stores the 256-bit Temporary HASH KEK encrypted by the Secret KEK in the Secrets metadata contained in volatile memory.
        - Stores the 256-bit Temporary HASH KEK in the Native State contained in volatile memory.
    - Generates and stores a 256-bit Master KEK encrypted by the 256-bit Temporary HASH KEK in the private shared preferences file, Secrets metadata contained in volatile memory, and Native State contained in volatile memory using AES Key Wrap.
- Generates and stores the following keys and CSPs when setting the user's workspace password:
    - Generates and stores a 256-bit SALT encrypted by a non-exportable 256-bit KEK in the private shared preferences file using AES/CBC-256 with no padding and a 128-bit IV.

        - Stores the 128-bit IV in the private shared preferences file.
    - Derives a 256-bit HASH KEK from the user's workspace password and the 256-bit SALT using PBKDF.
        - Stores the 256-bit HASH KEK encrypted by the Secret KEK in the Secrets metadata contained in volatile memory.
        - Stores the 256-bit HASH KEK in the Native State contained in volatile memory.
    - Stores the 256-bit Master KEK encrypted by the 256-bit HASH KEK in the private shared preferences file, Secrets metadata contained in volatile memory, and the Native State contained in volatile memory using AES Key Wrap.
- Generates and stores the following keys and CSPs when resetting the user's workspace password with a reset token:
    - Generates and stores a 256-bit Reset Token SALT encrypted by a non-exportable 256-bit KEK in the private shared preferences file using AES/CBC-256 with no padding and a 128-bit IV.
        - Stores the 128-bit IV in the private shared preferences file.
    - Derives a 256-bit Reset Token HASH KEK from the Reset Token Password and the 256-bit Reset Token SALT using PBKDF.
    - Stores the 256-bit Master KEK encrypted by the 256-bit Reset Token HASH KEK in the private shared preferences file using AES Key Wrap.
- Generates and stores a File Encryption Key (FEK) encrypted by the 256-bit Master KEK in the associated metadata of each encrypted file using AES Key Wrap.

1.4.1.1 Physical Boundaries

The physical boundary of the TOE is the physical perimeter of the evaluated device (Samsung mobile device with Android OS 9.0, Knox 3.3 and DualDAR 1.0) on which the TOE resides.

The UEM console is used to install and update the TOE.

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by the TOE.

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

1.4.1.2.1 Cryptographic support

The TOE operates on a Samsung mobile device and uses features provided by the platform for key storage. The TOE includes the *Suite B Cryptographic Algorithms* library, which implements its own algorithms for random bit generation, AES encryption/decryption, AES key wrapping, keyed-hashing functions, password-based Key Derivation, key pair generation, key establishment and cryptographic hashing.

1.4.1.2.2 User data protection

The TOE protects user data by providing an integrated file encryption capability that automatically encrypts new files and decrypts files upon user demand. The TOE utilizes 256-bit AES encryption for confidentiality.

1.4.1.2.3 Identification and authentication

The TOE authenticates a user by requiring a password before any file data decryption operation is initiated. Without the correct password, the user is unable to decrypt the keys necessary to obtain clear text data from the Android Enterprise workspace file system.

1.4.1.2.4 Security management

The TOE does not allow encryption/decryption operations while in the locked state until the user authenticates to the device upon first use of the TOE. The TOE allows the following user management capabilities:

- Change workspace password.
- Reset workspace password using a reset token from the Unified Endpoint Management (UEM) console. Samsung Knox DualDAR by default does not disable reset passwords thereby enabling key recovery. To disable all key recovery mechanisms simply do not set a password using a token, which will prevent a password reset from the IT admin.
- Configure password/passphrase complexity settings including the minimum and maximum lengths.
- Perform a cryptographic erase of the data.
- Configure the corrective behavior (wipe/disable workspace) and number of failed validation attempts required to trigger corrective behavior.

1.4.1.2.5 Protection of the TSF

The TOE relies on the physical boundary of the evaluated platform as well as the Android 9.0 operating system for the protection of the TOE's application components.

Updates to the TOE are handled via the UEM console.

1.4.1.2.6 Trusted path/channels

The TOE does not transmit any data between itself and another network entity. All of the data managed by the TOE resides on the evaluated platform (Samsung mobile device with Android OS 9.0, Knox 3.3, and DualDAR 1.0).

1.4.2 TOE Documentation

KeyW offers the following documentation to users for the installation and operation of their product. The following list of documents was examined as part of the evaluation.

[Guide] Android OS Suite B Data At Rest v1.2.1.0 – User Guide, Document Version 1.3, 05/19/2020.

