# MicroCloud X4 Security Target

Document ID

Version: 1.0

August 28, 2017

**Prepared For:**

Bluechip Systems LLC

2350 Mission College Blvd.

Suite 290

Santa Clara, CA 95054

**Prepared By:**

Kenji Yoshino

UL Verification Services Inc.

## Table of Contents

# 1 Security Target (ST) Introduction

The structure of this document is defined by CC v3.1r4 Part 1 Annex A.2, “Mandatory contents of an ST”:

- Section 1 contains the ST Introduction, including the ST reference, Target of Evaluation (TOE) reference, TOE overview, and TOE description.
- Section 2 contains conformance claims to the Common Criteria (CC) version, Protection Profile (PP) and package claims, as well as rationale for these conformance claims.
- Section 3 contains the security problem definition, which includes threats, Organizational Security Policies (OSP), and assumptions that must be countered, enforced, and upheld by the TOE and its operational environment.
- Section 4 contains statements of security objectives for the TOE, and the TOE operational environment as well as rationale for these security objectives.
- Section 5 contains definitions of any extended security requirements claimed in the ST.
- Section 6 contains the security functional requirements (SFR), the security assurance requirements (SAR), as well as the rationale for the claimed SFR and SAR.
- Section 7 contains the TOE summary specification, which includes the detailed specification of the IT security functions

## 1.1 Security Target Reference

The Security Target reference shall uniquely identify the Security Target.

| | |
|---|---|
| ST Title: | MicroCloud X4 Security Target |
| ST Version Number: | Version 1.0 |
| ST Author(s): | Kenji Yoshino |
| ST Publication Date: | August 28, 2017 |
| Keywords: | Full Drive Encryption, Encryption Engine, Authorization Acquisition |

## 1.2 Target of Evaluation Reference

The Target of Evaluation reference shall identify the Target of Evaluation.

| | |
|---|---|
| TOE Developer: | Bluechip Systems LLC<br>2350 Mission College Blvd.<br>Suite 290<br>Santa Clara, CA 95054 |
| TOE Name: | MicroCloud X4 |
| TOE Version: | MCX4-004 and MCX4-008 with MicroCloud Linux 3.4.110.1 and MicroCloud Manager 1.9 |

## 1.3 Target of Evaluation Overview

### 1.3.1 TOE Product Type

The TOE is classified as a full drive encryption device that performs Authorization Acquisition and Encryption Engine functions.

### 1.3.2 TOE Usage

The TOE is intended to be used in mobile devices. The TOE is a microSD card that provides data at rest (DAR) protections for user data stored on the TOE.

### 1.3.3 TOE Major Security Features Summary

- Cryptographic Support
- User Data Protection
- Security Management
- Protection of the TOE Security Functionality (TSF)

### 1.3.4 TOE IT environment hardware/software/firmware requirements

The TOE requires a Samsung Galaxy S5 SM-G900F host device running Android 4.6 – 6.0.

## 1.4 Target of Evaluation Description

### 1.4.1 Target of Evaluation Physical Boundaries

The TOE consists of the following hardware:

- MCX4-004 or MCX4-008
- InvenSense MPU-6500 Accelerometer (part of the host device)

and the following software and firmware:

- MicroCloud Linux 3.4.110.1 (running on the MCX4-004 or MCX4-008)
- MicroCloud Manager 1.9 (running on the MCX4-004 or MCX4-008)
- GreenFiles v1.8.0 (running on the host device)
- Bluechip DataHub Service v1.8.0 (running on the host device)

The guidance documentation that is part of the TOE is listed in Section 9, "References," within Table 10: TOE Guidance Documentation.

The MCX4-004 part number identifies TOE hardware with 4GB of storage. The MCX4-008 part number identifies TOE hardware with 8GB of storage.

Each part number runs identical firmware.

### 1.4.2 Target of Evaluation Logical Boundaries

The logical boundary of the TOE includes the security functions implemented exclusively by the TOE. These security functions are summarized in Section 1.3.3 above and are further described in the following subsections. A more detailed description of the implementation of these security functions are provided in Section 7, "TOE Summary Specification."

#### 1.4.2.1 Cryptographic Support

The TOE performs cryptographic operations using CAVP validated algorithms. The TOE performs random number generation, ECDSA signature verification, key derivation, and encryption/decryption to support full drive encryption functions.

#### 1.4.2.2 User Data Protection

The TOE encrypts all user data using ASE XTS with a 256 bit key.

#### 1.4.2.3 Security Management

The TOE allows users to change the data encryption key (DEK), cryptographically erase the DEK, and initiate firmware updates.

#### 1.4.2.4 Protection of the TSF

The TOE protects itself by running a suite of self-tests at power-up and by authenticating firmware updates by verifying a digital signature. The TOE does not store plaintext submasks in non-volatile memory.

## 1.5 Notation, Formatting, and Conventions

The notation, formatting, and conventions used in this Security Target are defined below; these styles and clarifying information conventions were developed to aid the reader.

Where necessary, the ST author has added application notes to provide the reader with additional details to aid understanding; they are italicized and usually appear following the element needing clarification. Those notes specific to the TOE are marked “ST Application Note;” those taken from the Protection Profiles (PPs) are marked “PP Application Note.”

The notation conventions that refer to iterations, assignments, selections, and refinements made in this Security Target are in reference to SARs and SFRs taken directly from CC Part 2 and Part 3 as well as any SFRs and SARs taken from a Protection Profile.

The notation conventions used in the PPs to indicate assignments, selections, and refinements of SARs and SFRs taken from CC Part 2 and Part 3 are not carried forward into this document. Additionally, obvious errors in the PPs are corrected and noted as such.

The CC permits four component operations (assignment, iteration, refinement, and selection) to be performed on requirement components. These operations are defined in Common Criteria, Part 1; paragraph 6.4.1.3.2, “Permitted operations on components” as:

- Iteration: allows a component to be used more than once with varying operations;
- Assignment: allows the specification of parameters;
- Selection: allows the specification of one or more items from a list; and
- Refinement: allows the addition of details.

Iterations performed by the ST Author are indicated by an identifier in parenthesis following the requirement number, e.g., FIA_UAU.1.1(1); the iterated requirement titles are similarly indicated, e.g., FIA_UAU.1(1). Iterations resulting from differences between the EE and AA PPs use and identifier that references the PP, e.g., FPT_KYP_EXT.1.1(AA).

Iterations performed in the Protection Profile are indicated by a letter in parenthesis following the requirement number, e.g., FCS_COP.1.1(c); the iterated requirement titles are similarly indicated, e.g., FCS_COP.1(c).

Assignments made by the ST author are identified with **bold text.**

Selections are identified with <u>underlined text</u>**.**

Refinements that add text use ***bold and italicized text*** to identify the added text***.*** Refinements that perform a deletion are described in an ST Application Note.

# 2 Conformance Claims

## 2.1 Common Criteria Conformance Claims

This Security Target is conformant to the Common Criteria Version 3.1r4, CC Part 2 extended [4], and CC Part 3 extended [5].

## 2.2 Conformance to Protection Profiles

This Security Target claims exact compliance to the collaborative Protection Profile for Full Drive Encryption – Encryption Engine, Version 1.0, January 26, 2015 and the collaborative Protection Profile for Full Disk Encryption – Authorization Acquisition, Version 1.0, January 26, 2015. These Protection Profiles will be referred to collectively as FDE or cPP for convenience throughout this Security Target.

The TOE complies with all technical decisions issued prior to July 26, 2017. The TOE complies with the update specified in TQ 150.

## 2.3 Conformance to Security Packages

This Security Target does not claim conformance to any security function requirements or security assurance requirements packages, neither as package-conformant or package-augmented.

