www.GossamerSec.com

Assurance Activity Report for Samsung Electronics Co., Ltd. Samsung KNOX File Encryption 1.5 - Spring

Version 0.2
03/17/2023

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

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

Document: AAR-VID11344

Revision History

Revision	Date	Authors	Summary
Version 0.1	02/14/2023	Compton	Initial draft
Version 0.2	03/17/2023	Compton	Addressed ECR comments

The TOE Evaluation was Sponsored by:

Samsung Electronics Co., Ltd.
416 Maetan-3dong, Yeongtong-gu, Suwon-si, Gyeonggi-do, 443-742 Korea

Evaluation Personnel:

- James Arnold
- Tammy Compton

Common Criteria Versions:

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

Common Evaluation Methodology Versions:

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

Table of Contents

1. Introduction

This document presents evaluations results of the Samsung Knox File Encryption 1.5 - Spring PP_APP_V1.4/MOD_FE_V1.0 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 References

The following evidence was used to complete the Assurance Activities:

- Samsung Electronics Co., Ltd. Samsung Knox File Encryption 1.5 - Spring Security Target, Version 0.4, March 17, 2023 (**ST**)
- Samsung Electronics Co., Ltd. Samsung Knox File Encryption 1.5 Key Management Description - Spring, Version 0.4, March 17, 2023 (**KMD**)
- Samsung File Encryption 1.5 Administrator Guide, Version 1.5, March 17, 2023 (**Admin Guide**)

1.2 Evaluated Platforms

The following table shows the model numbers of the mobile devices included in the evaluation of Knox File Encryption 1.5 (the version is listed as "DualDAR"):

Device Name	Model Number	Chipset Vendor	CPU	Android Version	TEE OS	Knox Version	DualDAR Version
Galaxy S23 Ultra 5G	SM-S918U	Qualcomm	Snapdragon 8 Gen 2 Mobile Platform	13	QSEE 5.24	3.9	1.5.1
Galaxy S22 Ultra 5G	SM-S908B	Samsung	Exynos 2200	13	TEEGRIS 4.3	3.9	1.5.1
Galaxy S22 Ultra 5G	SM-S901U	Qualcomm	Snapdragon 8 Gen 1 Mobile Platform	13	QSEE 5.12	3.9	1.5.1
Galaxy S21 Ultra 5G	SM-G998B	Samsung	Exynos 2100	13	TEEGRIS 4.2	3.9	1.5.1
Galaxy S21 Ultra 5G	SM-G998U	Qualcomm	Snapdragon 888	13	QSEE 5.11	3.9	1.5.1
Galaxy S20+ 5G	SM-G986B	Samsung	Exynos 990	13	TEEGRIS 4.1	3.9	1.5.0
Galaxy S20+ 5G	SM-G986U	Qualcomm	Snapdragon 865	13	QSEE 5.8	3.9	1.5.0

Table 1 - Evaluated Devices

In addition to the evaluated devices, the following device models are claimed as equivalent with a note about the differences between the evaluated device and the equivalent models.

Evaluated Device	CPU	Equivalent Devices	Differences
Galaxy S23 Ultra 5G	Snapdragon 8 Gen 2 Mobile Platform	Galaxy S23+ 5G Galaxy S23 5G	All devices have same software environments (Android, Kernel, TEE)
Galaxy S22 Ultra 5G	Exynos 2200	Galaxy S22+ 5G Galaxy S22 5G	All devices have same software environments (Android, Kernel, TEE)

Galaxy S22 5G	Snapdragon 8 Gen 1 Mobile Platform	Galaxy S22 Ultra 5G	All devices have same software environments (Android, Kernel, TEE)
		Galaxy S22+ 5G	
		Galaxy Tab S8 Ultra	
		Galaxy Tab S8+	
		Galaxy Tab S8	
Galaxy S21 Ultra 5G	Exynos 2100	Galaxy S21+ 5G	All devices have same software environments (Android, Kernel, TEE)
		Galaxy S21 5G	
Galaxy S21 Ultra 5G	Snapdragon 888	Galaxy S21+ 5G	All devices have same software environments (Android, Kernel, TEE)
		Galaxy S21 5G	
		Galaxy S21 5G FE	
		Galaxy Z Fold3 5G	
		Galaxy Z Flip3 5G	
Galaxy S20+ 5G	Exynos 990	Galaxy Note20 Ultra 5G	All devices have same software environments (Android, Kernel, TEE)
		Galaxy Note20 Ultra LTE	
		Galaxy Note20 5G	
		Galaxy Note20 LTE	
		Galaxy S20 Ultra 5G	
		Galaxy S20+ LTE	
		Galaxy S20 5G	
		Galaxy S20 LTE	
		Galaxy S20 FE	
Galaxy S20+ 5G	Snapdragon 865	Galaxy Z Fold2 5G	All devices have same software environments (Android, Kernel, TEE)
		Galaxy Note20 Ultra 5G	
		Galaxy Note20 5G	
		Galaxy Tab S7+	
		Galaxy Tab S7	
		Galaxy Z Flip 5G	
		Galaxy S20 Ultra 5G	
		Galaxy S20 5G	
		Galaxy S20 FE	
		Galaxy Note10+	
		Galaxy Note10	
		Galaxy Tab S6	
		Galaxy S10 5G	
		Galaxy S10	
		Galaxy S10e	
		Galaxy Fold 5G	
		Galaxy Fold	
		Galaxy Z Flip	

1.3 CAVP CERTIFICATE MAPPING

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

The evaluated devices utilize the following kernels for the Samsung Kernel Cryptographic Module (Kernel Crypto).

SoC	Kernel Version	Kernel Crypto Version
Qualcomm Snapdragon 8 Gen 2 Mobile Platform	5.15	2.3
Samsung Exynos 2200 Qualcomm Snapdragon 8 Gen 1 Mobile Platform	5.10	2.3
Samsung Exynos 2100 Qualcomm Snapdragon 888	5.4	2.2
Samsung Exynos 990 Qualcomm Snapdragon 865	4.19	2.1

Table 2 - Kernel Versions

The Samsung Kernel Cryptographic ("Kernel Crypto") Module provides the following algorithms used by the TOE.

Algorithm	NIST Standard	SFR Reference	Cert#
AES 256 CBC	FIPS 197, SP 800-38A	FCS_COP.1(1)	A3242
			A1456, A1455
			A970, A969
			C1394, C1393[1]
HMAC SHA-256	FIPS 198-1 & 180-4	FCS_COP.1(4)	A3242
			A1456, A1455
			A970, A969
			C1394, C1393

Table 3 - Samsung Kernel Cryptographic Algorithms

The evaluated devices utilize the Samsung SCrypto Cryptographic Module for cryptographic operations within the TEE on each device. The following table lists the TEE operating systems for each device.

SoC	TEE OS Version	SCrypto Version
Snapdragon 8 Gen 2 Mobile Platform	QSEE 5.24	2.7
Samsung Exynos 2200	TEEGRIS 4.3	2.6
Samsung Exynos 2100	TEEGRIS 4.2	2.6
Samsung Exynos 990	TEEGRIS 4.1	2.5
Snapdragon 8 Gen 1 Mobile Platform	QSEE 5.12	2.6
Qualcomm Snapdragon 888	QSEE 5.11	2.5
Qualcomm Snapdragon 865	QSEE 5.8	2.5

[1] Note that the SM8250 is another name for the Snapdragon 865.

Table 4 - TEE Environments

The Samsung SCrypto TEE library provides the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
AES CBC/GCM 256	FIPS 197, SP 800-38A/D	FCS_COP.1(1)	A3243, A915 A889, C1360

Table 5 - SCrypto TEE Cryptographic Algorithms

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profile and Extended Packages. This section also describes the findings for each activity. The TOE claims conformance to the Protection Profile for Application Software, Version 1.4, 07 October 2021 2021 (ASPP14) and the PP-Module for File Encryption, Version 1.0, 25 July 2019 (FE10).

The evidence identified in Section 1.1, References, was used to perform these Assurance Activities.

2.1 Cryptographic support (FCS)

2.1.1 Cryptographic Key Generation Services (ASPP14:FCS_CKM_EXT.1)

2.1.1.1 ASPP14:FCS_CKM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the generate no asymmetric cryptographic keys selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.

Section 6.1 of the ST states the TOE does not generate or use asymmetric keys.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.2 Password Conditioning (ASPP14:FCS_CKM_EXT.1/PBKDF)

2.1.2.1 ASPP14:FCS_CKM_EXT.1.1/PBKDF

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 ASPP14:FCS_CKM_EXT.1.2/PBKDF

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1/KeyedHash). No explicit testing of the formation of the submask from the input password is required. FCS_CKM.1.1/PBKDF: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1.

Section 6.1 of the ST states the TOE conditions the user password using a PBKDF2 (using HMAC-SHA-256) function that meets NIST SP 800-132 with 100,000 rounds to generate a 256-bit KEK to protect the Master Key (MKDD). All salts are generated using RBGs as specified in FCS_RBG_EXT.1. The description identifies the algorithm and settings (256-bit HMAC-SHA).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.3 Cryptographic Symmetric Key Generation (ASPP14:FCS_CKM.1/SK)

2.1.3.1 ASPP14:FCS_CKM.1.1/SK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked.

If the application is relying on random bit generation from the host platform, the evaluator shall verify the TSS includes the name/manufacturer of the external RBG and describes the function call and parameters used when calling the external DRBG function. If different external RBGs are used for different platforms, the evaluator shall verify the TSS identifies each RBG for each platform. Also, the evaluator shall verify the TSS includes a short description of the vendor's assumption for the amount of entropy seeding the external DRBG. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data.

Section 6.1 of the ST explains the TOE utilizes a number of different RBGs included in the platform:

1. An AES-256 CTR_DRBG provided by SCrypto in the TEE OS
2. A SHA-256 HMAC_DRBG provided by Kernel Crypto in the Android Linux kernel.

Each of these entropy sources will generate a 256-bit string. The APIs used by the TOE are available to applications that are part of the system services (not general Android applications). The platform entropy pools are seeded continually from the hardware noise source to ensure sufficient entropy is available for the generation of keys.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.4 File Encryption Key (FEK) Generation (FE10:FCS_CKM_EXT.2)

2.1.4.1 FE10:FCS_CKM_EXT.2.1

TSS Assurance Activities: The evaluator shall review the TSS to determine that a description covering how and when the FEKs are generated exists. The description must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate the FEKs. The evaluator shall verify that the description of how the FEKs are generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account.

Conditional: If 'using a Random Bit Generator' was selected, the evaluator shall verify that the TSS describes how the functionality described by FCS_RBG_EXT.1 (from the [AppPP]) is invoked to generate FEK. To the extent possible from the description of the RBG functionality in FCS_RBG_EXT.1 (from [AppPP]), the evaluator shall determine that the key size being requested is identical to the key size and mode to be used for the decryption/encryption of the user data (FCS_COP.1(1)) (from [AppPP]). (Per TD0455)

Conditional: If 'derived from a password/passphrase' is selected, the examination of the TSS section is performed as part of FCS_CKM_EXT.6 evaluation activities.