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 Conformant
- PP-Configuration for Application Software and File Encryption, Version 1.0, 25 July 2019 (CFG_APP-FE_V1.0)
  - The PP-Configuration includes the following components:
    - Base-PP: Protection Profile for Application Software, Version 1.3 (PP_APP_V1.3)
    - PP-Module: PP-Module for File Encryption, Version 1.0 (MOD_FE_V1.0)
- Technical Decisions:

TD No.	PP	Applied	Rationale
TD0416	PP_APP_V1.3	Yes	
TD0427	PP_APP_V1.3	Yes	
TD0434	PP_APP_V1.3	No	Windows not applicable
TD0435	PP_APP_V1.3	No	Linux not applicable
TD0437	PP_APP_V1.3	Yes	
TD0444	PP_APP_V1.3	No	Not a VPN Client
TD0445	PP_APP_V1.3	Yes	
TD0455	MOD_FE_1.0	Yes	
TD0465	PP_APP_V1.3	No	Not a .NET application
TD0472	MOD_FE_V1.0	Yes	
TD0473	PP_APP_V1.3	No	HTTPS not applicable
TD0486	PP_APP_V1.3	Yes	
TD0495	PP_APP_V1.3	No	X509 not applicable
TD0498	PP_APP_V1.3	Yes	
TD0505	PP_APP_V1.3	No	X509 not applicable
TD0510	PP_APP_V1.3	No	MacOS or iOS not applicable

2.1 Conformance Rationale

The ST conforms to the PP_APP_V1.3/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_V1.3/MOD_FE_V1.0 and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The PP_APP_V1.3/MOD_FE_V1.0 offers additional information about the identified security objectives, but that has not been reproduced here and the PP_APP_V1.3/MOD_FE_V1.0 should be consulted if there is interest in that material.

In general, the PP_APP_V1.3/MOD_FE_V1.0 has defined Security Objectives appropriate for file encryption software application and as such are applicable to the KeyW Protect for Samsung 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 power down after a period of time in the same fashion as the user electing to shutdown the system (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 exists 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_V1.3/MOD_FE_V1.0. The PP_APP_V1.3/MOD_FE_V1.0 defines the following extended requirements and since they are not redefined in this ST the PP_APP_V1.3/MOD_FE_V1.0 should be consulted for more information in regard to those CC extensions.

Extended SFRs:

- PP_APP_V1.3: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_V1.3:FCS_RBG_EXT.1: Random Bit Generation Services
- PP_APP_V1.3:FCS_STO_EXT.1: Storage of Credentials
- MOD_FE_V1.0:FCS_VAL_EXT.1: Validation
- PP_APP_V1.3:FDP_DAR_EXT.1: Encryption Of Sensitive Application Data
- PP_APP_V1.3:FDP_DEC_EXT.1: Access to Platform Resources
- PP_APP_V1.3: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_V1.3:FMT_CFG_EXT.1: Secure by Default Configuration
- PP_APP_V1.3:FMT_MEC_EXT.1: Supported Configuration Mechanism
- PP_APP_V1.3:FPR_ANO_EXT.1: User Consent for Transmission of Personally Identifiable Information
- PP_APP_V1.3:FPT_AEX_EXT.1: Anti-Exploitation Capabilities
- PP_APP_V1.3:FPT_API_EXT.1: Use of Supported Services and APIs
- PP_APP_V1.3:FPT_IDV_EXT.1: Software Identification and Versions
- MOD_FE_V1.0:FPT_KYP_EXT.1: Protection of Keys and Key Material
- PP_APP_V1.3:FPT_LIB_EXT.1: Use of Third Party Libraries
- PP_APP_V1.3:FPT_TUD_EXT.1: Integrity for Installation and Update
- PP_APP_V1.3:FPT_TUD_EXT.2: Integrity for Installation and Update
- PP_APP_V1.3: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_V1.3/MOD_FE_V1.0. The refinements and operations already performed in the PP_APP_V1.3/MOD_FE_V1.0 are not identified (e.g., highlighted) here, rather the requirements have been copied from the PP_APP_V1.3/MOD_FE_V1.0 and any residual operations have been completed herein. Of particular note, the PP_APP_V1.3/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_V1.3/MOD_FE_V1.0 which includes all the SARs for EAL 1. However, the SARs are effectively refined since requirement-specific 'Assurance Activities' are defined in the PP_APP_V1.3/MOD_FE_V1.0 that serve to ensure corresponding evaluations will yield more practical and consistent assurance than the EAL 1 assurance requirements alone. The PP_APP_V1.3/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 KeyW Protect for Samsung TOE.