## 2.4 Conformance Claims Rationale

To demonstrate that exact conformance is met, this rationale shows all threats are addressed, all Organizational Security Policies (OSP) are satisfied, no additional assumptions are made, all objectives have been addressed, and all SFRs and SARs have been instantiated.

The following address the completeness of the threats, OSP, and objectives, limitations on the assumptions, and instantiation of the SFRs and SARs:

- Threats
    - All threats defined in the cPP are carried forward to this ST;
    - No additional threats have been defined in this ST.
- Organizational Security Policies
    - All OSP defined in the cPP are carried forward to this ST;

- No additional OSPs have been defined in this ST.

- Assumptions
    - All assumptions defined in the cPP are carried forward to this ST;
    - No additional assumptions for the operational environment have been defined in this ST.
- Objectives
    - All objectives defined in the cPP are carried forward to this ST.
- All SFRs and SARs defined in the cPP are carried forward to this Security Target.

Rationale presented in the body of this ST shows all assumptions on the operational environment have been upheld, all the OSP are enforced, all defined objectives have been met and these objectives counter the defined threats.

Additionally, all SFRs and SARs defined in the cPP have been properly instantiated in this Security Target; therefore, this ST shows exact compliance to the cPP.

# 3 Security Problem Definition

## 3.1 Threats

The following table defines the security threats for the TOE, characterized by a threat agent, an asset, and an adverse action of that threat agent on that asset. These threats are taken directly from the cPP unchanged.

Table 1: Threats

| Threat | Description |
|---|---|
| T.UNAUTHORIZED_DATA_ ACCESS | The cPP addresses the primary threat of unauthorized disclosure of protected data stored on a storage device. If an adversary obtains a lost or stolen storage device (e.g., a storage device contained in a laptop or a portable external storage device), they may attempt to connect a targeted storage device to a host of which they have complete control and have raw access to the storage device (e.g., to specified disk sectors, to specified blocks). |
| T.KEYING_MATERIAL_ COMPROMISE | Possession of any of the keys, authorization factors, submasks, and random numbers or any other values that contribute to the creation of keys or authorization factors could allow an unauthorized user to defeat the encryption. The cPP considers possession of keying material of equal importance to the data itself. Threat agents may look for keying material in unencrypted sectors of the storage device and on other peripherals in the operating environment (OE), e.g. BIOS configuration, SPI flash, or TPMs. |
| T.AUTHORIZATION_GUESSIN G | Threat agents may exercise host software to repeatedly guess authorization factors, such as passwords and PINs. Successful guessing of the authorization factors may cause the TOE to release DEKs or otherwise put it in a state in which it discloses protected data to unauthorized users. |
| T.KEYSPACE_EXHAUST | Threat agents may perform a cryptographic exhaust against the key space. Poorly chosen encryption algorithms and/or parameters allow attackers to brute force exhaust the key space and give them unauthorized access to the data. |
| T.KNOWN_PLAINTEXT | Threat agents know plaintext in regions of storage devices, especially in uninitialized regions (all zeroes) as well as regions that contain well known software such as operating systems. A poor choice of encryption algorithms, encryption modes, and initialization vectors along with known plaintext could allow an attacker to recover the effective DEK, thus providing unauthorized access to the previously unknown plaintext on the storage device. |
| T.CHOSEN_PLAINTEXT | Threat agents may trick authorized users into storing chosen plaintext on the encrypted storage device in the form of an image, document, or some other file A poor choice of encryption algorithms, encryption modes, and initialization vectors along with the chosen plaintext could allow attackers to recover the effective DEK, thus providing unauthorized access to the previously unknown plaintext on the storage device. |
| T.UNAUTHORIZED_UPDATE | Threat agents may attempt to perform an update of the product which compromises the security features of the TOE. Poorly chosen update protocols, signature generation and verification algorithms, and parameters may allow attackers to install software and/or firmware that bypasses the intended security features and provides them unauthorized to access to data. |

## 3.2 Organizational Security Policies

There are no organizational security policies addressed by this cPP.

## 3.3 Assumptions

This section describes the assumptions on the operational environment in which the TOE is intended to be used. It includes information about the physical, personnel, and connectivity aspects of the environment. The operational environment must be managed in accordance with the provided guidance documentation. The following table defines specific conditions that are assumed to exist in an environment where the TOE is deployed. These assumptions are taken directly from the PP unchanged.

Table 2: Assumptions

| Assumption | Description |
|---|---|
| A.TRUSTED_CHANNEL | Communication among and between product components (e.g., AA and EE) is sufficiently protected to prevent information disclosure. In cases in which a single product fulfils both cPPs, then the communication between the components does not extend beyond the boundary of the TOE (e.g., communication path is within the TOE boundary). In cases in which independent products satisfy the requirements of the AA and EE, the physically close proximity of the two products during their operation means that the threat agent has very little opportunity to interpose itself in the channel between the two without the user noticing and taking appropriate actions. |
| A. INITIAL_DRIVE_STATE | Users enable Full Drive Encryption on a newly provisioned storage device free of protected data in areas not targeted for encryption. It is also assumed that data intended for protection should not be on the targeted storage media until after provisioning. The cPP does not intend to include requirements to find all the areas on storage devices that potentially contain protected data. In some cases, it may not be possible - for example, data contained in “bad” sectors. While inadvertent exposure to data contained in bad sectors or un-partitioned space is unlikely, one may use forensics tools to recover data from such areas of the storage device. Consequently, the cPP assumes bad sectors, unpartitioned space, and areas that must contain unencrypted code (e.g., MBR and AA/EE preauthentication software) contain no protected data. |
| A.TRAINED_USER | Users follow the provided guidance for securing the TOE and authorization factors. This includes conformance with authorization factor strength, using external token authentication factors for no other purpose and ensuring external token authorization factors are securely stored separately from the storage device and/or platform. The user should also be trained on how to power off their system. |
| A.PLATFORM_STATE | The platform in which the storage device resides (or an external storage device is connected) is free of malware that could interfere with the correct operation of the product. |
| A.POWER_DOWN | The user does not leave the platform and/or storage device unattended until all volatile memory is cleared after a power-off. This properly clears memories and locks down the device, so memory remnant attacks are infeasible.<br>Authorized users do not leave the platform and/or storage device in a mode where sensitive information persists in non-volatile storage (e.g., Lockscreen or sleep state). Users power the platform and/or storage device down or place it into a power managed state, such as a “hibernation mode”. |

| Table 2: Assumptions | |
|---|---|
| Assumption | Description |
| A.STRONG_CRYPTO | All cryptography implemented in the Operational Environment and used by the product meets the requirements listed in the cPP. This includes generation of external token authorization factors by a RBG. |
| A.SUCURE_STATE | Upon the completion of proper provisioning, the drive is only assumed secure when in a powered off state up until it is powered on and receives initial authorization. |
| A.SINGLE_USE_ET | External tokens that contain authorization factors are used for no other purpose than to store the external token authorization factors. |
| A.PASSWORD_STRENGTH | Authorized administrators ensure password/passphrase authorization factors have sufficient strength and entropy to reflect the sensitivity of the data being protected. |
| A.PLATFORM_I&A | The product does not interfere with or change the normal platform identification and authentication functionality such as the operating system login. It may provide authorization factors to the Operating system's login interface, but it will not change or degrade the functionality of the actual interface. |