Section 6.1 of the ST states the TOE generates FEKs utilizing the Samsung Kernel Cryptographic Module that is part of the platform. The module provides random strings using SHA-256 HMAC_DRBG, seeded from /dev/random with sufficient entropy to generate keys with 256-bit security strength.

The TOE automatically generates a new FEK every time a new file is created (as all files within the work profile will be encrypted). The FEK is stored (encrypted by the MKDD with AES-GCM) as part of a metadata header attached to the file. No user interaction is required to generate the FEK or to encrypt any file within the work profile.

The ST discussion is consistent with Section 2.4 of the Admin Guide that states the user does not directly interact with the File Encryption service. The user interacts with the Knox work profile, which then automatically encrypts all data stored within the work profile boundary.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 FE10:FCS_CKM_EXT.2.2

TSS Assurance Activities: The evaluator shall verify the TSS describes how a FEK is used for a protected resource and associated with that resource. The evaluator confirms that-per this description-the FEK is unique per resource (file or set of files) and that the FEK is established using the mechanisms specified in FCS_CKM_EXT.2.1).

Section 6.1 of the ST states the TOE utilizing the Samsung Kernel Cryptographic Module that is part of the platform. The module provides random strings using SHA-256 HMAC_DRBG, seeded from /dev/random with sufficient entropy to generate keys with 256-bit security strength.

The TOE automatically generates a new FEK every time a new file is created (as all files within the work profile will be encrypted). The FEK is stored (encrypted by the MKDD with AES-GCM) as part of a metadata header attached to the file.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall review the instructions in the AGD guidance to determine that any explicit actions that need to be taken by the user to establish a FEK exist-taking into account any differences that arise from different platforms-and are consistent with the description in the TSS.

Section 2.4 of the Admin Guide that states the user does not directly interact with the File Encryption service. The user interacts with the Knox work profile, which then automatically encrypts all data stored within the work profile boundary.

Component Testing Assurance Activities: None Defined

2.1.5 Key Encrypting Key (KEK) Support (FE10:FCS_CKM_EXT.3)

2.1.5.1 FE10:FCS_CKM_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall review the TSS to determine that a description covering how and when KEK(s) are generated exists. The description must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate the KEKs. The evaluator shall verify that the description of how the KEK(s) are generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account.

Conditional: If using a RBG was selected the evaluator shall examine the TSS and verify that it describes how the functionality described by FCS_RBG_EXT.1 (from the [AppPP]) is invoked to generate KEK(s). To the extent possible from the description of the RBG functionality in FCS_RBG_EXT.1 (from [AppPP]), the evaluator shall determine that the key size being requested is identical to the key size selected.

Conditional: If derived from a password/passphrase is selected the examination of the TSS section is performed as part of FCS_CKM_EXT.6 evaluation activities.

Section 6.1 of the ST states the TOE generates two KEKs. The first KEK is the PMKEK and the second KEK is the MKDD. The PMKEK is derived, via PBKDF2 (HMAC-SHA-256), from the password entered by the user at successful authentication to either the Knox work profile or Knox File Encryption. This is examined in FCS_CKM_EXT.6.

The MKDD is generated using AES-256 CTR_DRBG from the SCrypto module, seeded by raw output from the hardware noise source (similar to /dev/random inside the TEE) with sufficient entropy to generate keys with 256-bit strength. The key is then encrypted by the PMKEK to be stored in non-volatile memory. Both KEKs are 256-bit.

Component Guidance Assurance Activities: The evaluator shall review the instructions in the AGD guidance to determine that any explicit actions that need to be taken by the user to establish a KEK exist-taking into account any differences that arise from different platforms-and are consistent with the description in the TSS.

Section 2.1 of the Admin Guide states the service is designed to run without any user intervention as all files in the Knox work profile will be encrypted automatically. No explicit actions are required.

Component Testing Assurance Activities: None Defined

2.1.6 Cryptographic Key Destruction - per TD0644 (FE10:FCS_CKM_EXT.4)

2.1.6.1 FE10:FCS_CKM_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.6.2 FE10:FCS_CKM_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when they should be expected to be destroyed. The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when then should be expected to be destroyed.

KMD

The evaluator examines the KMD to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator shall check to ensure the KMD lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) and description in the KMD.

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the KMD to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

The evaluator shall check that the KMD identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

If the selection 'destruction of all KEKs protecting target key, where none of the KEKs protecting the target key are derived' is included the evaluator shall examine the TOE's keychain in the KMD and identify each instance when a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM_EXT.4.1. The evaluator shall verify that all of the keys capable of decrypting the target key are not able to be derived to reestablish the keychain after their destruction.

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside and when the keys and key material are no longer needed.

The evaluator shall verify the KMD includes a key lifecycle, that includes a description where key material reside, how the key material is used, how it is determined that keys and key material are no longer needed, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM_EXT.4.1 for the destruction.

Section 6.1 of the ST explains the TOE relies on the platform for destroying cryptographic keys when they are no longer in use. The platform cryptographic modules provide functionality to automatically overwrite (with zeros) of keys when they are released. For keys stored in Flash the TOE calls the platform to perform a block erase to the flash controller. The TOE only writes encrypted keys to flash storage.

KMD - Section 2.2 of the KMD contains a proprietary table that provides the following information about each key: Name, type, origin, size, storage location, when cleared and method of clearing. All information is consistent with the requirements.

Component Guidance Assurance Activities: There are a variety of concerns that may prevent or delay key destruction in some cases.

The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information.

The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.

Some examples of what is expected to be in the documentation are provided here.

When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wearleveling and garbage collection. This may create additional copies of the key that are logically

inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.

The drive should be healthy and contains minimal corrupted data and should be end of lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

Section 4.1 of the Admin Guide explains how to wipe the device. The wipe capabilities are not part of the Knox File Encryption software but are built into the underlying platform.

Component Testing Assurance Activities: These tests are only for key destruction provided by the application, test 2 does not apply to any keys using the selection 'new value of a key':

Test 1: [Conditional; applies when the application does not perform the zeroization (ie. garbage collecting) for each key held in volatile memory for FCS_CKM_EXT.4.1 (assuming the selection 'destruction of the reference followed by a request for garbage collection')] Applied to each key held in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the key destruction method was removal of power, then this test is unnecessary.

The evaluator shall:

1. Record the value of the key in the TOE subject to clearing.

2. Cause the cause the TOE or the underlying platform to dump to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Cause the TOE to stop the execution but not exit.

5. Cause the TOE to dump the entire memory of the TOE into a binary file.

6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

Steps #1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Test 2: [Conditional; applies when instructing the underlying platform to destroy the key] If new value of a key is selected this test does not apply.

Applied to each key held in non-volatile memory and subject to destruction by the TOE.

The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.

1. Identify the purpose of the key and what access should fail when it is deleted. (e.g. the file encryption key being deleted would cause data decryption to fail.)

2. Cause the TOE to clear the key.

3. Have the TOE attempt the functionality that the cleared key would be necessary for.

4. The test succeeds if Step #3 fails.

Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key, as the TOE has no visibility into the inner workings and completely relies on the underlying platform.

Test 3: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.

Test 4: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:

1. Record the logical storage location of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

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

Test 1– For volatile memory the evaluator searched volatile memory for MKDD and FEKs. None were found in the memory. The evaluator repeated this test for 2 variations - Knox Container and Whole Device encryption.

Test 2 -. The evaluator created several files and demonstrated they were accessible. The evaluator then removed the workspace and searched volatile memory for MKDD and FEKs associated with the files. None were found in the memory. The evaluator repeated this test for 2 variations - Knox Container and Whole Device encryption.

Test 3 &4 – Not applicable – Per the module supporting document "Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key, as the TOE has no visibility into the inner workings and completely relies on the underlying platform."

2.1.7 Cryptographic Password/Passphrase Conditioning (FE10:FCS_CKM_EXT.6)

2.1.7.1 FE10:FCS_CKM_EXT.6.1

TSS Assurance Activities: There are two aspects of this component that require evaluation: passwords/passphrases of the length specified in the requirement (at least 64 characters or a length defined by the platform) are supported, and that the characters that are input are subject to the selected conditioning function. These activities are separately addressed in the text below.

Support for minimum length: The evaluators shall check to ensure that the TSS describes the allowable ranges for password/passphrase lengths, and that at least 64 characters or a length defined by the platform may be specified by the user.

Support for character set: The evaluator shall check to ensure that the TSS describes the allowable character set and that it contains the characters listed in the SFR.

Support for PBKDF: The evaluator shall examine the TSS to ensure that the formation of all KEKs or FEKs (as decided in the FCS_CKM_EXT.3 selection) is described and that the key sizes match that described by the ST author.

The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_CKM_EXT.4.

For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1(4)). If any

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

No explicit testing of the formation of the submask from the input password is required.

Section 6.1 of the ST states depending on the FEB configuration, the TOE either utilizes the platform authentication services of the Knox work profile to access the user password or provides its own authentication dialog. In either case, the passwords follow common requirements. Passwords can be between 4-256 characters in length. The administrator may choose to require any password length within this range. Passwords may be composed of basic Latin characters (upper and lower case, numbers, and the special characters noted in the selection (see section 5.1.1.7)).

The entered password is conditioned with 100,000 rounds of PBKDF2 (using HMAC-SHA-256) to generate a 256-bit KEK. This KEK is used to encrypt the 256-bit MKDD using AES-GCM.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.2 FE10:FCS_CKM_EXT.6.2

TSS Assurance Activities: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1 (from the [AppPP]).

Section 6.1 in the ST in the FCS_CKM.1/PBKDF section states all salts are generated using RBGs as specified in FCS_RBG_EXT.1.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.3 FE10:FCS_CKM_EXT.6.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.4 FE10:FCS_CKM_EXT.6.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.5 FE10:FCS_CKM_EXT.6.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: Support for minimum length: The evaluators shall check the Operational Guidance to determine that there are instructions on how to generate large passwords/passphrases, and instructions on how to configure the password/passphrase length to provide entropy commensurate with the keys that the authorization factor is protecting.

Section 2.4.1.1 of the Admin Guide provides a reference to NIST SP 800-63B, section 5.1.1, Memorized Secrets for selecting strong passwords.

Component Testing Assurance Activities: Support for Password/Passphrase characteristics: In addition to the analysis above, the evaluator shall also perform the following tests on a TOE configured according to the Operational Guidance:

Test 1: Ensure that the TOE supports password/passphrase lengths as defined in the SFR assignments.

Test 2: Ensure that the TOE does not accept more than the maximum number of characters specified in FCS_CKM_EXT.6.1.

Test 3: Ensure that the TOE does not accept less than the minimum number of characters specified in FCS_CKM_EXT.6.4. If the minimum length is settable by the administrator, the evaluator determines the minimum length or lengths to test.

Test 4: Ensure that the TOE supports passwords consisting of all characters listed in FCS_CKM_EXT.6.2.