Requirement Class	Requirement Component
FCS: Cryptographic support	PP_APP_V1.3:FCS_CKM_EXT.1: Cryptographic Key Generation Services
	PP_APP_V1.3:FCS_CKM.1(1): Cryptographic Asymmetric Key Generation
	PP_APP_V1.3:FCS_CKM.2: Cryptographic Key Establishment
	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_V1.3:FCS_COP.1(1): Cryptographic Operation - Encryption/Decryption
	PP_APP_V1.3:FCS_COP.1(4): Cryptographic Operation - Keyed-Hash Message Authentication
	MOD_FE_V1.0:FCS_COP.1(5): Cryptographic operation (Key Wrapping)
	MOD_FE_V1.0:FCS_COP.1(7): Cryptographic operation (Key Encryption)
	MOD_FE_V1.0:FCS_SMC_EXT.1: Submask Combining
	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_V1.3:FCS_RBG_EXT.2: Random Bit Generation from Application
	PP_APP_V1.3:FCS_STO_EXT.1: Storage of Credentials
	MOD_FE_V1.0:FCS_VAL_EXT.1: Validation
FDP: User data protection	PP_APP_V1.3:FDP_DAR_EXT.1: Encryption Of Sensitive Application Data
	PP_APP_V1.3:FDP_DEC_EXT.1: Access to Platform Resources
	PP_APP_V1.3: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

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

Table 5-1 TOE Security Functional Components

5.1.1 Cryptographic support (FCS)

5.1.1.1 Cryptographic Key Generation Services (PP_APP_V1.3:FCS_CKM_EXT.1)

PP_APP_V1.3:FCS_CKM_EXT.1.1
The application shall [***implement asymmetric key generation***].

5.1.1.2 Cryptographic Asymmetric Key Generation (FCS_CKM.1(1))

PP_APP_V1.3:FCS_CKM.1(1)
The application shall [***implement functionality***] to generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [***ECC schemes using 'NIST curves' P-256, P-384 and [P-521] that meet the following: FIPS PUB 186-4, 'Digital Signature Standard (DSS)', Appendix B.4***].

5.1.1.3 Cryptographic Key Establishment (FCS_CKM.2)

PP_APP_V1.3:FCS_CKM.2.1
The application shall [***implement functionality***] to perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: **[*Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"*].**

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 the [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 [single overwrite consisting of [zeroes]

].

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 [***[16]***] 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 [[+ = _ / - ' " ***: ; , ?*** ` ~ \ | < > ***{ } [ ]***].

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-384***], with [***12345***] 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_V1.3:FCS_COP.1(1))

PP_APP_V1.3:FCS_COP.1.1(1)

The application shall perform encryption/decryption in accordance with a specified cryptographic algorithm [***AES-XTS (as defined in NIST SP 800-38E) mode***] and cryptographic key sizes [***256 bits***].

5.1.1.9 Cryptographic Operation - Keyed-Hash Message Authentication (PP_APP_V1.3:FCS_COP.1(4))

PP_APP_V1.3:FCS_COP.1.1(4)

The application shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-SHA-256 and [***SHA-384***] with key sizes [***256***] and message digest sizes 256 and [***384***] bits that meet the following: FIPS PUB 198-1 The Keyed-Hash Message Authentication Code and FIPS PUB 180-4 Secure Hash Standard.

5.1.1.10 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 the cryptographic key sizes [***256 bits (AES)***] that meet the following: [“***NIST SP 800-38F”***] and no other standards.

5.1.1.11 Cryptographic operation (Key Encryption) (MOD_FE_V1.0:FCS_COP.1(7))

MOD_FE_V1.0:FCS_COP.1.1(7)

The TSF shall [***use platform-provided functionality to perform Key Encryption***] in accordance with a specified cryptographic algorithm [*AES used in CBC mode*] and cryptographic key sizes [***256***] bits that meet the following: [*AES as specified in SP 800-38A*].

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

MOD_FE_V1.0:FCS_IV_EXT.1.1

The TSF shall [

- ***invoke platform-provided functionality to generate IVs,***
- ***generate IVs with the following properties [XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer]***

].

5.1.1.13 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): [***
  - ***utilization of the platform key storage,***
  - ***implementation of key wrapping as specified in FCS_COP.1(5),***
  - ***implementation of key encryption as specified in FCS_COP.1(7)***

  ] while maintaining an effective strength of [[256 bits] for symmetric keys] commensurate with the strength of the FEK

] and [

- ***other supplemental key chains that protect a key or keys in the primary key chain using the following method(s): [***
  - ***implementation of key combining as specified in FCS_SMC_EXT.1,***

  ]

].

5.1.1.14 Random Bit Generation Services (PP_APP_V1.3:FCS_RBG_EXT.1)

PP_APP_V1.3:FCS_RBG_EXT.1.1

The application shall [***implement DRBG functionality***] for its cryptographic operations.

5.1.1.15 Random Bit Generation from Application (PP_APP_V1.3:FCS_RBG_EXT.2)

PP_APP_V1.3: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 [***CTR_DRBG (AES)***].

PP_APP_V1.3: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.

5.1.1.16 Submask Combining (MOD_FE_V1.0:FCS_SMC_EXT.1)

MOD_FE_V1.0:FCS_SMC_EXT.1.1
The TSF shall [combine submasks using the following method [***SHA-256, HMAC-SHA-384***] to generate an intermediate key].