# 4 Security Objectives

## 4.1 Security Objectives for the Operational Environment

Table 3: Security Objectives for the Operational Environment

| Objective | Description |
|---|---|
| OE.TRUSTED_CHANNEL | Communication among and between product components (e.g., AA and EE) is sufficiently protected to prevent information disclosure. |
| OE.INITIAL_DRIVE_STATE | The OE provides a newly provisioned or initialized storage device free of protected data in areas not targeted for encryption. |
| OE.PASSPHRASE_STRENGTH | An authorized administrator will be responsible for ensuring that the passphrase authorization factor conforms to guidance from the Enterprise using the TOE. |
| OE.POWER_DOWN | Volatile memory is cleared after power-off so memory remnant attacks are infeasible. |
| OE.SINGLE_USE_ET | External tokens that contain authorization factors will be used for no other purpose than to store the external token authorization factor. |
| OE.TRAINED_USERS | Authorized users will be properly trained and follow all guidance for securing the TOE and authorization factors. |
| OE.STRONG_ENVIRONMENT_CRYPTO | The Operating Environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE and Appendix A. |
| OE.PLATFORM_STATE | The platform in which the storage device resides (or an external storage device is connected) is free of malware that could interfere with the correct operation of the product. |
| OE.PLATFORM_I&A | The Operational Environment will provide individual user identification and authentication mechanisms that operate independently of the authorization factors used by the TOE. |

# 5 Extended Components Definition

This section provides definition of the extended security functional and assurance requirements; the components that are CC Part 2 extended, and CC Part 3 extended, i.e., NIAP interpreted requirements, and extended requirements.

## 5.1 Extended Security Functional Requirements Definitions

There are no extended Security Functional Requirements defined in this Security Target. All extended SFRs were taken from the cPP.

## 5.2 Extended Security Assurance Requirements Definitions

There are no extended Security Assurance Requirements defined in this Security Target. All extended SARs were taken from the cPP.

# 6 Security Requirements

This section describes the security functional and assurance requirements for the TOE; those that are CC Part 2 conformant, CC Part 2 extended, CC Part 3 conformant, and CC Part 3 extended.

## 6.1 Security Functional Requirements

This section describes the functional requirements for the TOE. The security functional requirement components in this security target are CC Part 2 conformant or CC Part 2 extended as defined in Section 2, Conformance Claims. Operations that were performed in the cPP are not signified in this section. Operations performed by the ST are denoted according to the formatting conventions in Section 1.5.

Table 4: Security Functional Requirements

| # | SFR | Description |
|---|---|---|
| 1 | FCS_AFA_EXT.1 | Authorization Factor Acquisition |
| 2 | FCS_CKM.1 | Cryptographic key generation (Data Encryption Key) |
| 3 | FCS_CKM.1(c) | Cryptographic key generation (Symmetric Keys) |
| 4 | FCS_CKM_EXT.4 | Cryptographic Key and Key Material Destruction |
| 5 | FCS_CKM.4 | Cryptographic key destruction |
| 6 | FCS_COP.1(a) | Cryptographic Operation (Signature Verification) |
| 7 | FCS_COP.1(b) | Cryptographic Operation (Hash Algorithm) |
| 8 | FCS_COP.1(c) | Cryptographic Operation (Keyed Hash Algorithm) |
| 9 | FCS_COP.1(d) | Cryptographic operation (Key Wrapping) |
| 10 | FCS_COP.1(f) | Cryptographic Operation (AES Data Encryption/Decryption) |
| 11 | FCS_KYC_EXT.1 | Key Chaining |
| 12 | FCS_KYC_EXT.2 | Key Chaining |
| 13 | FCS_PCC_EXT.1 | Cryptographic Password Construct and Conditioning |
| 14 | FCS_RBG_EXT.1 | Cryptographic Operation (Random Bit Generation) |
| 15 | FCS_SMV_EXT.1 | Validation |
| 16 | FCS_SNI_EXT.1 | Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) |
| 17 | FDP_DSK_EXT.1 | Protection of Data on Disk |
| 18 | FMT_SMF.1 | Specification of Management Functions |
| 19 | FPT_KYP_EXT.1 | Protection of Key and Key Material |
| 20 | FPT_TUD_EXT.1 | Trusted Update |
| 21 | FPT_TST_EXT.1 | TSF Testing |

### 6.1.1 Class FCS: Cryptographic Support

#### 6.1.1.1 FCS_AFA_EXT.1 Authorization Factor Acquisition

**FCS_AFA_EXT.1.1**

The TSF shall accept the following authorization factors:

- a submask derived from a password authorization factor conditioned as defined in FCS_PCC_EXT.1.

***PP Application Note***

*This requirement specifies what authorization factors the TOE accepts from the user. A password entered by the user is one authorization factor that the TOE must be able to condition, as specified in FCS_PCC_EXT.1. Another option is a SmartCard authorization factor, with the differentiating feature is how the value is generated – either by the TOE's RBG or by the platform. An external USB token may also be used, with the submask value generated either by the TOE's RBG or by the platform.*

*The TOE may accept any number of authorization factors, and these are categorized as "submasks". The ST Author selects the authorization factors they support, and there may be multiple methods for a selection.*

*Use of multiple authorization factors is preferable; if more than one authorization factor is used, the submasks produced must be combined using FCS_SMC_EXT.1 specified in Appendix A.*

**Assurance Activity**

**TSS**

The evaluators shall first examine the TSS section 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_PCC_EXT.1 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.

**Operational Guidance**

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.

**KMD**

The evaluator shall examine the Key Management Description to confirm that the initial authorization factors (submasks) directly contribute to the unwrapping of the BEV.

The evaluator shall verify the KMD describes how a submask is produced from the authorization factor (including any associated standards to which this process might conform), and verification is performed to ensure the length of the submask meets the required size (as specified in this requirement).

**Test**

The password authorization factor is tested in FCS_PCC_EXT.1. The evaluator shall also perform the following tests:

- Test 1 [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.

#### 6.1.1.2 FCS_CKM.1 Cryptographic Key Generation (Data Encryption Key)

**FCS_CKM.1.1**

The TSF shall

- generate a DEK using the RBG as specified in FCS_RBG_EXT.1 (Appendix B)

that is 256 bits in length.

***PP Application Note***

*The purpose of this requirement is to explain DEK generation during provisioning.*

*If the TOE can be configured to obtain a DEK through more than one method, the ST Author chooses the applicable options within the selection. For example, the TOE may generate random numbers with an approved RBG to create a DEK, as well as provide an interface to accept a DEK from the environment.*

*If the ST Author chooses the first and/or third option in the selection the corresponding requirement is pulled from Appendix A and included in the body of the ST.*

**Assurance Activity**

**TSS**

The evaluator shall examine the TSS to determine that it describes how the TOE obtains a DEK (either generating the DEK or receiving from the environment).

If the TOE generates a DEK, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. If the DEK is generated outside of the TOE, the evaluator checks to ensure that for each platform identified in the TOE the TSS, it describes the interface used by the TOE to invoke this functionality. The evaluator uses the description of the interface between the RBG and the TOE to determine that it requests a key greater than or equal to the required key sizes.

**KMD**

If the TOE received the DEK from outside the host platform, then the evaluator shall examine the TSS to determine that the DEK is sent wrapped using the appropriate encryption algorithm. The evaluator shall verify that the KMD describes how the TOE unwraps the DEK.

**Test**

The evaluator shall perform the following test activities:

- The evaluator shall configure the TOE to ensure the functionality of all selections.

#### 6.1.1.3 FCS_CKM.1(c) Cryptographic Key Generation (Symmetric Keys)

**FCS_CKM.1.1(c)**

The TSF shall generate symmetric cryptographic keys using a Random Bit Generator as specified in FCS_RBG_EXT.1 and specified cryptographic key sizes 256 bit that meet the following: No Standard.

***PP Application Note***

*Symmetric keys may be used to generate keys along the key chain.*

#### 6.1.1.4 FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction

**FCS_CKM_EXT.4.1**

The TSF shall destroy all keys and keying material when no longer needed.