Conditioning: No explicit testing of the formation of the authorization factor from the input password/passphrase is required.

The evaluator repeated these tests for 2 variations - Knox Container and Whole Device encryption:

Test 1 - The evaluator configured a password of 256 characters and demonstrated that it could be accepted. The evaluator also demonstrated that a 4-character password was acceptable (minimum length).

Test 2 – As part of test 1, the evaluator attempted to set a 257-character password. It appeared to accept the password but only the first 16 characters are needed to pass the authentication check (because it was truncated as expected).

Test 3 – As part of test 1, the evaluator attempted to set a 3-character password. The password attempt was rejected.

Test 4 – As part of test 1, the evaluator successfully created passwords with all the claimed characters.

2.1.8 Cryptographic operation (Key Wrapping) (FE10:FCS_COP.1(5))

2.1.8.1 FE10:FCS_COP.1.1(5)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Conditional: If use platform provided functionality was selected, then the evaluator shall examine the TSS to verify that it describes how the FEK encryption/decryption is invoked.

Conditional: If implement functionality was selected, The evaluator shall check that the TSS includes a description of encryption function(s) used for key wrapping. The evaluator should check that this description of the selected encryption function includes the key sizes and modes of operations as specified in the selection above. The evaluator shall check that the TSS describes the means by which the TOE satisfies constraints on algorithm parameters included in the selections made for 'cryptographic algorithm' and 'list of standards'.

The evaluator shall verify the TSS includes a description of the key wrap function(s) and shall verify the key wrap uses an approved key wrap algorithm according to the appropriate specification.

KMD

The evaluator shall review the KMD to ensure that all keys are wrapped using the approved method and a description of when the key wrapping occurs.

Section 6.1 of the ST states the TOE wraps all stored encryption keys using 256-bit AES-GCM according to NIST SP 800-38D as provided by the platform encryption modules. The FCS_KYC_EXT.1 discussion explains how all keys are wrapped with AES-GCM and explains when each is encrypted.

KMD – The KMD provides details of key sizes and modes of operations and explains how each key is wrapped.

Component Guidance Assurance Activities: If multiple encryption modes are supported, the evaluator examines the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

Not applicable. Only one encryption mode/key size is supported.

Component Testing Assurance Activities: The evaluation activity tests specified for AES in GCM mode in the underlying [AppPP] shall be performed in the case that 'GCM' is selected in the requirement.

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

The evaluator will test the authenticated encryption functionality of AES-KW for EACH combination of the following input

parameter lengths:

128 and 256 bit key encryption keys (KEKs)

Three plaintext lengths. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall

be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or

equal to 64 semi-blocks (4096 bits).

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To

determine correctness, the evaluator will use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator will test the authenticated-decryption functionality of AES-KW using the same test as for authenticatedencryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW

authenticated-decryption.

The evaluator will test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW

authenticated-encryption with the following change in the three plaintext lengths:

One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).

One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

The evaluator will test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP

authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AESKWP authenticated-decryption.

AES-CCM Tests

It is not recommended that evaluators use values obtained from static sources such as

http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the

AES-CCM implementation.

The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input

parameter and tag lengths: Keys: All supported and selected key sizes (e.g., 128, 256 bits). Associated Data: Two or three

values for associated data length: The minimum (= 0 bytes) and maximum (= 32 bytes) supported associated data lengths, and

2^16 (65536) bytes, if supported. Payload: Two values for payload length: The minimum (= 0 bytes) and maximum (= 32 bytes)

supported payload lengths. Nonces: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13) in bytes. Tag: All supported tag

lengths (4, 6, 8, 10, 12, 14, 16) in bytes.

The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare

the ciphertext with the result of encryption of the same inputs with a known good implementation.

Variable Associated Data Test

For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the

evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the

resulting ciphertext.

Variable Payload Text

For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the

evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the

resulting ciphertext.

Variable Nonce Test

For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the

evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the

resulting ciphertext.

Variable Tag Test

For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the

evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the

resulting ciphertext.

Decryption-Verification Process Test

To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length,

payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and

obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

Test – See section 1.3 for an identification of CAVP certificates.

2.1.9 Cryptographic Operation - Keyed-Hash Message Authentication - per TD0626 (ASPP14:FCS_COP.1/KeyedHash)

2.1.9.1 ASPP14:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

Test – See section 1.3 for an identification of CAVP certificates.

2.1.10 Cryptographic Operation - Encryption/Decryption (ASPP14:FCS_COP.1/SKC)

2.1.10.1 ASPP14:FCS_COP.1.1/SKC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required modes and key sizes is present.

Section 6.1 of the ST states the key sizes and algorithms used cannot be changed. As such, no Guidance is required.

Component Testing Assurance Activities: The evaluator shall perform all of the following tests for each algorithm implemented by the TSF and used to satisfy the requirements of this PP:

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all- zeros key. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

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

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3- tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

CT[1] = AES-CBC-Encrypt(Key, IV, PT)

PT = IV

else:

CT[i] = AES-CBC-Encrypt(Key, PT)

PT = CT[i-1]

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

AES-GCM Monte Carlo Tests

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

128 bit and 256 bit keys

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

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

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

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

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

AES-XTS Tests

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

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

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

Using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt. The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

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

AES-CCM Tests

It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

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

Keys: All supported and selected key sizes (e.g., 128, 256 bits).

Associated Data: Two or three values for associated data length: The minimum (. 0 bytes) and maximum (. 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.

Payload: Two values for payload length: The minimum (. 0 bytes) and maximum (. 32 bytes) supported payload lengths.

Nonces: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13) in bytes.

Tag: All supported tag lengths (4, 6, 8, 10, 12, 14, 16) in bytes.

The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

Variable Associated Data Test

For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Payload Test

For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Nonce Test

For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Tag Test

For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Decryption-Verification Process Test

To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, IV, and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

To test the encrypt functionality, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all zeros key, and the other five shall be encrypted with a 256-bit all zeros key. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input.

To test the encrypt functionality, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. Five of the key values shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.

To test the encrypt functionality, the evaluator shall supply the two sets of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values an an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second shall have 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit pairs. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros for i in [1, N]. The ciphertext value in each pair shall be the value that results in an all zeros plaintext when decrypted with its corresponding key.

To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 128-bit key value of all zeros and using a 256 bit key value of all zeros, respectively, and an IV of all zeros. Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128]. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.

Test 2: Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

Test 3: Monte-Carlo Test

For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine.

The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode

Input: PT, Key

for i = 1 to 1000:

CT[i] = AES-ECB-Encrypt(Key, PT)

PT = CT[i]

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

Test – See section 1.3 for an identification of CAVP certificates.

2.1.11 Initialization Vector Generation (FE10:FCS_IV_EXT.1)

2.1.11.1 FE10:FCS_IV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes how nonces are created uniquely and how IVs and tweaks are handled (based on the AES mode). The evaluator shall confirm that the nonces are unique and the IVs and tweaks meet the stated requirements.

Section 6.1 of the ST states that the TOE generates unique IVs for AES-CBC used to encrypt data storage (files). The IVs are generated by using the Samsung Kernel Crypto HMAC_DRBG API when the AES FEK for a file is created.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.12 Cryptographic Key Derivation Function (FE10:FCS_KDF_EXT.1)

2.1.12.1 FE10:FCS_KDF_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 and SP 800-132.

Section 6.1 of the ST explains the TOE conditions the user password using a PBKDF2 (using HMAC-SHA-256) function that meets NIST SP 800-132 with 100,000 rounds to generate a 256-bit KEK to protect the Master Key (MKDD).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.13 Key Chaining and Key Storage (FE10:FCS_KYC_EXT.1)

2.1.13.1 FE10:FCS_KYC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS contains a high level description of all keychains and authorization methods selected in FIA_AUT_EXT.1 that are used to protect the KEK or FEK.

KMD

The evaluator shall examine the KMD to ensure it describes each key chain in detail, and these descriptions correspond with the selections of the requirement. The description of each key chain shall be reviewed to ensure the options for maintaining the key chain are documented.

The evaluator shall verify the KMD to ensure that it describes how each key chain process functions, such that it does not expose any material that might compromise any key in the chain. A high-level description should include a diagram illustrating the keychain(s) implemented and detail where all keys and keying material is stored or how the keys or key material are derived. The evaluator shall examine the primary key chain to ensure that at no point

the chain could be broken without a cryptographic exhaust or knowledge of the KEK or FEK and the effective strength of the FEK is maintained throughout the Key Chain as specified in the requirement.

Section 6.1 explains that the PMKEK and MKDD are 256-bit AES keys while all FEKs are effectively 128-bit AES keys. The MKDD is protected by the PMKEK, which is derived from the password using PBKDF2. The encrypted MKDD is stored in TrustZone as part of the DualDAR Client Trusted App data. The MKDD is used to encrypt the FEK of each file in the FEB (the FEK is stored in the associated file metadata). All keys are encrypted with AES-GCM.

KMD –

The vendor provided a proprietary set of diagrams identifying each key and each operation on each key. The evaluator analyzed those diagrams to ensure the keychain was properly protected and all claimed strengths (256-bits) are maintained throughout the lifetime of the keychain.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.14 Random Bit Generation Services (ASPP14:FCS_RBG_EXT.1)

2.1.14.1 ASPP14:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'use no DRBG functionality' is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

If 'implement DRBG functionality' is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.

If 'invoke platform-provided DRBG functionality' is selected, the evaluator performs the following activities.

The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used correctly for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

Section 6.1 of the ST states the TOE utilizes a number of different RBGs included in the platform:

1. An AES-256 CTR_DRBG provided by SCrypto in the TEE OS
2. A SHA-256 HMAC_DRBG provided by Kernel Crypto in the Android Linux kernel.

Each of these entropy sources will generate a 256-bit string. The APIs used by the TOE are available to applications that are part of the system services (not general Android applications).

The Security Target explains the APIs utilized and how those related to RBGs sources in the underlying platform – those APIs are not among those identified in this testing assurance activity given the alternate nature of the implementation. The KMD contains the internal details.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If 'invoke platform-provided DRBG functionality' is selected, the following tests shall be performed:

The evaluator shall decompile the application binary using a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

The following are the per-platform list of acceptable APIs:

Platforms: Android....

The evaluator shall verify that the application uses at least one of javax.crypto.KeyGenerator class or the java.security.SecureRandom class or /dev/random or /dev/urandom.

Platforms: Microsoft Windows....

The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

Platforms: Apple iOS....

The evaluator shall verify that the application invokes either SecRandomCopyBytes, CCRandomGenerateBytes or CCRandomCopyBytes, or uses /dev/random directly to acquire random.

Platforms: Linux....

The evaluator shall verify that the application collects random from /dev/random or /dev/urandom.

Platforms: Oracle Solaris....

The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

Platforms: Apple macOS....