5.1.1.17 Storage of Credentials (PP_APP_V1.3:FCS_STO_EXT.1)

PP_APP_V1.3:FCS_STO_EXT.1.1
The application shall [***not store any credentials***] 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].

5.1.2 User data protection (FDP)

5.1.2.1 Encryption Of Sensitive Application Data (PP_APP_V1.3:FDP_DAR_EXT.1)

PP_APP_V1.3:FDP_DAR_EXT.1.1
The application shall [***implement functionality to encrypt sensitive data as defined in the EP for File Encryption***] in non-volatile memory.

5.1.2.2 Access to Platform Resources (PP_APP_V1.3:FDP_DEC_EXT.1)

PP_APP_V1.3:FDP_DEC_EXT.1.1
The application shall restrict its access to [***[no hardware resources]***].
PP_APP_V1.3:FDP_DEC_EXT.1.2
The application shall restrict its access to [***[shared files]***].

5.1.2.3 Network Communications (PP_APP_V1.3:FDP_NET_EXT.1)

PP_APP_V1.3:FDP_NET_EXT.1.1
The application shall restrict network communication to [***no network communication***].

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 application shall [***implement platform-provided functionality to 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_V1.3:FMT_CFG_EXT.1)

PP_APP_V1.3: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_V1.3:FMT_CFG_EXT.1.2
The application shall be configured by default with file permissions which protect the application binaries and data files from modification by normal unprivileged users.

5.1.4.2 Supported Configuration Mechanism (PP_APP_V1.3:FMT_MEC_EXT.1)

PP_APP_V1.3:FMT_MEC_EXT.1.1
The application shall [***invoke the mechanisms recommended by the platform vendor for storing and setting configuration options***]. (TD0437 applied)

5.1.4.3 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 authentication factors,***
- ***perform a cryptographic erase of the data by the destruction of FEKs or KEKs protecting the FEKs as described in FCS_CKM_EXT.4.1,***
- ***configure the number of failed validation attempts required to trigger corrective behavior,***
- ***configure the corrective behavior to issue in the event of an excessive number of failed validation attempts,***

- [***configure password/passphrase complexity setting***]

].

5.1.4.4 Specification of Management Functions (PP_APP_V1.3:FMT_SMF.1)

PP_APP_V1.3:FMT_SMF.1.1

The TSF shall be capable of performing the following management functions [***no management functions beyond those in MOD_FE_V1.0:FMT_SMF.1(2)***].

5.1.5 Privacy (FPR)

5.1.5.1 User Consent for Transmission of Personally Identifiable Information (PP_APP_V1.3:FPR_ANO_EXT.1)

PP_APP_V1.3: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_V1.3:FPT_AEX_EXT.1)

PP_APP_V1.3:FPT_AEX_EXT.1.1

The application shall not request to map memory at an explicit address except for [***none***].

PP_APP_V1.3:FPT_AEX_EXT.1.2

The application shall [***not allocate any memory region with both write and execute permissions***].

PP_APP_V1.3:FPT_AEX_EXT.1.3

The application shall be compatible with security features provided by the platform vendor.

PP_APP_V1.3: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_V1.3: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_V1.3:FPT_API_EXT.1)

PP_APP_V1.3:FPT_API_EXT.1.1

The application shall use only documented platform APIs.

5.1.6.3 Software Identification and Versions (PP_APP_V1.3:FPT_IDV_EXT.1)

PP_APP_V1.3:FPT_IDV_EXT.1.1

The application shall be versioned with [***[a multi-part 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 when [***

- ***wrapped, as specified in FCS_COP.1(5),***
- ***encrypted, as specified in FCS_COP.1(7)***

]].

5.1.6.5 Use of Third Party Libraries (PP_APP_V1.3:FPT_LIB_EXT.1)

PP_APP_V1.3:FPT_LIB_EXT.1.1

The application shall be packaged with only [**no third-party libraries**].

5.1.6.6 Integrity for Installation and Update (PP_APP_V1.3:FPT_TUD_EXT.1)

PP_APP_V1.3:FPT_TUD_EXT.1.1
The application shall [***leverage the platform***] to check for updates and patches to the application software.

PP_APP_V1.3:FPT_TUD_EXT.1.2
The application shall [***leverage the platform***] to query the current version of the application software.

PP_APP_V1.3:FPT_TUD_EXT.1.3
The application shall not download, modify, replace or update its own binary code.

PP_APP_V1.3:FPT_TUD_EXT.1.4
The application installation package and its updates shall be digitally signed such that its platform can cryptographically verify them prior to installation.

PP_APP_V1.3: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 (PP_APP_V1.3:FPT_TUD_EXT.2)

PP_APP_V1.3:FPT_TUD_EXT.2.1
The application shall be distributed using the format of the platform-supported package manager.