***PP Application Note***

*Keys, including intermediate keys and key material that are no longer needed are destroyed in volatile memory by using an approved method, FCS_CKM.4.1. Examples of keys are intermediate keys, submasks, and DEK. There may be instances where keys or key material that are contained in persistent storage are no longer needed and require destruction. Based on their implementation, vendors will explain when certain keys are no longer needed. There are multiple situations in which key material is no longer necessary, for example, a wrapped key may need to be destroyed when a password is changed. However, there are instances when keys are allowed to remain in memory, for example, a device identification key.*

**Assurance Activity**

**TSS**

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 shall verify the KMD includes a high level description of the areas where keys and key material resides 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.4 for the destruction.

#### 6.1.1.5 FCS_CKM.4 Cryptographic Key Destruction

**FCS_CKM.4.1**

The TSF shall erase cryptographic keys in accordance with a specified cryptographic key erasure method

- For volatile memory, the erasure shall be executed by a single direct overwrite consisting of zeroes following by a read-verify.
- For non-volatile storage, the erasure shall be executed by:
  - A single overwrite of key data storage location consisting of a static pattern, followed by a read-verify. If read-verification of the overwritten data fails, the process shall be repeated again;

that meets the following: no standard.

***PP Application Note***

*Keys, including intermediate keys and key material that are no longer needed are destroyed in volatile memory by using one of these approved methods. In these cases, the destruction method conforms to one of methods specified in this requirement. Cryptographic Erase is considered a well defined term for the destruction of key information. Some solutions support write access to media locations where keys are stored, thus allow for destruction of cryptographic keys via direct overwrites of key and key material data. In other cases storage virtualization techniques on system and/or device level could result in multiple copies of key data and/or the underlying media technology does not support direct overwrites of locations where key data are stored. Note that onetime programmable memories are excluded.*

**Assurance Activity**

**TSS**

The evaluator shall verify the TSS provides a high level description of how keys and key material are destroyed.

**KMD**

The evaluator shall check to ensure the KMD lists each type of key material its origin, possible temporary locations (e.g. key register, cache memory, stack, FIFO) and storage location. The evaluator shall verify that the KMD describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, etc.).

The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

The evaluator shall check to ensure the KMD lists each type of key material (software-based key storage, BEVs, passwords, etc.) and its origin, storage location, and the method for destruction for each key.

**Test**

For each software and firmware key clearing situation the evaluator shall repeat the following tests for Volatile Memory. For the test below, “key” refers to keys and key material.

These tests do not apply to hardware devices, such as Self Encrypting Drives.

- Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared correctly, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

  For each key subject to clearing, including copies of keys that are originally encrypted and stored in non-volatile memory by the TOE, the evaluator shall:

  1. Attach to the TOE software/firmware with a debugger.
  2. Record the value of the key in the TOE subject to clearing.
  3. Cause the TOE to perform a normal cryptographic processing with the key from #1.
  4. Cause the TOE to clear the key.
  5. Cause the TOE to stop the execution but not exit.
  6. Cause the TOE to dump the entire memory footprint of the TOE into a binary file.
  7. Search the content of the binary file created in #6 for instances of the known key value from #2.

  The test succeeds if no copies of the key from #4 are found in step #7 above and fails otherwise.

  The evaluator shall perform this test on all keys, including those persisted in encrypted form, to ensure intermediate copies are cleared.

**ST Application Note**

The Assurance Activities were updated as specified in Technical Query 150.

#### 6.1.1.6 FCS_COP.1(a) Cryptographic Operations (Signature Verification)

**FCS_COP.1.1(a)**

The TSF shall perform cryptographic signature services (verification) in accordance with a

- Elliptic Curve Digital Signature Algorithm with a key size of 256 bits or greater

that meets the following:

- FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" P-256, P-384, and no other curves; ISO/IEC 14888-3, Section 6.4, for ECDSA schemes.

***PP Application Note***

*The ST Author should choose the algorithm implemented to perform digital signatures. For the algorithm(s) chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.*

**Assurance Activity**

**TSS**

The evaluator shall check the TSS to ensure that it describes the overall flow of the signature verification. This should at least include identification of the format and general location (e.g., "firmware on the hard drive device" vice "memory location 0x00007A4B") of the data to be used in verifying the digital signature; how the data received from the operational environment are brought on to the device; and any processing that is performed that is not part of the digital signature algorithm (for instance, checking of certificate revocation lists).

**Test**

Each section below contains the tests the evaluators must perform for each type of digital signature scheme. Based on the assignments and selections in the requirement, the evaluators choose the specific activities that correspond to those selections.

It should be noted that for the schemes given below, there are no key generation/domain parameter generation testing requirements. This is because it is not anticipated that this functionality would be needed in the end device, since the functionality is limited to checking digital signatures in delivered updates. This means that the domain parameters should have already been generated and encapsulated in the hard drive firmware or on-board non-volatile storage. If key generation/domain parameter generation is required, the evaluation and validation scheme must be consulted to ensure the correct specification of the required assurance activities and any additional components.

The following tests are conditional based upon the selections made within the SFR.

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

ECDSA Algorithm Tests

**ECDSA FIPS 186-4 Signature Verification Test**

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

**Signature Verification Test**

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's authentic and unauthentic signatures. The evaluator shall inject errors in the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

#### 6.1.1.7 FCS_COP.1(b) Cryptographic Operation (Hash Algorithm)

**FCS_COP.1.1(b)**

The TSF shall perform cryptographic hashing services in accordance with SHA-256 that meet the following: ISO/IEC 10118-3:2004.

***PP Application Note***

*The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(a) (SHA 256 should be chosen for AES 128-bit keys, SHA 512 should be chosen for AES-256-bit keys). The selection of the standard is made based on the algorithms selected.*

**Assurance Activity**

**TSS**

The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

**Operational Guidance**

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

**Test**

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

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this cPP.

Short Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 99*i, where 1 ≤i ≤ m. For SHA-512, the length of the i-th message is 1024 + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. For SHA-512, the length of the i-th message is 1024 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

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

#### 6.1.1.8 FCS_COP.1(c) Cryptographic Operation (Keyed Hash Algorithm)

**FCS_COP.1.1(c)**

The TSF shall perform keyed-hash message authentication in accordance with HMAC-SHA-256 and cryptographic key sizes **32 to 32768 bits** that meet the following: ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2".

***PP Application Note***

*The key size [k] in the assignment falls into a range between L1 and L2 (defined in ISO/IEC 10118 for the appropriate hash function for example for SHA-256 L1 = 512, L2 =256) where L2 ≤ k ≤ L1.*

**Assurance Activity**

**TSS**

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

Test

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 using a known good implementation.

### 6.1.1.9 FCS_COP.1(d) Cryptographic Operation (Key Wrapping)

**FCS_COP.1.1(d)**

The TSF shall perform key wrapping in accordance with a specified cryptographic algorithm AES in the following modes KW and the cryptographic key size 256 bits that meet the following: ISO/IEC 18033-3 (AES), NIST SP 800-38F.

***PP Application Note***

*This requirement is used in the body of the ST if the ST Author chooses to use key wrapping in the key chaining approach that is specified in FCS_KYC_EXT.2 or as a means of accepting a wrapped DEK in FCS_CKM.1. For the purposes of this requirement, key wrapping consists of authenticated encryption and decryption.*

**Assurance Activity**

**TSS**

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

### 6.1.1.10 FCS_COP.1(f) Cryptographic Operation (AES Data Encryption/Decryption)

**FCS_COP.1.1(f)**

The TSF shall perform data encryption and decryption in accordance with a specified cryptographic algorithm AES used in XTS mode and cryptographic key sizes 256 bits that meet the following: AES as specified in ISO/IEC18033-3, XTS as specified in IEEE 1619.