The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

Test - The TOE includes a kernel driver and TrustZone applications. It is not plausible to decompile those components and while source code for the applicable components has been provided, that has not been accepted instead of decompilation. The Security Target explains the APIs utilized and how those related to RNG sources in the underlying platform – those APIs are not among those identified in this testing assurance activity given the alternate nature of the implementation.

The evaluator obtained the actual binaries and used the strings command to identify the RBG APIs.

2.1.15 Storage of Credentials (ASPP14:FCS_STO_EXT.1(1))

2.1.15.1 ASPP14:FCS_STO_EXT.1.1(1)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

Section 6.1 of the ST provides the necessary information for each persistent key. The following is explained: The TOE uses a TrustZone application to protect the MKDD that is unlocked when the user successfully authenticates. The AES-GCM encrypted key is stored within the TrustZone boundary.

The TOE protects individual FEKs (and IVs) by encrypting them using AES-GCM with the MKDD and placing it in the file metadata.

The PMKEK is conditioned from the password using a PBKDF2 function following NIST SP 800-132.

Further details about each key are in the KMD.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/SKC or conditioned according to FCS_CKM.1.1/AK and FCS_CKM.1/PBKDF. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

Platforms: Android....

The evaluator shall verify that the application uses the Android KeyStore or the Android KeyChain to store certificates.

Platforms: Microsoft Windows....

The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

Platforms: Apple iOS....

The evaluator shall verify that all credentials are stored within a Keychain.

Platforms: Linux....

The evaluator shall verify that all keys are stored using Linux keyrings.

Platforms: Oracle Solaris....

The evaluator shall verify that all keys are stored using Solaris Key Management Framework (KMF).

Platforms: Apple macOS....

The evaluator shall verify that all credentials are stored within Keychain

Android-specific test: not applicable - The TOE does not store certificates or invoke platform functionality for this function.

The Security Target explains that the MKDD is encrypted and stored in the TrustZone boundary and that the individual FEKs are encrypted and stored in metadata (appearing as random data). TrustZone keys are not accessible even in encrypted form. In order to ensure the FEK is actually encrypted (and MKDD is not accessible), the evaluators searched flash for the known plaintext value of the MKDD and FEK on each test device in each applicable configuration. The searches found no keys.

2.1.16 Storage of Credentials (ASPP14:FCS_STO_EXT.1(2))

2.1.16.1 ASPP14:FCS_STO_EXT.1.1(2)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

See ASPP14:FCS_STO_EXT.1(1) for a description.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/SKC or conditioned according to FCS_CKM.1.1/AK and FCS_CKM.1/PBKDF. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

Platforms: Android....

The evaluator shall verify that the application uses the Android KeyStore or the Android KeyChain to store certificates.

Platforms: Microsoft Windows....

The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

Platforms: Apple iOS....

The evaluator shall verify that all credentials are stored within a Keychain.

Platforms: Linux....

The evaluator shall verify that all keys are stored using Linux keyrings.

Platforms: Oracle Solaris....

The evaluator shall verify that all keys are stored using Solaris Key Management Framework (KMF).

Platforms: Apple macOS....

The evaluator shall verify that all credentials are stored within Keychain

See ASPP14:FCS_STO_EXT.1(1).

2.1.17 Validation (FE10:FCS_VAL_EXT.1)

2.1.17.1 FE10:FCS_VAL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.17.2 FE10:FCS_VAL_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Conditional: If 'validating' is selected in FCS_VAL_EXT.1.1, the evaluator shall examine the TSS to determine which authorization factors

support validation.

The evaluator shall examine the TSS to ensure that it contains a high-level description of how how the submasks are validated. If multiple submasks are used within the TOE, the evaluator shall verify that the TSS describes how each is validated (e.g., each submask validated before combining, once combined validation takes place).

Conditional: If 'receiving assertion' is selected in FCS_VAL_EXT.1.1, the evaluator shall examine the TSS to verify that it describes the enviroments that can be leveraged with the TOE and how each claims to perform validation. The evaluator shall ensure that none of the stated platform validation mechanisms weaken the key chain of the product.

Section 6.1 of the ST explains that when used with a Knox work profile, the TOE relies on the platform to validate the user credentials via the authentication interface for the Knox work profile. When used to encrypt the entire device, the TOE provides its own authentication interface immediately after the device lock screen. A successful authentication in either case will provide the TOE with access to the entered password to begin the process for accessing the MKDD and hence the protected files.

The MKDD can only be decrypted by the PMKEK, which must be conditioned from the user's authentication credentials. This ensures that the files cannot be accessed without a successful user authentication.

The TOE is a component of the Samsung Knox software package. The TOE is available and can be used only on devices where it is provided from Samsung (i.e. this is not available as a separate application that could be downloaded and installed separately). The operational environments for the TOE are listed in the ST as part of Table 1 - Evaluated Devices in the columns: Android Version, TEE OS and Knox Version.

Component Guidance Assurance Activities: If the validation functionality is configurable, the evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established.

Not applicable. The validation function is not configurable.

Component Testing Assurance Activities: None Defined

2.2 User data protection (FDP)

2.2.1 Encryption Of Sensitive Application Data (ASPP14:FDP_DAR_EXT.1)

2.2.1.1 ASPP14:FDP_DAR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application. The evaluator shall then ensure that the following activities cover all of the

sensitive data identified in the TSS. If not store any sensitive data is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory. The evaluator shall also ensure that this is consistent with the filesystem test below.

Section 6.2 of the ST explains the sensitive data processed by the TOE. The TOE leverages the platform TrustZone for protection of the MKDD. The TOE includes a Trusted App which leverages an encryption module within TrustZone to encrypt (and decrypt) the MKDD.

To encrypt files, the TOE implements a service which will intercept file requests to encrypt (and decrypt) the files using the included cryptographic modules. All files written inside the Knox work profile boundary, including user data, temporary files or any other type of file created by an application, are considered sensitive data and will be encrypted. When used to encrypt the entire device, all files except a small number of system apps (needed for the device to function properly and specified by Samsung) are encrypted. All other files are automatically considered sensitive data and will be encrypted.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1. The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted.

If 'leverage platform-provided functionality' is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis.

Platforms: Android....

The evaluator shall inspect the TSS and verify that it describes how files containing sensitive data are stored with the MODE_PRIVATE flag set.

Platforms: Microsoft Windows....

The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.

Platforms: Apple iOS....

The evaluator shall inspect the TSS and ensure that it describes how the application uses the Complete Protection, Protected Unless Open, or Protected Until First User Authentication Data Protection Class for each data file stored locally.

Platforms: Linux....

The Linux platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

Platforms: Oracle Solaris....

The Solaris platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

Platforms: Apple macOS....

The macOS platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

Test - Note that the TOE does not have any default locations to save protected data; rather it protects any data files automatically that are stored within the Knox workspace storage areas. The evaluator used a developer tool to create a file in a known location. The evaluator found that the file was created by the test script exactly where the script indicated and nowhere else and furthermore found that the file was seemingly encrypted both by the test device full disk encryption as well as a second time by the DualDAR function as expected.

2.2.2 Access to Platform Resources (ASPP14:FDP_DEC_EXT.1)

2.2.2.1 ASPP14:FDP_DEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.

Not applicable – the TOE does not utilize any hardware resources.

Testing Assurance Activities: Platforms: Android....

The evaluator shall verify that each uses-permission entry in the AndroidManifest.xml file for access to a hardware resource is reflected in the selection.

Platforms: Microsoft Windows....

For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as ID_CAP_ISV_CAMERA, ID_CAP_LOCATION,

ID_CAP_NETWORKING, ID_CAP_MICROPHONE, ID_CAP_PROXIMITY and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of the required hardware resources.

Platforms: Apple iOS....

The evaluator shall verify that either the application or the documentation provides a list of the hardware resources it accesses.

Platforms: Linux....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Platforms: Oracle Solaris....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Platforms: Apple macOS....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Not applicable – the TOE does not utilize any hardware resources.

2.2.2.2 ASPP14:FDP_DEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information repository which it accesses, identify the justification as to why access is required.

Not applicable – the TOE does not utilize any sensitive information repositories.

Testing Assurance Activities: Platforms: Android....

The evaluator shall verify that each uses-permission entry in the AndroidManifest.xml file for access to a sensitive information repository is reflected in the selection.

Platforms: Microsoft Windows....

For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS,ID_CAP_APPOINTMENTS,ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.

Platforms: Apple iOS....

The evaluator shall verify that either the application software or its documentation provides a list of the sensitive information repositories it accesses.

Platforms: Linux....

The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Platforms: Oracle Solaris....

The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Platforms: Apple macOS....

The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Not applicable – the TOE does not utilize any sensitive information repositories.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.3 Network Communications (ASPP14:FDP_NET_EXT.1)

2.2.3.1 ASPP14:FDP_NET_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

Test 1: The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.

Test 2: The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).

Platforms: Android....

If 'no network communication' is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a <uses-permission> or <uses-permission-sdk-23> tag containing android:name='android.permission.INTERNET'. In this case, it is not necessary to perform the above Tests 1 and 2, as the platform will not allow the application to perform any network communication.

Test - The evaluator examined the AndroidManifest.xml file and found no instances of “INTERNET”. The remaining components are kernel drivers which do not have access to the network stack. This was tested in ASPP14:FTP_DIT_EXT.1.

2.2.4 Protection of Data in Power Managed States (FE10:FDP_PM_EXT.1)

2.2.4.1 FE10:FDP_PM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.4.2 FE10:FDP_PM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.4.3 FE10:FDP_PM_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes the state(s) that are supported by this capability. For each state, the evaluator ensures that the TSS contains a description of how the state is entered, and the actions of the TSF on entering the state, specifically addressing how multiple open resources (of each type) are protected, and how keying material associated with these resources is protected (if different from that described elsewhere). The TSF shall also describe how the state is exited, and how the requirements are met during this transition to an operational state.

The evaluator shall verify the TSS provides a description of what keys and key material are destroyed when entering any protected state.

KMD

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside. The evaluator shall verify the KMD includes a key lifecycle that includes a description where key material reside, how the key material is used, and how the material is destroyed once a claimed power state is entered and that the documentation in the KMD follows FCS_CKM_EXT.4.1 for the destruction.

Section 6.2 of the ST states the TOE supports three discrete states on the device:

1. Off: the device is powered off (i.e. no power to the device)
2. Data Lock: the device is powered on, not authenticated to the TOE
3. Data Unlock: the device is powered on, authenticated to the TOE

The TOE spends most of its time between states two (Data Lock) and three (Data Unlock). It is not possible to move directly from Off to Data Unlock; the TOE always starts in the Data Lock state on startup.

During the Off state, the MKDD and any unlocked FEKS 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 transition to the Data Unlock state. How the TOE came to be in the Data Lock state (i.e. either directly from Off or from Data Unlock), does not change the functionality of the state; in either case successful authentication is required to move to the Data Unlock state. The authentication will allow the MKDD to be decrypted, which in turn will allow the FEKs to be decrypted.