PP_APP_V1.3: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.

5.1.7 Trusted path/channels (FTP)

5.1.7.1 Protection of Data in Transit (PP_APP_V1.3:FTP_DIT_EXT.1)

PP_APP_V1.3:FTP_DIT_EXT.1.1
The application shall [***not transmit any [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: Labelling 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 5-2 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 a Samsung mobile device with Android OS 9.0, Knox 3.3, and DualDAR 1.0 and uses features provided by the platform for key storage. The mobile device utilizes the Samsung Exynos 9820 and Qualcomm Snapdragon 855 (SDM855) processors.

The TOE includes the *Suite B Cryptographic Algorithms* library, which implements its own algorithms for random bit generation, AES encryption/decryption, AES key wrapping, keyed-hashing functions, password-based key derivation, key pair generation, key establishment and cryptographic hashing. This library provides the following cryptographic services:

Functions	Standards	CAVP Certificates
Key Generation		
ECC key generation schemes using 'NIST curves' P-256, P-384 and P-521	FIPS Pub 186-4 Appendix B.4	C1412
Key Establishment		
Elliptic curve-based key establishment schemes	NIST SP 800-56A	C1412
Encryption/Decryption		
AES-XTS (256 bits)	FIPS PUB 197, NIST SP 800-38E	C1412
Cryptographic Hashing		
SHA-256 SHA-384	FIPS Pub 180-4	C1412
Keyed-Hash Message Authentication		
HMAC-SHA-256 HMAC-SHA-384	FIPS Pub 198-1, FIPS Pub 180-4	C1412
Key Wrapping		
AES Key Wrap (256-bit)	NIST SP 800-38F	C1412
Password-Based Key Derivation		
HMAC-SHA-384	NIST SP 800-132	Vendor-Affirmed
Random Bit Generation		
AES-CTR-DRBG (256 bits)	NIST SP 800-90A	C1412

Table 6-1 Cryptographic Functions, Standards and CAVP Certificates

The Cryptographic support function satisfies the following security functional requirements:

- PP_APP_V1.3:FCS_CKM_EXT.1: The TOE generates a static 521-bit ECC Client keypair using the “Key Pair Generation by Testing Candidates” method. The TOE also generates an ephemeral 521-bit ECC Native keypair using the “Key Pair Generation by Testing Candidates” method.

- PP_APP_V1.3:FCS_CKM.1(1): The TOE implements asymmetric key generation using ECC schemes and NIST curves P-256, P-384 and P-521 that meet FIPS PUB 186-4 'Digital Signature Standard (DSS)' Appendix B.4.
- PP_APP_V1.3:FCS_CKM.2: The TOE performs key establishment using ECC key pairs for protection of communication between Native and Java clients running in the same platform. The TOE uses P-521 keys during this key establishment.
- MOD_FE_V1.0:FCS_CKM_EXT.2: The TOE automatically generates a new FEK whenever an application inside the Android Enterprise workspace creates a new file. The FEK never changes during the life of a file. The TOE obtains 256-bits of random bits from the *Suite B Cryptographic Algorithms* library's DRBG, which becomes the new FEK for the file.
- MOD_FE_V1.0:FCS_CKM_EXT.3: The TOE utilizes the user's workspace password to derive a 256-bit HASH KEK and obtains a 256-bit Master KEK from the *Suite B Cryptographic Algorithms* library's DRBG. The 256-bit Master KEK is stored encrypted by the 256-bit HASH KEK in the private shared preferences file.
- MOD_FE_V1.0:FCS_CKM_EXT.4: When a user workspace password is obtained by the TOE application from the Samsung Knox DualDAR API, the TOE uses the *Suite B Cryptographic Algorithms* library's CleanUp service to zeroize the buffer holding the workspace password. The workspace password is no longer needed in volatile memory once the 256-bit HASH KEK can be derived from the workspace password.

  When the TOE application obtains a KEK from the private shared preferences file, the TOE performs a read/verify after zeroing volatile memory into which the KEK was read. The TOE invokes the appropriate *Suite B Cryptographic Algorithms* library's CleanUp service to zeroize a KEK when it is no longer needed. The 256-bit HASH KEK is no longer needed in volatile memory once the application is "Data Locked". The 256-bit Master KEK is no longer needed in volatile memory once the application has completed using it to unwrap the FEK of a file. The FEK is cleared when the file is deleted.
- MOD_FE_V1.0:FCS_CKM_EXT.6: A user workspace password prompt must be presented to the user immediately following the installation of the TOE on the Samsung mobile device and prior to decryption of any file. The workspace password is used to derive a 256-bit HASH KEK by applying the PBKDF2 algorithm to the workspace password, meeting SP 800-132. This derivation uses an HMAC-SHA-384 algorithm, 12345 iterations, and a 256-bit SALT value to create the 256-bit HASH KEK. The SALT value is obtained from the *Suite B Cryptographic Algorithms* library's DRBG. The workspace password must be between 4-16 characters and is configurable via the UEM console. The available character set is listed in 5.1.1.7.
- PP_APP_V1.3:FCS_COP.1(1): The TOE implements its own AES in the XTS mode, with key sizes of 256 bits as defined by NIST SP 800-38E (refer to