***PP Application Note***

*This cPP allows for software encryption or hardware encryption. In software encryption, the TOE can provide the data encryption/decryption or the host platform could provide the encryption/decryption. Conversely, for hardware encryption, the encryption/decryption could be provided by a variety of mechanisms - dedicated hardware within a general purpose controller, the storage device's SOC, or a dedicated (co-)processor.*

*If XTS Mode is selected, a cryptographic key of 256-bit or of 512-bit is allowed as specified in IEEE 1619. XTS-AES key is divided into two AES keys of equal size -for example, AES-128 is used as the underlying algorithm, when 256-bit key and XTS mode are selected. AES-256 is used when a 512-bit key and XTS mode are selected.*

*The intent of this requirement is to specify the approved AES modes that the ST Author may select for AES encryption of the appropriate information on the hard disk. For the first selection, the ST author should indicate the mode or modes supported by the TOE implementation. The second selection indicates the key size to be used, which is identical to that specified for FCS_CKM.1(1). The third selection must agree with the mode or modes chosen in the first selection. If multiple modes are supported, it may be clearer in the ST if this component was iterated.*

**Assurance Activity**

**TSS**

The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for encryption.

**Guidance**

If multiple encryption modes are supported, the evaluator examines the guidance documentation to determine how a specific mode/key-size is chosen by the end user.

**Test**

The following tests are conditional based upon the selections made in the SFR.

**<u>AES-CBC Tests</u>**

**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 shallbe 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 Test**

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

**128 bit and 256 bit keys**

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

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

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

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.

**XTS-AES Test**

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.

#### 6.1.1.11 FCS_KYC_EXT.1 Key Chaining

**FCS_KYC_EXT.1.1**

The TSF shall maintain a key chain of: intermediate keys originating from one or more submask(s) to the BEV using the following method(s): key wrapping as specified in FCS_COP.1(d) while maintaining an effective strength of 256 bits.

**FCS_KYC_EXT.1.2**

The TSF shall provide a 256 bit BEV to the EE only after the TSF has successfully performed the validation process as specified in FCS_VAL_EXT.1.

***Application Note***

*Key Chaining is the method of using multiple layers of encryption keys to ultimately secure the BEV. The number of intermediate keys will vary – from one (e.g., taking the conditioned password authorization factor and directly using it as the BEV) to many. This applies to all keys that contribute to the ultimate wrapping or derivation of the BEV; including those in areas of protected storage (e.g. TPM stored keys, comparison values).*

*Multiple key chains to the BEV are allowed, as long as all chains meet the key chain requirement.*

*Once the ST Author has selected a method to create the chain (either by deriving keys or unwrapping them or encrypting keys or using RSA Key Transport), they pull the appropriate requirement out of Appendix B. It is allowable for an implementation to use for any or all methods.*

*For FCS_KYC_EXT.1.2, the validation process is defined in FCS_VAL_EXT.1, Appendix B. If that selection is made by the ST Author, then FCS_VAL_EXT.1 is pulled into the body of the ST.*

*The method the TOE uses to chain keys and manage/protect them is described in the Key Management Description; see Key Management Description for more information.*

**Assurance Activity**

**TSS**

The evaluator shall verify the TSS contains a high-level description of the BEV sizes – that it supports BEV outputs of no fewer 128 bits for products that support only AES-128, and no fewer than 256 bits for products that support AES-256.

**KMD**

The evaluator shall examine the KMD describes a high level description of the key hierarchy for all authorizations methods selected in FCS_AFA_EXT.1 that are used to protect the BEV. The evaluator shall examine the KMD to ensure it describes the key chain in detail. The description of the key chain shall be reviewed to ensure it maintains a chain of keys using key wrap or key derivation methods that meet FCS_COP.1(d) and FCS_KDF_EXT.1.

The evaluator shall examine the KMD to ensure that it describes how the key chain process functions, such that it does not expose any material that might compromise any key in the chain. (e.g. using a key directly as a compare value against a TPM) This description must include a diagram illustrating the key hierarchy implemented and detail where all keys and keying material is stored or what it is derived from. The evaluator shall examine the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or the initial authorization value and the effective strength of the BEV is maintained throughout the Key Chain.

The evaluator shall verify the KMD includes a description of the strength of keys throughout the key chain.

#### 6.1.1.12 FCS_KYC_EXT.2 Key Chaining

**FCS_KYC_EXT.2.1**

The TSF shall accept a BEV of <u>256 bits</u> from the AA.

**FCS_KYC_EXT.2.2**

The TSF shall maintain a chain of intermediary keys originating from the BEV to the DEK using the following method(s): <u>key wrapping as specified in FCS_COP.1(d)</u> while maintaining an effective strength of <u>256 bits</u>.

***PP Application Note***

*Key Chaining is the method of using multiple layers of encryption keys to ultimately secure the protected data encrypted on the drive. The number of intermediate keys will vary –from two (e.g., using the BEV as an intermediary key to wrap the DEK to many). This applies to all keys that contribute to the ultimate wrapping or derivation of the DEK; including those in areas of protected storage (e.g. TPM stored keys, comparison values).*

*Once the ST Author has selected a method to create the chain (either by deriving keys or unwrapping them), they pull the appropriate requirement out of Appendix B. It is allowable for an implementation to use both methods.*

*The method the TOE uses to chain keys and manage/protect them is described in the Key Management Description; see Key Management Description for more information.*

**Assurance Activity**

**KMD**

The evaluator shall examine the KMD describes a high level description of the key hierarchy for all authorizations methods selected in FCS_AFA_EXT.1 that are used to protect the BEV. The evaluator shall examine the KMD to ensure it describes the key chain in detail. The description of the key chain shall be reviewed to ensure it maintains a chain of keys using key wrap or key derivation methods that meet FCS_KDF_EXT.1, FCS_COP.1(d), FCS_COP.1(e), FCS_COP.1(g).

The evaluator shall examine the KMD to ensure that it describes how the key chain process functions, such that it does not expose any material that might compromise any key in the chain. (e.g. using a key directly as a compare value against a TPM) This description must include a diagram illustrating the key hierarchy implemented and detail where all keys and keying material is stored or what it is derived from. The evaluator shall examine the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or knowledge of the BEV and the effective strength of the DEK is maintained throughout the Key Chain.

The evaluator shall verify the KMD includes a description of the strength of keys throughout the key chain.

#### 6.1.1.13 FCS_PCC_EXT.1 Cryptographic Password Construct and Conditioning

**FCS_PCC_EXT.1.1**

A password used to generate a password authorization factor shall enable up to **4096** characters in the set of {upper case characters, lower case characters, numbers, and **all other 8 bit values**} and shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC-SHA-256, with **1500** iterations, and output cryptographic key sizes 256 that meet the following: NIST SP 800-132.

***PP Application Note***

*The password is represented on the host machine as a sequence of characters whose encoding depends on the TOE and the underlying OS. This sequence must be conditioned into a string of bits that forms the submask to be used as input into the key chain. Conditioning can be performed using one of the identified hash functions or the process described in NIST SP 800-132; the method used is selected by the ST Author. If 800-132 conditioning is specified, then the ST author fills in the number of iterations that are performed. 800-132 also requires the use of a pseudo-random function (PRF) consisting of HMAC with an approved hash function. The ST author selects the hash function used, also includes the appropriate requirements for HMAC.*

**Assurance Activity**

**TSS**

The evaluator shall ensure the TSS describes the manner in which the TOE enforces the construction of passwords, including the length, and requirements on characters (number and type). The TSS also provides a description of how the password is conditioned and the evaluator ensures it satisfies the requirement.