When the TOE transitions to the Data Lock state from the Data Unlock state, the MKDD and any FEKs are cleared from volatile storage and any open files are closed. The transition to the Data Lock state can be initiated by the device being locked or by a timeout period (tied to inactivity in the work profile or by an administrator configuration).

Inactivity settings are controlled by the work profile configuration. These settings are not part of the TOE configuration but are used to determine when the Data Lock transition should occur. The transition itself controlled by the Data Lock setting which specifies the amount of inactive time that can pass before an automatic transition to the Data Lock state. The Data Lock timeout starts after the work profile inactivity period has been reached.

When the FEB configuration is for a Knox work profile, inactivity settings are controlled by the work profile configuration. Similarly, when the FEB configuration is for the whole device, the inactivity settings are controlled by the device configuration. These settings are not part of the TOE configuration but are used to determine when the Data Lock transition should occur. The transition itself controlled by the Data Lock setting which specifies the amount of inactive time that can pass before an automatic transition to the Data Lock state. The Data Lock timeout starts after the device or work profile inactivity period has been reached.

KMD – Section 2.2 of the KMD provides detailed key information. These diagrams support the description in Section 6.2 of the ST. The description of when keys are cleared also matches that in FCS_CKM_EXT.4.

Component Guidance Assurance Activities: The evaluator shall check the Operational Guidance to determine that it describes the states that are supported by the TOE, and provides information related to the correct configuration of these modes and the TOE.

The evaluator shall validate that guidance documentation contains clear warnings and information on conditions in which the TOE may end up in a non-protected state. In that case it must contain mitigation instructions on what to do in such scenarios.

Section 2.1 of the Admin Guide provides an overview of the TOE. The description does not explicitly list the states but does discuss that Knox File Encryption is designed as a framework which can be used for the Knox work profile. Through this service, all files (per the configuration) that are read or written when Knox File Encryption is enabled will be filtered and encrypted/decrypted automatically. The service does not require the user or any apps to be aware of the service, only that Knox File Encryption to be enabled for the work profile or device. Section 2.4.1 discusses the user password necessary for the encryption/decryption.

The Admin Guide in Section 2.2 clearly states that the deployment of Knox File Encryption is tied to the deployment of a device. When creating a Knox work profile, the administrator must select the DualDAR option to enable Knox File Encryption. When configuring the whole device, it must be enabled during the initial device configuration. This is the only step necessary to activate Knox File Encryption on a supported Samsung.

Component Testing Assurance Activities: The following tests must be performed by the evaluator for each supported State, type of resource, platform, and authorization factor:

Test 1: Following the Operational guidance, configure the Operational Environment and the TOE so that the lower power state of the platform is enabled and protected by the TOE. Open several resources (documented in the test report) that are protected. Invoke the lower power state. On resumption of normal power attempt to access a previously-opened protected resource, observe that an incorrect entry of the authorization factor(s) does not result in access to the system, and that correct entry of the authorization factor(s) does result in access to the resources.

Test - The Security Target defines only one low-power state when the data locked state. The evaluator demonstrated that a number of encrypted files can be decrypted while in a normal operational state. The evaluator then locked the test device and waited for the configured data lock timeout limit (1 minute) and then tried to access those files again. The evaluator then unlocked the test device and then unlocked the DualDAR work profile in each case and then tried to access those files a final time. The evaluator observed that the files could be decrypted only while the DualDAR work profile was unlocked, otherwise the decryption keys were not available. The evaluator repeated this test for 2 variations - Knox Container and Whole Device encryption.

2.2.5 Protection of Selected User Data (FE10:FDP_PRT_EXT.1)

2.2.5.1 FE10:FDP_PRT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.5.2 FE10:FDP_PRT_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it lists each type of resource that can be encrypted (e.g., file, directory) and what 'encrypted' means in terms of the resource (e.g., 'encrypting a directory' means that all of the files contained in the directory are encrypted, but the data in the directory itself (which are filenames and pointers to the files) are not encrypted).

The evaluator shall also confirm that the TSS describes how each type of resource listed is encrypted and decrypted by the TOE. The evaluator shall ensure that this description includes the case where an existing file or set of files is encrypted for the first time; a new file or set of files is created and encrypted; an existing file or set of files is re-encrypted (that is, it had been initially encrypted; it was decrypted (by the TOE) for use by the user, and is then subsequently re-encrypted); and corresponding decryption scenarios. If other scenarios exist due to product implementation/features, the evaluator shall ensure that those scenarios are covered in the TSS as well.

The evaluator shall examine the TSS to ensure there is a high-level description of how the FEK is protected.

The evaluator shall examine the TSS to ensure there is a description of how the FEK is protected.

The evaluator shall examine the TSS to ensure that it describes all temporary files/resources created or memory used during the decryption/encryption process and when those files/resources or memory is no longer needed.

The TSS shall describe how the TSF or TOE platform deletes the non-volatile memory (for example, files) and volatile memory locations after the TSF is done with its decryption/encryption operation.

Section 6.2 of the ST explains the TOE automatically encrypts all files within the FEB, regardless of the point of creation (i.e. a direct user request to create a file vs. an application creating the file on behalf of the user). No temporary files are used during the encryption/decryption process as all file content access is handled through in-place buffers. The TOE only encrypts the file contents; file metadata is not encrypted (the encrypted version of the FEK is considered to be metadata).

The TOE is invoked when the FEB is unlocked by the user entering their credentials. Until successful authentication, all files are encrypted and not accessible. When the FEB enters the Data Lock state, the MKDD and all FEKs are cleared from memory, closing all applications and leaving all data in non-volatile memory encrypted (a Flush Operation is called when a work profile is locked, the device is automatically rebooted when the device is encrypted).

The TOE intercepts all read/write calls from within the FEB and automatically encrypts/decrypts the files utilizing the platform encryption libraries.

See FE10:FCS_KYC_EXT.1 for a discussion of how the FEK is protected

Component Guidance Assurance Activities: If the TOE creates temporary objects and these objects can be protected through administrative measures (e.g., the TOE creates temporary files in a designated directory that can be protected through configuration of its access control permissions), then the evaluator shall check the Operational Guidance to ensure that these measures are described.

If there are special measures necessary to configure the method by which the file or set of files are encrypted (e.g., choice of algorithm used, key size, etc.), then those instructions shall be included in the Operational Guidance and verified by the evaluator. In these cases, the evaluator checks to ensure that all non-TOE products used to satisfy the requirements of the ST that are described in the Operational Guidance are consistent with those listed in the ST, and those tested by the evaluation activities of this PP-Module.

There are no additional guidance evaluation activities for FDP_PRT_EXT.1.2.

Not applicable. The TOE does not create any temporary objects and the encryption happens automatically.

Component Testing Assurance Activities: The evaluator shall also perform the following tests. All instructions for configuring the TOE and each of the environments must be included in the Operational Guidance and used to establish the test configuration.

For each resource and decryption/encryption scenario listed in the TSS, the evaluator shall ensure that the TSF is able to successfully encrypt and decrypt the resource using the following methodology:

Monitor the temporary resources being created (if any) and deleted by the TSF-the tools used to perform the monitoring (e.g., procmon for a Windows system) shall be identified in the test report. The evaluator shall ensure that these resources are consistent with those identified in the TSS, and that they are protected as specified in the Operational Guidance and are deleted when the decryption/encryption operation is completed.

Test 1: This test only applies for application provided functionality.

1. Using a file editor, create and save a text file that is encrypted per the evaluation configured encryption policy. The contents of the file will be limited to a known text pattern to ensure that the text pattern will be present in all encryption/decryption operations performed by the TOE.

2. Exit the file editor so that the file (including its known text pattern) has 'completed the decryption/encryption operation' and process memory containing the known text pattern is released.

3. The evaluator will take a dump of volatile memory and search the retrieved dump for the known pattern. The test fails if the known plaintext pattern is found in the memory dump.

4. The evaluator will search the underlying non-volatile storage for the known pattern. The test fails if the known plaintext pattern is found in the search.

Test – The evaluator downloaded a text editor onto the phone and created a known string. The evaluator searched volatile memory for the string and demonstrated it could not be found. The evaluator then used a developer tool to demonstrate the known string was not available. The evaluator repeated this test for 2 variations - Knox Container and Whole Device encryption.

2.2.6 Destruction of Plaintext Data (FE10:FDP_PRT_EXT.2)

2.2.6.1 FE10:FDP_PRT_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes all temporary file (or set of files) that are created in the filesystem of the host during the decryption/encryption process, and that the TSS describes how these files are deleted after the TSF is done with its decryption/encryption operation. Note that if other objects/resources are created on the host that are 1) persistent and 2) visible to other processes (users) on that host that are not filesystem objects, those objects shall be identified and described in the TSS as well.

See FE10:FDP_PRT_EXT.1 for a description of the encryption process. No temporary objects are created as part of this process.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 1: If the TSS creates temporary files/resources during file decryption/encryption, the evaluator shall perform the following tests to verify that the temporary files/resources are destroyed. If the product supported shared files per FIA_FCT_EXT.2, this test must be repeated with a shared file. The evaluator shall use a tool (e.g., procmon for a Windows system) that is capable of monitoring the creation and deletion of files during the decryption/encryption process is performed. A tool that can search the contents of the hard drive (e.g., winhex) will also be needed. The tools used to perform the monitoring shall be identified in the test report.

(Creating an encrypted document)

Open an editing application.

Create a special string inside the document. The string could be 5-10 words. It is recommended to remove the spaces. This will create a one page document.

Start the file monitoring tool.

Save and close the file.

Encrypt the file using the TOE (if the TOE does not encrypt automatically for the user).

Analysis Steps

If needed, exit/close the TOE.

Stop the file monitoring tool. View the results. Identify any temporary files that were created during the encryption process. Examine to see if the temporary files were destroyed when the TOE closed.

If temporary files remain, these temporary files should be examined to ensure that no plaintext data remains. If plaintext data is found in these files, the test fails.

Search the contents of the hard drive (using the second tool) for the plaintext string used above. (The search should be performed using both ASCII and Unicode formats.)

If the string is found, this means that plaintext from the test fails.

(Creating, encrypting a blank document and then adding text):

Encrypt a blank document using the tool.

Create a special string inside the document. The string could be 5-10 words. It is recommended to remove the spaces. This will create a one page document.

Start the file monitoring tool.

Save and close the file.

Perform the 'Analysis Steps' listed above.

Test - Not applicable – the TOE does not create temporary files or support shared files.

2.2.7 PROTECTION OF THIRD-PARTY DATA (FE10:FDP_PRT_EXT.3)

2.2.7.1 FE10:FDP_PRT_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes how the TOE detects and encrypts temporary files (or set of files) that are created in the filesystem of the host by third party products.

Section 6.2 of the ST states the TOE automatically intercepts all read/write requests for files contained within the FEB. This ensures that any temporary files created by applications within the FEB are automatically encrypted without the application needing to be made aware.

Component Guidance Assurance Activities: [conditional] If any configuration is required for this process the evaluator shall verify it is described in the guidance documentation.