Random Bit Generation		
AES-CTR-DRBG (256 bits)	NIST SP 800-90A	C1412

- **Table 6-1**). Users do not choose an encryption mode; all FEKs in the system are 256-bit AES-XTS mode keys.
- PP_APP_V1.3:FCS_COP.1(4): The TOE implements its own Keyed-Hash for HMAC-SHA-256 and HMAC-SHA-384 that is compliant with FIPS Pub 180-4 and FIPS Pub 198-1 (refer to

Random Bit Generation		
AES-CTR-DRBG (256 bits)	NIST SP 800-90A	C1412

- **Table 6-1**). The TOE performs HMAC-SHA-384 when performing the password-based key derivation using SHA-384, with a key that is the length of the workspace password and output length of 256.
- MOD_FE_V1.0:FCS_COP.1(5): The TOE performs key wrapping of the 256-bit Master KEK and FEK using an AES 256 bit key wrap conformant to NIST SP 800-38F (refer to

Random Bit Generation		
AES-CTR-DRBG (256 bits)	NIST SP 800-90A	C1412

- **Table 6-1**).
- MOD_FE_V1.0:FCS_COP.1(7): The TOE uses platform services for AES/CBC key encryption of the static 521-bit ECC Client private key, 256-bit Temporary HASH KEK, 256-bit SALT, and 256-bit Reset Token SALT using non-exportable 256-bit AES KEKs from the platform key store conformant to NIST SP 800-38A.
- MOD_FE_V1.0:FCS_IV_EXT.1: The TOE invokes platform-provided functionality to generate Initialization Vectors (IV) for AES/CBC key encryption. The TOE generates Initialization Vectors (IV) using the AES CTR-DRBG provided by the *Suite B Cryptographic Algorithms* library's DRBG. The TOE uses tweaks from the approved DRBG when encrypting file data using a FEK.
- MOD_FE_V1.0:FCS_KYC_EXT.1: The 256-bit Master KEK and FEK are both 256-bit AES keys, protected using AES Key Wrap when stored. The 256-bit Master KEK is stored encrypted in the private shared preferences file wrapped by the 256-bit HASH KEK derived from the user's workspace password (using PBKDF2) and a 256-bit SALT value. The FEK is stored encrypted in the associated metadata of the file it protects. The FEK is stored only after being AES Key Wrapped by the 256-bit Master KEK. The 256-bit HASH KEK is stored encrypted in the Secrets metadata contained in volatile memory by a 256-bit Secret KEK derived from the Shared Secret (SS) using SHA-256. The 256-bit SALT value is stored encrypted in the private shared preferences file encrypted by a non-exportable 256-bit AES KEK from the platform key store using AES in CBC mode.
- PP_APP_V1.3:FCS_RBG_EXT.1: The TOE uses the *Suite B Cryptographic Algorithms* library's DRBG source internal to the TOE, which is configured for AES CTR-DRBG (256 bits).
- PP_APP_V1.3:FCS_RBG_EXT.2: The TOE performs all DRBG services via the *Suite B Cryptographic Algorithms* library's DRBG source internal to the TOE, which is configured for AES CTR-DRBG (256 bits). The *Suite B Cryptographic Algorithms* library's DRBG source leverages the platform-based Non-Deterministic Random Number Generator (NDRNG) outside the logical cryptographic boundary within the physical cryptographic boundary as the Source of Entropy Input (SEI) for seeding/reseeding purposes by reading the minimum entropy and nonce from /dev/random in accordance with NIST SP 800-90A.
- MOD_FE_V1.0:FCS_SMC_EXT.1: The TOE derives a 256-bit Secret KEK from the Shared Secret (SS) using SHA-256. The TOE derives a 256-bit HASH KEK by applying the PBKDF2 algorithm to the workspace password, meeting SP 800-132, which uses the HMAC-SHA-384 algorithm. These intermediate keys are the same strength as the FEK.
- PP_APP_V1.3:FCS_STO_EXT.1: The TOE does not store any credentials.
- MOD_FE_V1.0:FCS_VAL_EXT.1: Prior to decrypting any FEK for access to file data, the TOE obtains a workspace password from the user. The TOE then derives a 256-bit HASH KEK from the workspace password (using PBKDF2) and a 256-bit SALT value (stored encrypted by a non-exportable 256-bit AES KEK from the platform key store in the private shared preferences file). This 256-bit HASH KEK is then used to unwrap the 256-bit Master KEK that was stored in the private shared preferences file. If the given workspace password is invalid, the derived 256-bit HASH KEK will be unable to unwrap the 256-bit Master KEK. Thus, a change to a user's workspace password requires the old workspace password to unwrap the 256-bit Master KEK, and a new workspace password to provide the new wrap for the 256-bit Master KEK.