**KMD**

The evaluator shall examine the KMD to ensure that the formation of the BEV and intermediary keys is described and that the key sizes match that selected by the ST Author.

The evaluator shall check that the KMD 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 KMD 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 BEV as specified above.

**Test**

The evaluator shall also perform the following tests:

- Test 1: Ensure that the TOE supports passwords/passphrases of a minimum length of 64 characters.
- Test 2: If the TOE supports a password/passphrase length up to a maximum number of characters, n (which would be greater than 64), then ensure that the TOE will not accept more than n characters.
- Test 3: Ensure that the TOE supports passwords consisting of all characters assigned and supported by the ST author.

#### 6.1.1.14 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

**FCS_RBG_EXT.1.1**

The TSF shall perform all deterministic random bit generation services in accordance with NIST SP 800-90A using CTR_DRBG (AES).

**FCS_RBG_EXT.1.2**

The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from ***a RNG provided by the host platform*** with a minimum of 256 bits of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 “Security Strength Table for Hash Functions”, of the keys and hashes that it will generate.

***ST Application Note***

*The ST Author refined the SFR by deleting the selection for specifying the number of software and/or hardware based noise sources. The TRRT approved seeding the RBG from an RNG provided by the host platform, because all SFRs requiring the use of random numbers allow for the use of a RBG implemented by the TSF or a RNG provided by the host platform.*

***PP Application Note***

*ISO/IEC 18031:2011 contains different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used and include the specific underlying cryptographic primitives used in the requirement. While any of the identified hash functions (SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed. Table C.2 in ISO/IEC 18031:2011 provides an identification of Security strengths, Entropy and Seed length requirements for the AES-128 and 256 Block Cipher.*

*The CTR_DRGB in ISO/IEC 18031:2011 requires using derivation function, whereas NIST SP 800-90A does not. Either model is acceptable. In the first selection in FCS_RBG_EXT.1.1, the ST Author choses the standard they are compliant.*

*The first selection in FCS_RBG_EXT.1.2 the ST author fills in how many entropy sources are used for each type of entropy source they employ. It should be noted that a combination of hardware and software based noise sources is acceptable.*

*It should be noted that the entropy source is considered to be a part of the RBG and if the RBG is included in the TOE, the developer is required to provide the entropy description outlined in Appendix D. The documentation *and tests* required in the Evaluation Activity for this element necessarily cover each source indicated in FCS_RBG_EXT.1.2.*

**Assurance Activity**

**TSS**

For any RBG services provided by a third party, the evaluator shall ensure the TSS includes a statement about the expected amount of entropy received from such a source, and a full description of the processing of the output of the third-party source. The evaluator shall verify that this statement is consistent with the selection made in FCS_RBG_EXT.1.2 for the seeding of the DRBG. If the ST specifies more than one DRBG, the evaluator shall examine the TSS to verify that it identifies the usage of each DRBG mechanism.

**Entropy Documentation**

The evaluator shall ensure the Entropy Essay provides all of the required information as described in Appendix D of the cPP. The evaluator assesses the information provided and ensures the TOE is providing sufficient entropy when it is generating a Random Bit String.

**Operational Guidance**

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected DRBG mechanism(s), if necessary, and provides information regarding how to instantiate/call the DRBG for RBG services needed in this cPP.

**Test**

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable by the TOE, the evaluator shall perform 15 trials for each configuration. The evaluator shall verify that the instructions in the operational guidance for configuration of the RNG are valid.

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

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

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

**Entropy input**: the length of the entropy input value must equal the seed length.

**Nonce:** If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

**Personalization string**: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

**Additional input**: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

#### 6.1.1.15 FCS_SMV_EXT.1 Validation

**FCS_SMV_EXT.1.1**

The TSF shall validate a BEV using the following methods: <u>key wrap as specified in FCS_COP.1(d)</u>.

**FCS_SMV_EXT.1.2**

The TSF shall <u>block validation after a **10** of consecutive failed validation attempts</u>.

***PP Application Note***

*"Validation" of the BEV can occur at any point in the key chain, including when the DEK is decrypted. For the purposes of this requirement, validating a key derived from the BEV equates to "validating" the BEV. The purpose of performing secure validation is to not expose any material that might compromise the submask(s).*

*The TOE validates the BEV prior to allowing the user access to the data stored on the drive. When the key wrap in FCS_COP.1(d) is used, the validation is performed inherently.*

*The delay must be enforced by the TOE, but this requirement is not intended to address attacks that bypass the product (e.g. attacker obtains hash value or "known" crypto value and mounts attacks outside of the TOE, such as a third party password crackers). The cryptographic functions (i.e., hash, decryption) performed are those specified in FCS_COP.1(b) and FCS_COP.1(f).*

**Assurance Activity**

**TSS**

The evaluator shall examine the TSS to determine which authorization factors support validation.

The evaluator shall examine the TSS to review a high-level description if multiple submasks are used within the TOE, how the submasks are validated (e.g., each submask validated before combining, once combined validation takes place).

**KMD**

The evaluator shall examine the KMD to verify that it described the method the TOE employs to limit the number of consecutively failed authorization attempts.

The evaluator shall examine the vendor's KMD to ensure it describes how validation is performed. The description of the validation process in the KMD provides detailed information how the TOE validates the BEV.

The KMD describes how the process works, such that it does not expose any material that might compromise the submask(s).

**Operational Guidance**

[conditional] If 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.

**Test**

The evaluator shall perform the following tests:

- Test 1: The evaluator shall determine the limit on the average rate of the number of consecutive failed authorization attempts. The evaluator will test the TOE by entering that number of incorrect authorization factors in consecutive attempts to access user data. If the limit mechanism includes any "lockout" period, the time period tested should include at least one such period. Then the evaluator will verify that the TOE behaves as described in the TSS.

#### 6.1.1.16 FCS_SNI_EXT.1 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)

**FCS_SNI_EXT.1.1**

The TSF shall only use salts that are generated by a RNG as specified in FCS_RBG_EXT.1.

**FCS_SNI_EXT.1.2**

The TSF shall only use unique nonces with a minimum size of 64 bits.

**FCS_SNI_EXT.1.3**

The TSF shall create IVs in the following manner:

- CBC: IVs shall be non-repeating,
- CCM: Nonce shall be non-repeating,
- XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer,
- GCM: IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^32 for a given secret key.

***PP Application Note***

*This requirement covers several important factors –the salt must be random, but the nonces only have to be unique. FCS_SNI_EXT.1.3 specifies how the IV should be handled for each encryption mode. Assigned consecutively could mean using a one-up counter. Additionally, nonce is called as Starting Variable (SV) in ISO/IEC 19772.*

*Tweak values shall be non-negative numbers, starting at an arbitrary non-negative number, and all subsequent tweak values shall be incremented from the initial value.*

**Assurance Activity**

**TSS**

The evaluator shall ensure the TSS describes how salts are generated. The evaluator shall confirm that the salt is generating using an RBG described in FCS_RBG_EXT.1 or by the Operational Environment. If external function is used for this purpose, the TSS should include the specific API that is called with inputs.

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.

### 6.1.2 Class FDP: User Data Protection

#### 6.1.2.1 FDP_DSK_EXT.1 Extended: Protection of Data on Disk

**FDP_DSK_EXT.1.1**

The TSF shall perform Full Drive Encryption in accordance with FCS_COP.1(f), such that the drive contains no plaintext protected data.

**FDP_DSK_EXT.1.2**

The TSF shall encrypt all protected data without user intervention.