Not applicable – all encryption performed by the TOE is done automatically.

Component Testing Assurance Activities: The evaluator shall utilize any third party application that would be protected under this protection to generate files, then verify those files are being encrypted.

Test – The evaluator installed a Chrome browser and used it to create data within the protected work profile/device. The evaluator then used a developer-provided tool to demonstrate that Chrome's data was encrypted. The evaluator repeated this test for 2 variations - Knox Container and Whole Device encryption.

2.3 IDENTIFICATION AND AUTHENTICATION (FIA)

2.3.1 SUBJECT AUTHORIZATION (FE10:FIA_AUT_EXT.1)

2.3.1.1 FE10:FIA_AUT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes how user authentication is performed. The evaluator shall verify that the authorization methods listed in the TSS are specified and included in the requirements in the ST.

Requirement met by the TOE:

The evaluator shall first examine the TSS to ensure that the authorization factors specified in the ST are described. For password-based factors the examination of the TSS section is performed as part of FCS_CKM_EXT.6 Evaluation Activities.

Additionally in this case, the evaluator shall verify that the operational guidance discusses the characteristics of external authorization factors (e.g., how the authorization factor must be generated; format(s) or standards that the authorization factor must meet) that are able to be used by the TOE. If other authorization factors are specified, then for each factor, the TSS specifies how the factors are input into the TOE.

Requirement met by the platform:

The evaluator shall examine the TSS to ensure a description is included for how the TOE is invoking the platform functionality and how it is getting an authorization value that has appropriate entropy.

Section 6.3 of the ST states the TOE utilizes the Knox work profile authentication service to acquire the user password so MKDD can be decrypted and the encrypted data unlocked. The platform provides the password directly to the TOE (without modification).

If the TOE is configured to encrypt the whole device, it provides its own authentication dialog to acquire the user password to decrypt the MKDD and unlock encrypted data.

The provided password is conditioned by the TOE with 100,000 rounds of PBKDF2 (using HMAC-SHA-256) to generate a 256-bit KEK.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance includes instructions for all of the authorization factors. The AGD will discuss the characteristics of external authorization factors (e.g., how the authorization factor is generated; format(s) or standards that the authorization factor must meet, configuration of the TPM device used) that are able to be used by the TOE.

Section 2.4.1.1 of the Admin Guide discusses passwords. It explains a user must set a password and then provides a reference to NIST SP 800-63B, section 5.1.1, Memorized Secrets for information on selecting a strong password.

Component Testing Assurance Activities: The evaluator shall ensure that authorization using each selected method is tested during the course of the evaluation, setting up the method as described in the operational guidance and ensuring that authorization is successful and that failure to provide an authorization factor results in denial to access to plaintext data.

[conditional]: If there is more than one authorization factor, ensure that failure to supply a required authorization factor does not result in access to the decrypted plaintext data.

Test 1 – (Knox Container case) The DualDAR work profile and test device were locked and the evaluator was unable to decrypt a file; the test device was unlocked (with the DualDAR work profile still locked) and the evaluator was still unable to decrypt a file; a failed attempt (wrong password) to unlock the DualDAR work profile is made and the evaluator was unable to decrypt a file; and finally the DualDAR work profile is unlocked with the correct authentication factor and the evaluator was able to successfully decrypt a file.

(Whole device case) The DualDAR device was locked and the evaluator was unable to decrypt a file; the test device was unlocked (with the DualDAR password not yet entered) and the evaluator was still unable to decrypt a file; a failed attempt (wrong password) to unlock the DualDAR is made and the evaluator was unable to decrypt a file; and finally the DualDAR is unlocked with the correct authentication factor and the evaluator was able to successfully decrypt a file.

Test 2 - Not applicable – there is only one claimed authorization factor (password).

2.4 SECURITY MANAGEMENT (FMT)

2.4.1 Secure by Default Configuration (ASPP14:FMT_CFG_EXT.1)

2.4.1.1 ASPP14:FMT_CFG_EXT.1.1

TSS Assurance Activities: The evaluator shall check the TSS to determine if the application requires any type of credentials and if the application installs with default credentials.

Section 6.4 of the ST explains that depending on the FEB configuration, the TOE either relies on the platform to handle authentication credentials or manages them directly.

When the FEB is configured to encrypt a Knox work profile, the TOE relies on the platform to handle authentication credentials. As part of the setup of the Knox work profile for the evaluated configuration the user is required to setup their credentials.

When the FEB is configured to encrypt the device, the TOE provides its own configuration to manage the authentication credentials. The TOE does not provide any default credentials and as part of the setup of this configuration the user is required to enter their credentials. The user cannot complete the setup without entering these credentials.

The TOE relies on Android permissions to restrict access to files created by the TOE.

The TOE is a service bundled as part of the Knox software on the device. The TOE does not have its own data folder but is a feature within the wider Knox software. The configuration of the TOE is maintained and protected within the Knox configuration.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If the application uses any default credentials the evaluator shall run the following tests.

Test 1: The evaluator shall install and run the application without generating or loading new credentials and verify that only the minimal application functionality required to set new credentials is available.

Test 2: The evaluator shall attempt to clear all credentials and verify that only the minimal application functionality required to set new credentials is available.

Test 3: The evaluator shall run the application, establish new credentials and verify that the original default credentials no longer provide access to the application.

Tests 1, 2, & 3 - Not applicable – the TOE does not use any default credentials.

2.4.1.2 ASPP14:FMT_CFG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

Platforms: Android....

The evaluator shall run the command find -L . -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Platforms: Microsoft Windows....

The evaluator shall run the SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. For Windows Universal Applications the evaluator shall consider the requirement met because of the AppContainer sandbox.

Platforms: Apple iOS....

The evaluator shall determine whether the application leverages the appropriate Data Protection Class for each data file stored locally.

Platforms: Linux....

The evaluator shall run the command find -L. -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Platforms: Oracle Solaris....

The evaluator shall run the command find . -perm -002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Platforms: Apple macOS....

The evaluator shall run the command find . -perm +002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Test – The evaluator identified the application folders for the test. The evaluator then issued the ls -alR|grep -E '^.....w.' command over the application folder to show no world writeable files were identified.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.2 Supported Configuration Mechanism - per TD0624 (ASPP14:FMT_MEC_EXT.1)

2.4.2.1 ASPP14:FMT_MEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall review the TSS to identify the application's configuration options (e.g. settings) and determine whether these are stored and set using the mechanisms supported by the platform or implemented by the application in accordance with the PP-Module for File Encryption. At a minimum the TSS shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an SFR.

Conditional: If 'implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption' is selected, the evaluator shall ensure that the TSS identifies those options, as well as indicates where the encrypted representation of these options is stored.

Section 6.4 of the ST states that the TOE configuration is stored as part of the device or the Knox work profile configuration and is not maintained separately.

The TOE is a service bundled as part of the Knox software on the device. The TOE does not have its own data folder but is a feature within the wider Knox software. The configuration of the TOE is maintained and protected within the Knox configuration.

Section 2.1 of the Admin Guide supports this by stating the Knox File Encryption service runs in the background and utilizes the Samsung Android cryptographic modules included in the platform to provide file encryption services. The service is designed to run without any user intervention and all files (as determined by the configuration) will be encrypted automatically. It is an integrated component of the device image, and is not a separately installed app.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If 'invoke the mechanisms recommended by the platform vendor for storing and setting configuration options' is chosen, the method of testing varies per platform as follows:

Platforms: Android....

The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least one file exists at location /data/data/package/shared_prefs/ (for SharedPreferences) and/or /data/data/package/files/datastore (for DataStore), where the package is the Java package of the application. For SharedPreferences the evaluator shall examine the XML file to make sure it reflects the changes made to the configuration to verify that the application used SharedPreferences and/or PreferenceActivity to store the configuration data. For DataStore the evaluator shall use a protocol buffer analyzer to examine the file to make sure it reflects the changes made to the configuration to verify that the application used DataStore to store the configuration data.

Platforms: Microsoft Windows....

The evaluator shall determine and verify that Windows Universal Applications use either the Windows.Storage namespace, Windows.UI.ApplicationSettings namespace or the IsolatedStorageSettings namespace for storing application specific settings. For .NET applications, the evaluator shall determine and verify that the application uses one of the locations listed in https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/ for storing application specific settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify that ProcMon logs show corresponding changes to the Windows Registry or C:directory.

Platforms: Apple iOS....

The evaluator shall verify that the app uses the user defaults system or key-value store for storing all settings.

Platforms: Linux....

The evaluator shall run the application while monitoring it with the utility strace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that strace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration), in the user's home directory (for user-specific configuration), or /var/lib/ (for configurations controlled by UI and not intended to be directly modified by an administrator).

Platforms: Oracle Solaris....

The evaluator shall run the application while monitoring it with the utility dtrace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that dtrace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration) or in the user's home directory (for user-specific configuration).

Platforms: Apple macOS....

The evaluator shall verify that the application stores and retrieves settings using the NSUserDefaults class.

If ' implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption' is selected, for all configuration options listed in the TSS as being stored and protected

using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

Test – (Knox Container mode) - The only configuration data stored by the TOE is its data lock timeout value and that is stored with the rest of the work profile configuration settings. The evaluator changed the lock timeout to 75 seconds and demonstrated it was enforced.

(Whle Device Mode) - - The only configuration data stored by the TOE is its data lock timeout and its minimum DualDAR password length and that are stored with the rest of the device DualDAR configuration settings. The evaluator changed the lock timeout to 10 minutes and changed the minimum DualDAR password length to 6 and demonstrated both were enforced.

2.4.3 Specification of Management Functions (ASPP14:FMT_SMF.1)

2.4.3.1 ASPP14:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall verify that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

The only management requirement specified for both modes is to “specify a time period after device lock to enter the Data Lock state.” Section 2.3.1 of the Admin Guide provides the setting to address this requirement. Section 2.3.2 provides the interfaces for minimum password length and resetting the token.

Component Testing Assurance Activities: The evaluator shall test the application's ability to provide the management functions by configuring the application and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

Test – (Knox Container mode) The evaluator tested the lockout timeout value. The DualDAR work profile time limit was set to 60 seconds and an encrypted file was decrypted. After waiting just over 1 minute, the file could no longer be decrypted. The workspace was unlocked and the timeout changed to 300 seconds. Again, the file was decrypted and then the file could also be decrypted at about 4 minutes, while it could no longer be decrypted at over 5 minutes as expected.

(Whole device mode) The evaluator tested the lockout timeout value. The DualDAR time limit was set to 300 seconds and an encrypted file was decrypted. After waiting just over 5 minutes, the file could no longer be decrypted. The DualDAR and device was unlocked and the timeout changed to 600 seconds. Again, the file was decrypted and then the file could also be decrypted at about 9 minutes, while it could no longer be decrypted at over 10 minutes as expected.

In addition to setting the data lock timeout limit, in Device Owner deployments, the DualDAR TOE has two additional management functions: 1) the minimum DualDAR password length can be set (this applies when a second password is explicitly configured for DualDAR encryption), and 2) a password reset token can be generated that is held by the MDM to be used to reset the DualDAR password when it might otherwise not be known.