6.2 User data protection

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

- PP_APP_V1.3:FDP_DAR_EXT.1: The TOE provides its own CAVP certified cryptographic algorithms to encrypt all files in the Samsung mobile device's Android Enterprise workspace directory structure. The files and associated metadata are the sensitive data.

- PP_APP_V1.3:FDP_DEC_EXT.1: After installation, the TOE indicates its intent to access shared files.
- PP_APP_V1.3:FDP_NET_EXT.1: The TOE does not transmit any information over a network.
- MOD_FE_V1.0:FDP_PM_EXT.1: The TOE has three states: Powered-off, Data Locked, and Data Unlocked. In the Powered-off state, the device is off and TOE is not available.

  On startup, the device places the TOE into the Data Locked state. From the Data Locked state, the user must authenticate successfully to Android Enterprise workspace to transition to the Data Unlocked state. In the Data Unlocked phase, the passphrase is used to unwrap the 256-bit Master KEK which is then used to decrypt the FEK.

  The "Data Locked" state is when the screen is locked, workspace password locked or an inactivity period lapsed. 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 workspace password) to decrypt files following a mobile device entering the "Data Locked" state.
- MOD_FE_V1.0:FDP_PRT_EXT.1: The TOE automatically encrypts all files stored in the Android Enterprise workspace directories upon the file's creation. No temporary files are used during the encryption/decryption process as all file content processing is done with in-place buffers. A workspace password is required in order to decrypt the contents of a file. The TOE encrypts only the file contents and not the file metadata. The Samsung Knox DualDAR API is used to relay all file operations within the Android Enterprise workspace to the TOE, which encrypts or decrypts the file contents automatically. The TOE encrypts all workspace data received by applications until the Samsung mobile device becomes data locked, so no clear text version of the files are ever created on the Android Enterprise workspace file system. When the device is data unlocked, the TOE encrypts all Android Enterprise workspace files as they are created by other applications.
- MOD_FE_V1.0:FDP_PRT_EXT.2: No temporary files are used during the encryption/decryption process as all file content processing is done with in-place buffers.

6.3 Identification and authentication

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

- MOD_FE_V1.0:FIA_AUT_EXT.1: The TOE utilizes the Samsung Knox DualDAR API to request user credentials. The TOE makes its own decisions regarding the correctness of the workspace password provided by the user and treats the workspace password as case-sensitive.

6.4 Security management

The Security management function satisfies the following security functional requirements:

- PP_APP_V1.3:FMT_CFG_EXT.1: After the TOE application has been installed, the Samsung mobile device user is prompted to define the first user workspace password. The TOE supports only a single mobile device user having one workspace password. The user may change the workspace password through the platform's interface. On the Samsung mobile device, a mobile device user has RW access to the directories containing shared files, thus allowing the application to have RW permission to the directories containing shared files.
- PP_APP_V1.3:FMT_MEC_EXT.1: The TOE uses a very limited set of configuration data. This includes only the private native state file that contains the native state file descriptor and a private shared preferences file that contains the following encrypted keys and CSPs:
  - client private key initialization vector
  - encrypted client private key
  - password salt initialization vector

- encrypted password salt
- temporary password hash initialization vector
- encrypted temporary password hash
- encrypted key encryption key (master key)

- MOD_FE_V1.0:FMT_SMF.1(2) and PP_APP_V1.3:FMT_SMF.1: The TOE allows the following management functions:
  - Change the workspace password used to authenticate the user and protect the 256-bit Master KEK. The TOE requires both the old and new workspace password during this change, which unwraps the 256-bit Master KEK using the old workspace password and wraps the 256-bit Master KEK using the new workspace password.
  - Reset workspace password using a reset token from the UEM console.
  - Configure password/passphrase complexity settings including the minimum and maximum lengths.
  - Perform a cryptographic erase of the data by the destruction of the 256-bit Master KEK.
  - Configure the corrective behavior (wipe/disable workspace) and number of failed validation attempts required to trigger corrective behavior.

6.5 Privacy

The Privacy function satisfies the following security functional requirements:

- PP_APP_V1.3:FPR_ANO_EXT.1: The TOE does not transmit any information over a network.