***PP Application Note***

*The intent of this requirement is to specify that encryption of any protected data will not depend on a user electing to protect that data. The drive encryption specified in FDP_DSK_EXT.1 occurs transparently to the user and the decision to protect the data is outside the discretion of the user, which is a characteristic that distinguishes it from file encryption. The definition of protected data can be found in the glossary.*

*The cryptographic functions that perform the encryption/decryption of the data may be provided by the environment. If the TOE provides the cryptographic functions to encrypt/decrypt the data, the ST Author pulls FCS_COP.1(f) from the Appendix A and includes it in the main body of the ST.*

**Assurance Activity**

**TSS**

The evaluator shall examine the TSS to ensure that the description is comprehensive in how the data is written to the disk and the point at which the encryption function is applied. The TSS must make the case that standard methods of accessing the disk drive via the host platforms operating system will pass through these functions.

For the cryptographic functions that are provided by the Operational Environment, the evaluator shall check the TSS to ensure it describes--for each platform identified in the ST--the interface(s) used by the TOE to invoke this functionality.

The evaluator shall verify the TSS in performing the evaluation activities for this requirement. The evaluator shall ensure the comprehensiveness of the description, confirms how the TOE writes the data to the disk drive, and the point at which it applies the encryption function.

The evaluator shall verify that the TSS describes the initialization of the TOE and the activities the TOE performs to ensure that it encrypts all the storage devices entirely when a user or administrator first provisions the TOE. The evaluator shall verify the TSS describes areas of the disk that it does not encrypt (e.g., portions associated with the Master Boot Records (MBRs), boot loaders, partition tables, etc.). If the TOE supports multiple disk encryptions, the evaluator shall examine the administration guidance to ensure the initialization procedure encrypts all storage devices on the platform.

**Operational Guidance**

The evaluator shall review the AGD guidance to determine that it describes the initial steps needed to enable the FDE function, including any necessary preparatory steps. The guidance shall provide instructions that are sufficient, on all platforms, to ensure that all hard drive devices will be encrypted when encryption is enabled.

**Test**

The evaluator shall verify the KMD includes a description of the data encryption engine, its components, and details about its implementation (e.g. for hardware: integrated within the device's main SOC or separate co-processor, for software: initialization of the product, drivers, libraries (if applicable), logical interfaces for encryption/decryption, and areas which are not encrypted (e.g. boot loaders, portions associated with the Master Boot Record (MBRs), partition tables, etc.)). The evaluator shall verify the KMD provides a functional (block) diagram showing the main components (such as memories and processors) and the data path between, for hardware, the device's host interface and the device's persistent media storing the data, or for software, the initial steps needed to the activities the TOE performs to ensure it encrypts the storage device entirely when a user or administrator first provisions the product. The hardware encryption diagram shall show the location of the data encryption engine within the data path. The evaluator shall validate that the hardware encryption diagram contains enough detail showing the main components within the data path and that it clearly identifies the data encryption engine.

The evaluator shall verify the KMD provides sufficient instructions for all platforms to ensure that when the user enables encryption, the product encrypts all hard storage devices. The evaluator shall verify that the KMD describes the data flow from the device's host interface to the device's persistent media storing the data. The evaluator shall verify that the KMD provides information on those conditions in which the data bypasses the data encryption engine (e.g. read-write operations to an unencrypted Master Boot Record area).

The evaluator shall verify that the KMD provides a description of the platform's boot initialization, the encryption initialization process, and at what moment the product enables the encryption. The evaluator shall validate that the product does not allow for the transfer of user data before it fully initializes the encryption. The evaluator shall ensure the software developer provides special tools which allow inspection of the encrypted drive either in-band or out-of-band, and may allow provisioning with a known key.

The evaluator shall perform the following tests:

1. Write data to random locations, perform required actions and compare:

- Ensure TOE is initialized and, if hardware, encryption engine is ready;
    - Provision TOE to encrypt the storage device. For SW Encryption products, or hybrid products use a known key and the developer tools.

  - Determine a random character pattern of at least 64 KB;
  - Retrieve information on what the device TOE's lowest and highest logical address is for which encryption is enabled;
- Write pattern to storage device in multiple locations:
        - For HW Encryption, randomly select several logical address locations within the device's lowest to highest address range and write pattern to those addresses;
    - For SW Encryption, write the pattern using multiple files in multiple logical locations.
- Verify data is encrypted:
        - For HW Encryption,
    - Engage device's functionality for generating a new encryption key, thus performing an erase of the key per FCS_CKM.4;
    - Read from the same locations at which the data was written;
    - Compare the retrieved data to the written data and ensure they do not match
- For SW Encryption, using developer tools;
        - Review the encrypted storage device for the plaintext pattern at each location where the file was written and confirm plaintext pattern cannot be found.
        - Using the known key, verify that each location where the file was written, the plaintext pattern can be correctly decrypted using the key.
        - If available in the developer tools, verify there are no plaintext files present in the encrypted range

### 6.1.3 Class FMT: Security Management

#### 6.1.3.1 FMT_SMF.1 Specification of Management Functions

**FMT_SMF.1.1**

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

a) Change the DEK, as specified in FCS_CKM.1, when reprovisioning or when commanded,
b) cryptographically erase the DEK,
c) allowing authorized users to change authorization factors or set of authorization factors used,
d) initiate TOE firmware/software updates,
e) <u>no other functions</u>.

***ST Application Note***

*The ST author combined the EE and AA versions of FPT_KYP_EXT.1.1 and associated Assurance Activities according to "FDE Paper compliant ST-v0.3".*

***PP Application Note***

*The intent of this requirement is to express the management capabilities that the TOE possesses. This means that the TOE must be able to perform the listed functions. Item (e) is used to specify functionality that may be included in the TOE, but is not required to conform to the cPP. Configure cryptographic functionality could include key management functions, for example, the BEV will be wrapped or encrypted, and the EE will need to unwrap or decrypt the BEV. In item e, if no other management functions are provided (or claimed), then "no other functions" should be selected.*

*Changing the DEK would require the data to be re-encrypted with the new DEK, but allows the user the ability to generate new DEKs.*

*For the purposes of this document, key sanitization means to destroy the DEK, using one of the approved destruction methods. In some implementations, changing the DEK could be the same functionality as cryptographically erasing the DEK.*

**Assurance Activity**

**TSS**

Option A: The evaluator shall ensure the TSS describes how the TOE changes the DEK.

Option B: The evaluator shall ensure the TSS describes how the TOE cryptographically erases the DEK.

Option C: The evaluator shall ensure the TSS describes the methods by which users may change the set of all authorization factor values supported

Option D: The evaluator shall ensure the TSS describes the process to initiate TOE firmware/software updates.

Option E: If additional management functions are claimed in the ST, the evaluator shall ensure the TSS describes the additional functions.

**KMD**

Option E: If the TOE offers the functionality to import an encrypted DEK, the evaluator shall ensure the KMD describes how the TOE imports a wrapped DEK and performs the decryption of the wrapped DEK.

**Operational Guidance**

Option A: The evaluator shall review the AGD guidance and shall determine that the instructions for changing a DEK exist. The instructions must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate or re-generate the DEK.

Option C: The evaluator shall examine the operational guidance to ensure that it describes how selected authorization factor values are changed.

Option D: The evaluator shall examine the operational guidance to ensure that it describes how to initiate TOE firmware/software updates.

Option E: Default Authorization Factors: It may be the case that the TOE arrives with default authorization factors in place. If it does, then the selection in section E must be made so that there is a mechanism to change these authorization factors. The operational guidance shall describe the method by which the user changes these factors when they are taking ownership of the device. The TSS shall describe the default authorization factors that exist.

Disable Key Recovery: The guidance for disabling this capability shall be described in the AGD documentation.