The tester set the minimum password length for DualDAR to be 6. The tester then changed the password to a six character password, observing that a 5 character password would not be accepted. The tester set the minimum password length for DualDAR to be 8. The tester then changed the password to an eight character password, observing that a 7 character password would not be accepted.

For the reset token, the tester used the MDM app to first ensure there was no reset token. The tester then generated a reset token and then made it Active. The tester then verified there was a reset token. Finally, the evaluator changed the password using the reset token.

2.4.4 Specification of File Encryption Management Functions (FE10:FMT_SMF.1(2))

2.4.4.1 FE10:FMT_SMF.1.1(2)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Conditional Activities: The evaluator shall examine the TSS to ensure that it describes the sequence of activities that take place from an implementation perspective when this activity is performed (for example, how it determines which resources are associated with the KEK, the decryption and re-encryption process), and ensure that the KEK and FEK are not exposed during this change.

Cryptographic Configuration: None for this requirement.

Section 6.4 of the ST states that if the FEB is set for a Knox work profile, when the work profile is removed (such as by unenrollment or by policy by failed login attempts as defined by the work profile management), the MKDD will be erased. If the FEB configuration is for the whole device, the device must be factory reset to erase the MKDD (either by local control or by MDM).

Component Guidance Assurance Activities: Conditional Activities: The evaluator shall examine the Operational Guidance to ensure that it describes how the password/passphrase-based authorization factor is to be changed.

Cryptographic Configuration: The evaluator shall determine from the TSS for other requirements (FCS_*, FDP_PRT_EXT, FIA_AUT_EXT) what portions of the cryptographic functionality are configurable. The evaluator shall then review the AGD documentation to determine that there are instructions for manipulating all of the claimed mechanisms.

Section 2.4.1 of the Admin Guide states the user must configure a password for the Knox work profile. Detailed instructions for configuring these methods can be found under “Change unlock method” in the Knox work profile Guide.

Component Testing Assurance Activities: Cryptographic Erase: If the TOE uses stored FEKS or KEKs, the evaluator shall examine the key chain to determine that the keys destroyed by a cryptographic erase will result in the data becoming unrecoverable. Testing for this activity is performed for other components in this PP-Module.

Test - The ST explains that the PMKEK is derived from the work profile credential and used to decrypt MKDD which is used to decrypt the FEKs to in turn decrypt files. Once the MKDD is destroyed (e.g., with the work profile), FEKs can no longer be determined and hence affected files can no longer be decrypted.

No direct test activities are defined for this component.

2.5 Privacy (FPR)

2.5.1 User Consent for Transmission of Personally Identifiable (ASPP14:FPR_ANO_EXT.1)

2.5.1.1 ASPP14:FPR_ANO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted.

Section 6.5 of the ST states the TOE does not transmit any information over a network.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If require user approval before executing is selected, the evaluator shall run the application and exercise the functionality responsibly for transmitting PII and verify that user approval is required before transmission of the PII.

Test - Not applicable – the TOE does not transmit any information over a network.

2.6 Protection of the TSF (FPT)

2.6.1 Anti-Exploitation Capabilities (ASPP14:FPT_AEX_EXT.1)

2.6.1.1 ASPP14:FPT_AEX_EXT.1.1

TSS Assurance Activities: The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

Section 6.6 of the ST states the TOE components are all compiled with the -fstack-protector option set to enable stack-based buffer overflow protection.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform. For those platforms requiring the same application running on two different systems, the evaluator may alternatively use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings.

Platforms: Android....

The evaluator shall run the same application on two different Android systems. Both devices do not need to be evaluated, as the second device is acting only as a tool. Connect via ADB and inspect /proc/PID/maps. Ensure the two different instances share no memory mappings made by the application at the same location.

Platforms: Microsoft Windows....

The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

Platforms: Apple iOS....

The evaluator shall perform a static analysis to search for any mmap calls (or API calls that call mmap), and ensure that no arguments are provided that request a mapping at a fixed address.

Platforms: Linux....

The evaluator shall run the same application on two different Linux systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Platforms: Oracle Solaris....

The evaluator shall run the same application on two different Solaris systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Platforms: Apple macOS....

The evaluator shall run the same application on two different Mac systems. The evaluator shall then compare their memory maps using vmmap PID to ensure the two different instances share no mapping locations.

Test - The evaluator collected a /proc/PID/maps for the running process from each device and compared the results. The mappings were different between the devices. Additionally, the evaluator also collected the kernel address location 5 times (each after a reboot) for each test device since the TOE utilizes kernel drivers. The kernel addresses were different across the 5 boots.

2.6.1.2 ASPP14:FPT_AEX_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

Platforms: Android....

The evaluator shall perform static analysis on the application to verify that

o mmap is never invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked.

Platforms: Microsoft Windows....

The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the /NXCOMPAT flag was used during compilation to verify that DEP protections are enabled for the application.

Platforms: Apple iOS....

The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

Platforms: Linux....

The evaluator shall perform static analysis on the application to verify that both

o mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Oracle Solaris....

The evaluator shall perform static analysis on the application to verify that both

o mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Apple macOS....

The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

Test - The evaluator decompiled the apk. After the decompilation, the evaluator performed a search for the mmap, mprotect permissions. No other occurrences were identified by the evaluator and therefore the evaluator confirmed that neither is invoked within the implementation of the TOE.

2.6.1.3 ASPP14:FPT_AEX_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

Platforms: Android....

Applications running on Android cannot disable Android security features, therefore this requirement is met and no evaluation activity is required.

Platforms: Microsoft Windows....

If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data

Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-us/windows/security/threatprotection/windows-defender-exploit-guard/customize-exploit-protection.

If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

Platforms: Apple iOS....

Applications running on iOS cannot disable security features, therefore this requirement is met and no evaluation activity is required.

Platforms: Linux....

The evaluator shall ensure that the application can successfully run on a system with either SELinux or AppArmor enabled and in enforce mode.

Platforms: Oracle Solaris....

The evaluator shall ensure that the application can run with Solaris Trusted Extensions enabled and enforcing.

Platforms: Apple macOS....

The evaluator shall ensure that the application can successfully run on macOS without disabling any security features.

Test - Not applicable - applications running on Android cannot disable Android security features.

2.6.1.4 ASPP14:FPT_AEX_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:

Platforms: Android....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored under /data/data/package/ where package is the Java package of the application.

Platforms: Microsoft Windows....

For Windows Universal Applications the evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Platforms: Apple iOS....

The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

Platforms: Linux....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Platforms: Oracle Solaris....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Platforms: Apple macOS....

The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Test – The evaluator used the app to encrypt and decrypt files (as part of ASPP14:FMT_MEC_EXT.1 testing). The evaluator then searched the /data partition of both modes to ensure no executables were stored there.

2.6.1.5 ASPP14:FPT_AEX_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

Platforms: Microsoft Windows....

Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element.

For other code, the evaluator shall review the TSS and verify that the /GS flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of /GS.

For PE , the evaluator will disassemble each and ensure the following sequence appears:

mov rcx, QWORD PTR [rsp+(...)]

xor rcx, (...)

call (...)

For ELF executables, the evaluator will ensure that each contains references to the symbol _stack_chk_fail.

Tools such as Canary Detector may help automate these activities.

Test - The evaluator obtained the actual ELF binaries and used the strings command to identify fstack protection markers as follows – "__stack_chk_fail" was found in each case.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.2 Use of Supported Services and APIs (ASPP14:FPT_API_EXT.1)

2.6.2.1 ASPP14:FPT_API_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS lists the platform APIs used in the application.

Section 6.6 of the ST explains the TOE uses only platform provided APIs as noted below:

- All
  - Android SDK API Level 33
  - Samsung Knox API Level 36
- DualDAR Client
  - Samsung SCrypto Module – AES operations, HMAC-SHA, DRBG

- DualDAR Driver
    - Samsung Kernel Cryptographic Module – AES operations, DRBG

Android SDK API Level 33 - The Android SDK is readily available online for level 33 (Android 13) – see https://developer.android.com/studio/releases/platforms.

Samsung Knox API Level 36 **-** The Samsung Knox SDK is also available – see Knox SDK 3.9 https://docs.samsungknox.com/dev/knox-sdk/index.htm

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall then compare the list with the supported APIs (available through e.g. developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

Test - The developer provided SDKs for the Qualcomm/QSEE (G986B/G998B/S908B) and Exynos/TEGRIS (G986U/G998U/S901U/S918U) platforms as well as Kernel Crypto headers for each platform and Scrypto module headers. The evaluator was able to verify the DRBG API calls. Additionally, the evaluator used the Android and Knox APIs to verify those APIs.

2.6.3 Software Identification and Versions (ASPP14:FPT_IDV_EXT.1)

2.6.3.1 ASPP14:FPT_IDV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'other version information' is selected the evaluator shall verify that the TSS contains an explanation of the versioning methodology.

Section 6.6 of the ST states that the platform user interface provides a method to query the current version of many components, including the TOE software. The TOE software version can be viewed in the *About Phone/Software information* section in Settings. Under the Knox version section of the display, the software version can be seen as the DualDAR version. The DualDAR Version in Table 1 - Evaluated Devices shows the specific software version evaluated.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall install the application, then check for the / existence of version information. If SWID tags is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

Test – The evaluator checked for the version in the Software Information -> Knox Version setting and verified its existence.

2.6.4 Protection of Keys and Key Material (FE10:FPT_KYP_EXT.1)

2.6.4.1 FE10:FPT_KYP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS for a high level description of the method(s) used to protect keys stored in non-volatile memory.

KMD

The evaluator shall verify the KMD to ensure it describes the storage location of all keys and the protection of all keys stored in non-volatile memory. The description of the key chain shall be reviewed to ensure FCS_COP.1(5) is followed for the storage of wrapped or encrypted keys in non-volatile memory and plaintext keys in non-volatile memory meet one of the criteria for storage.

Section 6.6 of the ST explains the TOE stores the MKDD and the FEKs separately in non-volatile memory. The MKDD is stored in the TrustZone where it is wrapped with a key derived from the credentials of the Knox work profile password (using PBKDF2).

The TOE stores a FEK as part of the file metadata. The FEK is encrypted using AES-GCM with the MKDD KEK.

KMD –

Section 2.2 provides a detailed diagram and description of the key hierarchy. The discussion and diagram match the claims in the FCS_COP.1(5) requirement.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.5 Use of Third Party Libraries (ASPP14:FPT_LIB_EXT.1)

2.6.5.1 ASPP14:FPT_LIB_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall install the application and survey its installation directory for dynamic libraries. The evaluator shall verify that libraries found to be packaged with or employed by the application are limited to those in the assignment.

Test - The evaluator searched the /data partition dumps of both modes to identify applicable folders. The evaluator found no evidence of any 3rd party libraries in the numerous folders identified.