6.6 Protection of the TSF

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

- ALC_TSU_EXT.1: KeyW accepts bug reports (including reports for security vulnerabilities) through a Technical Support Contact form on the www.keywcorp.com/contact-us/ web site. KeyW reviews all bug reports when making product changes to resolve issues associated with the TOE. KeyW makes updates and code patches to resolve issues as quickly as possible then makes the updates available to customers. TOE updates are distributed by KeyW to customers, whom can then utilize the UEM console to distribute the update to individual devices.
- PP_APP_V1.3:FPT_AEX_EXT.1: The TOE does not map memory to a specific address, does not allocate memory with both write and execute permission and does not write user-modifiable files to locations which contain executables. The application is compiled with FSTACK protection STRONG. The TOE executes on Samsung mobile devices running Android OS 9.0 and utilizes security features from this platform.
- PP_APP_V1.3:FPT_API_EXT.1: The TOE utilizes the APIs identified throughout the TSS and those listed in Section 7.
- PP_APP_V1.3:FPT_IDV_EXT.1: The TOE labels its software using a multi-part unique release number that can be queried by users.
- MOD_FE_V1.0:FPT_KYP_EXT.1: All FEKs are wrapped using AES Key Wrap with the 256-bit Master KEK prior to being stored in the associated metadata alongside the file the FEK is protecting. The 256-bit Master KEK is stored encrypted in the private shared preferences file wrapped by the 256-bit HASH KEK derived from the user's workspace password (using PBKDF2) and a 256-bit SALT value. The 256-bit SALT value is stored encrypted in the private shared preferences file encrypted by a non-exportable 256-bit AES KEK from the platform key store using AES in CBC mode. The TOE never stores keys or key material in non-volatile memory in clear text form.

- PP_APP_V1.3:FPT_LIB_EXT.1: The TOE runs on a Samsung mobile device with Android OS 9.0, Knox 3.3, and DualDAR 1.0, utilizing libraries provided by the platform. The TOE does not utilize any other 3rd party libraries.
- PP_APP_V1.3:FPT_TUD_EXT.1/2: The TOE relies upon the platform to provide a mechanism that can check for product updates, to query the current version of the TOE, and to support the installation of an update. The TOE is installed via the UEM console.

6.7 Trusted path/channels

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

- PP_APP_V1.3:FTP_DIT_EXT.1: The TOE application does not transmit any sensitive data over the network. The only communication occurring as a result of TOE activity is the network traffic associated with checking for the currently available version of the TOE.

7. Platform API List

The following is a list of the platform APIs invoked by the TOE related to supporting the security functionality of the TOE.

- Samsung Knox SDK v3.3 API Reference

API	Reference
com.samsung.android.knox.ddar.DualDARClient	https://docs.samsungknox.com/devref/knox-sdk/reference/com/samsung/android/knox/ddar/DualDARClient.html
ddar::abstract_crypto "ddar.h"	https://docs.samsungknox.com/devref/knox-sdk/reference/native/html/classddar_1_1abstract__crypto.html

- Android OS 9.0 Platform API Reference

API	Reference
java.security.Key	https://developer.android.com/reference/java/security/Key
java.security.KeyStore	https://developer.android.com/reference/java/security/KeyStore
javax.crypto.Cipher	https://developer.android.com/reference/javax/crypto/Cipher
javax.crypto.KeyGenerator	https://developer.android.com/reference/javax/crypto/KeyGenerator
javax.crypto.SecretKey	https://developer.android.com/reference/javax/crypto/SecretKey
javax.crypto.spec.IvParameterSpec	https://developer.android.com/reference/javax/crypto/spec/IvParameterSpec
javax.crypto.spec.SecretKeySpec	https://developer.android.com/reference/javax/crypto/spec/SecretKeySpec
android.content.SharedPreferences	https://developer.android.com/reference/android/content/SharedPreferences
android.security.keystore.KeyGenParameterSpec	https://developer.android.com/reference/android/security/keystore/KeyGenParameterSpec
android.security.keystore.KeyProperties	https://developer.android.com/reference/android/security/keystore/KeyProperties
org.json.JSONObject	https://developer.android.com/reference/org/json/JSONObject
org.json.JSONTokener	https://developer.android.com/reference/org/json/JSONTokener
android.util.Log	https://developer.android.com/reference/android/util/Log

- Android OS 9.0 NDK API Reference

API	Reference
<linux/ashmem.h>	https://android.googlesource.com/platform/external/kernel-headers/+/refs/tags/android-9.0.0_r48/original/uapi/linux/ashmem.h
<sys/mman.h>	https://android.googlesource.com/platform/bionic/+/refs/tags/android-9.0.0_r48/libc/include/sys/mman.h
<fcntl.h>	https://android.googlesource.com/platform/bionic/+/refs/tags/android-9.0.0_r48/libc/include/fcntl.h
<sys/ioctl.h>	https://android.googlesource.com/platform/bionic/+/refs/tags/android-9.0.0_r48/libc/include/sys/ioctl.h
<unistd.h>	https://android.googlesource.com/platform/bionic/+/refs/tags/android-9.0.0_r48/libc/include/unistd.h
<android/log.h>	https://developer.android.com/ndk/reference/group/logging