2.6.6 Integrity for Installation and Update (ASPP14:FPT_TUD_EXT.1)

2.6.6.1 ASPP14:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall check to ensure the guidance includes a description of how updates are performed.

Section 3.1 of the Admin Guide explains how to check for and perform updates of the device. Since the TOE is bundled with the platform, the platform provides the ability to check for and install updates.

Testing Assurance Activities: The evaluator shall check for an update using procedures described in either the application documentation or the platform documentation and verify that the application does not issue an error. If it is updated or if it reports that no update is available this requirement is considered to be met.

Test - The evaluator attempted to update all devices and found no available updates. The evaluator continued testing and found the TOE to continue to operate as expected.

2.6.6.2 ASPP14:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall verify guidance includes a description of how to query the current version of the application.

Section 3.2 of the Admin Guide has the procedure for checking the current version of the TOE.

Testing Assurance Activities: The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

Test - See Test Case ASPP14:FPT_IDV_EXT.1.1 where the TOE version is examined. The evaluator found that the version matches the ST and guidance as required.

2.6.6.3 ASPP14:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall verify that the application's executable files are not changed by the application.

Platforms: Apple iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

For all other platforms, the evaluator shall perform the following test:

Test 1: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

Test - First the evaluator found the TOE binary, captured a copy and recorded the sha1 hash both on the phone platform and for the file pulled from the phone) for each test device. The evaluator then performed the operations and finally accessed encrypted files and repeated the hash collection procedures. The hashes remain unchanged for the TOE binary on the test devices.

2.6.6.4 ASPP14:FPT_TUD_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS identifies how updates to the application are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.

Section 6.6 of the ST states the TOE relies on the platform for all updates as it is considered a part of the operating system. The platform provides the ability to check for and install updates.

Section 3.1 of the Admin Guide explains when updates are made available, they are signed by Samsung with a private key that is unique to the device/carrier combination (i.e. a Galaxy S20 on Verizon will not have an update signed with the same key as a Galaxy S20 on AT&T). The public key is embedded in the bootloader image, and is used to verify the integrity and validity of the update package. This signature covers the entirety of the update, including any updates for Knox File Encryption.

Section 3.1 of the Admin Guide also explains how to check for updates and then how to update the device.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.6.5 ASPP14:FPT_TUD_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS identifies how the application is distributed.

Section 6.6 of the ST states the TOE relies on the platform for all updates as it is considered a part of the operating system.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If 'with the platform' is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If 'as an additional package' is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

Test - After flashing the devices and normal Android initialization, the evaluator was able to utilize the TOE without installing any additional applications other than a test application that is necessary to issue the commands to create a Knox Workspace container where DualDAR is enabled.

All other testing was performed on the newly flashed test phones with a Knox container with DualDAR enabled without installing additional applications except as are identified in the applicable test (e.g., Avast Mobile Security).

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7 TRUSTED PATH/CHANNELS (FTP)

2.7.1 PROTECTION OF DATA IN TRANSIT - PER TD0655 (ASPP14:FTP_DIT_EXT.1)

2.7.1.1 ASPP14:FTP_DIT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.

Section 6.7 of the ST states the TOE does not transmit any data over a network.

Component Guidance Assurance Activities: None Defined

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

Test 1: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS, TLS, DTLS, SSH, or IPsec in accordance with the selection in the ST.

Test 2: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.

Test 3: The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

Platforms: Android....

If 'not transmit any data' is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a uses-permission or uses-permission-sdk-23 tag containing android:name='android.permission.INTERNET'. In this case, it is not necessary to perform the above Tests 1, 2, or 3, as the platform will not allow the application to perform any network communication.

Platforms: Apple iOS....

If 'encrypt all transmitted data' is selected, the evaluator shall ensure that the application's Info.plist file does not contain the NSAllowsArbitraryLoads or NSExceptionAllowsInsecureHTTPLoads keys, as these keys disable iOS's Application Transport Security feature.

Test - The evaluator examined the AndroidManifest.xml file and found no instances of “INTERNET”. The remaining components are kernel drivers which do not have access to the network stack.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the PP_APP_V1.4/MOD_FE_V1.0 that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: There are no specific assurance activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

The assurance activities from the PP_APP_V1.4 have been performed and the analysis of the evaluator is documented in the previous sections of this document.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: Some of the contents of the operational guidance will be verified by the assurance activities in Section 5.1 and evaluation of the TOE according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE. The documentation must describe the process for verifying updates to the TOE by verifying a digital signature -this may be done by the TOE or the underlying platform. The evaluator shall verify that this process includes the following steps: Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory). Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature. The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

No user configuration is available for the cryptographic features of the TOE.

Section 3. 1 of the Admin Guide describes updates including the how to obtain them and install them.

Section 2.1 explains the security feature being evaluated. It clearly states the scope of the product. Samsung Knox File Encryption is a software service designed to provide a second layer of encryption to files stored on the device independent of the default file encryption for the device.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As indicated in the introduction above, there are significant expectations with respect to the documentation - especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

Section 1.4 identifies all the platforms for the evaluation. The instructions address all the platforms.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

The Admin Guide identifies a specific version of the TOE for which it is relevant.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: The 'evaluation evidence required by the SARs' in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

The evaluator shall ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler flags). The evaluator shall ensure that this documentation also

includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

The Admin Guide identifies a specific version of the TOE for which it is relevant and identifies the platforms for the evaluation.

3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activities: The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall verify that this description addresses the entire application.

The evaluator shall also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described. The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days. The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE.

The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

Section 6.6 of the ST explains that Samsung utilizes industry best practices to ensure their devices and all components on the device are patched to mitigate security flaws. Samsung provides a web portal for reporting potential security issues (https://security.samsungmobile.com/securityReporting.smsb) with instructions about how to securely contact and communicate with Samsung.

Samsung will create updates and patches to resolve reported issues as quickly as possible, at which point the update is provided to the wireless carriers. The delivery time for resolving an issue depends on the severity, and can be as rapid as a few days before the carrier handoff for high priority cases. The wireless carriers perform additional tests to ensure the updates will not adversely impact their networks and then plan device rollouts once that testing is complete. Carrier updates usually take at least two weeks to as much as two months (depending on the type and severity of the update) to be rolled out to customers. However, the Carriers also release monthly Maintenance Releases in order to address security-critical issues, and Samsung itself maintains a security blog (http://security.samsungmobile.com) in order to disseminate information directly to the public.

Samsung communicates with the reporting party to inform them of the status of the reported issue. Further information about updates is handled through the carrier release notes. Issues reported to Google directly are handled through Google's notification processes.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's Assurance Activities.

While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS, SSH). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a 'fail' and 'pass' result (and the supporting details), and not just the 'pass' result.

The evaluator created a proprietary Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The evaluator used the following test configuration:

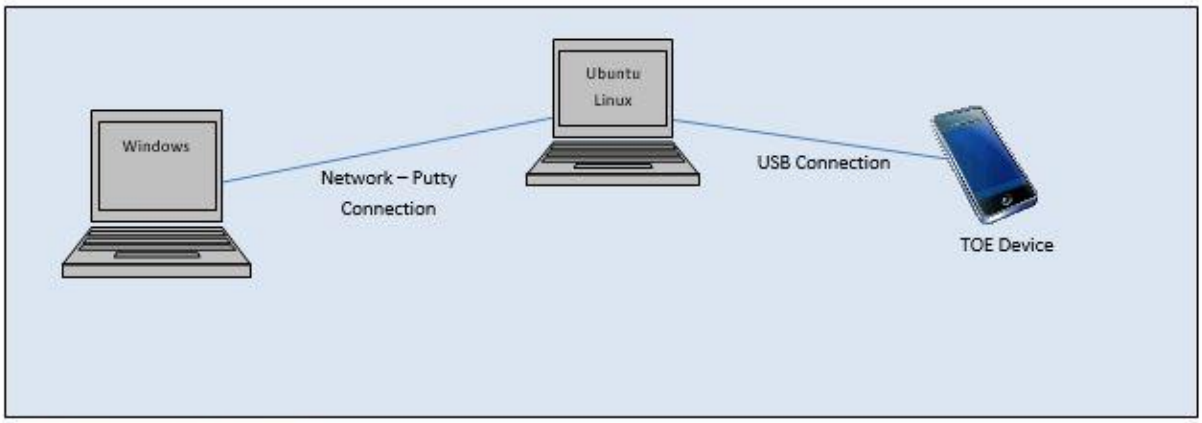

Figure 1 Evaluator Test Setup 1

The TOE was fully tested on a Samsung Galaxy S23 (Qualcomm), Samsung Galaxy S22 (Qualcomm and Exynos), Samsung Galaxy S21 (Qualcomm and Exynos), and Samsung Galaxy S20 (Qualcomm and Exynos). The evaluator used the following test tools: Windows, Linux, Putty, adb, HxD, Samsung developed test programs, Windows Security, Virus & threat protection, and Avast Mobile Security.

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

The evaluator searched the National Vulnerability Database (https://web.nvd.nist.gov/vuln/search), Vulnerability Notes Database (http://www.kb.cert.org/vuls/) http://www.kb.cert.org/vuls/) on 03/17/2023 with the following search terms Samsung S23, Galaxy S23, Samsung S23+, Galaxy S23+, SM-S918, SM-S916, SM-S911, Samsung S22, Galaxy S22, Samsung S22+, Galaxy S22+, SM-S908, SM-S906, SM-S901, Samsung S21, Galaxy S21, Samsung S21+, Galaxy S21+, SM-G998, SM-G996, SM-G991, SM-G990, Samsung S20, Galaxy S20, Samsung S20+, Galaxy S20+, SM-G988, SM-G986, SM-G985, SM-G981, SM-G980, SM-G781, SM-G780, Samsung Note20, Galaxy Note20, SM-N986, SM-N985, SM-N981, SM-N980, Samsung Tab S8, Galaxy Tab S8, Samsung Tab S8+, Galaxy Tab S8+, SM-X906, SM-

X900, SM-X808, SM-X806, SM-X800, SM-X708, SM-X706, SM-X700, Samsung Tab S7, Galaxy Tab S7, Samsung Tab S7+, Galaxy Tab S7+, SM-T978, SM-T976, SM-T975, SM-T970, SM-T878, SM-T875, SM-T870, Samsung Z Flip, Galaxy Z Flip, SM-F707, SM-F700, Samsung Z Flip4, Galaxy Z Flip4, SM-F721, Samsung Z Flip3, Galaxy Z Flip3, SM-F711, Samsung Z Fold4, Galaxy Z Fold4, SM-F936, Samsung Z Fold3, Galaxy Z Fold3, SM-F926, Samsung Z Fold2, Galaxy Z Fold2, SM-F916, Knox, DualDAR, BoringSSL, containercore, and Android.

Additionally, the evaluator ran Windows Defender against the apk to ensure it contained no viruses. The evaluator also installed the Avast Mobile Security antivirus app on each TOE device, checked to ensure the definitions were current, and ran a scan on each device. In each case no issues were identified.