**Test**

Option A and B: The evaluator shall verify that the TOE has the functionality to change and cryptographically erase the DEK (effectively removing the ability to retrieve previous user data).

Option C: The evaluator shall initialize the TOE such that it requires the user to input an authorization factor in order to access encrypted data.

- Test 1: The evaluator shall first provision user authorization factors, and then verify all authorization values supported allow the user access to the encrypted data. Then the evaluator shall exercise the management functions to change a user's authorization factor values to a new one. Then he or she will verify that the TOE denies access to the user's encrypted data when he or she uses the old or original authorization factor values to gain access.

Option D: The evaluator shall verify that the TOE has the functionality to initiate TOE firmware/software updates.

Option E: If additional management functions are claimed, the evaluator shall verify that the additional features function as described.

- Test 2: [conditional] If the TOE provides default authorization factors, the evaluator shall change these factors in the course of taking ownership of the device as described in the operational guidance. The evaluator shall then confirm that the (old) authorization factors are no longer valid for data access.
- Test 3 [conditional] If the TOE provides key recovery capability whose effects are visible at the TOE interface, then the evaluator shall devise a test that ensures that the key recovery capability has been or can be disabled following the guidance provided by the vendor.

### 6.1.4 Class FPT: Protection of the TSF

#### 6.1.4.1 FPT_KYP_EXT.1 Extended: Protection of Key and Key Material

**FPT_KYP_EXT.1.1**

The TSF shall only store keys in non-volatile memory when wrapped, as specified in FCS_COP.1(d) or encrypted, as specified in FCS_COP.1(g) or FCS_COP.1(e) unless the key meets any one of following criteria:

- <u>The plaintext key is not part of the key chain as specified in FCS_KYC_EXT.1 or FCS_KYC_EXT.2.</u>

***ST Application Note***

*The TSF does not store any plaintext key material in non-volatile memory; however, the cPP requires the selection of one of the caveats.*

*The ST author combined the EE and AA versions of FPT_KYP_EXT.1.1 and associated Assurance Activities according to "FDE Paper compliant ST-v0.3".*

***PP Application Note***

*The plaintext key storage in non-volatile memory is allowed for several reasons. If the keys exist within protected memory that is not user accessible on the TOE or OE, the only methods that allow it to play a security relevant role for protecting the BEV or the DEK is if it is a key split or providing additional layers of wrapping or encryption on keys that have already been protected.*

*When stored in non-volatile memory (even in protected storage), the DEK is always encrypted (wrapped) and only exists in plaintext form in volatile memory, when it is being used to encrypt or decrypt data. Provisioning keys may exist in plaintext form in non-volatile memory before provisioning by the drive owner.*

*If the TOE does not store keys in non-volatile memory, a statement in the TSS stating that keys are never stored in non-volatile memory is all that is required and no evaluation activity needs to be performed.*

*This requirement is addressing the keys related to the encryption of user data –specifically keys from within the key chain.*

***ST Application Note***

*The TSF does not store any plaintext key material in non-volatile memory; however, the cPP requires the selection of one of the caveats.*

**Assurance Activity**

**KMD**

The evaluator shall verify the KMD for a high level description of method used to protect keys stored in non-volatile memory.

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(d) or FCS_COP.1(g) 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.

#### 6.1.4.2 FPT_TUD_EXT.1 Trusted Update

**FPT_TUD_EXT.1.1**

The TSF shall provide authorized users the ability to query the current version of the TOE software/firmware.

**FPT_TUD_EXT.1.2**

The TSF shall provide authorized users the ability to initiate updates to TOE software/firmware.

**FPT_TUD_EXT.1.3**

The TSF shall verify updates to the TOE software/firmware using a digital signature by the manufacturer prior to installing those updates.

***PP Application Note***

*The digital signature mechanism referenced in the third element is the one specified in FCS_COP.1(a) in Appendix A. While this component requires the TOE to implement the update functionality itself, it is acceptable to perform the cryptographic checks using functionality available in the Operational Environment.*

**Assurance Activity**

**TSS**

The evaluator shall examine the TSS to ensure that it describes information stating that an authorized source signs TOE updates and will have an associated digital signature. The evaluator shall examine the TSS contains a definition of an authorized source along with a description of how the TOE uses public keys for the update verification mechanism in the Operational Environment. The evaluator ensures the TSS contains details on the protection and maintenance of the TOE update credentials.

If the Operational Environment performs the signature verification, then the evaluator shall examine the TSS to ensure it describes --for each platform identified in the ST--the interface(s) used by the TOE to invoke this cryptographic functionality.

**Operational Guidance**

The evaluator ensures that the Operational Guidance describes how the vendor provides updates for the TOE; the processing associated with verifying the digital signature of the updates (as defined in FCS_COP.1(a)); and the actions that take place for successful and unsuccessful cases.

**Test**

The evaluators shall perform the following tests (if the TOE supports multiple signature each using a different hash algorithm, then the evaluator performs tests 2 and 3 for different combinations of authentic and unauthentic digital signatures and hashes, as well as for digital signature alone):

Test 1: The evaluator performs the version verification activity to determine the current version of the TOE. After the update tests described in the following tests, the evaluator performs this activity again to verify that the version correctly corresponds to that of the update.

Test 2: The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that an update successfully installs on the TOE. The evaluator shall perform a subset of other assurance activity tests to demonstrate that the update functions as expected. FPT_TST_EXT.1 TSF Testing.

### 6.1.4.3 FPT_TST_EXT.1 Extended: TSF Testing

**FPT_TST_EXT.1.1**

The TSF shall run a suite of the following self-tests <u>during initial start-up (on power on)</u> to demonstrate the correct operation of the TSF.

***PP Application Note***

*The tests regarding cryptographic functions implemented in the TOE can be deferred, as long as the tests are performed before the function is invoked.*

*If FCS_RBG_EXT.1 is implemented by the TOE and according to NIST SP 800-90, the evaluator shall verify that the TSS describes health tests that are consistent with section 11.3 of NIST SP 800-90.*

*If any FCS_COP functions are implemented by the TOE, the TSS shall describe the known-answer self-tests for those functions.*

*The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TSF, the method by which those functions are tested. The TSS will describe, for each of these functions, the method by which correct operation of the function/component is verified. The evaluator shall determine that all of the identified functions/components are adequately tested on start-up.*

**Assurance Activity**

**TSS**

The evaluator shall verify that the TSS describes the known-answer self-tests for cryptographic functions.

The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TOE and the method by which the TOE tests those functions. The evaluator shall verify that the TSS includes each, for each of these functions, the method by which the TOE verifies the correct operation of the function. The evaluator shall verify that the TSF data are appropriate for TSF Testing. For example, more than blocks are tested for AES in CBC mode, output of AES in GCM mode is tested without truncation, or 512-bit key is used for testing HMAC-SHA-512.

If FCS_RBG_EXT.1 is implemented by the TOE and according to NIST SP 800-90, the evaluator shall verify that the TSS describes health tests that are consistent with section 11.3 of NIST SP 800-90.

If any FCS_COP functions are implemented by the TOE, the TSS shall describe the known-answer self-tests for those functions.

The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TSF, the method by which those functions are tested. The TSS will describe, for each of these functions, the method by which correct operation of the function/component is verified. The evaluator shall determine that all of the identified functions/components are adequately tested on start-up.

## 6.2 Security Assurance Requirements

This Security Target is conformant with the assurance requirements specified in the PP.

Table 5: Assurance Requirements

| Assurance Class | Assurance Component |
|---|---|
| Security Target (ASE) | Conformance claims (ASE_CCL.1) |
| | Extended components definition (ASE_ECD.1) |
| | ST introduction (ASE_INT.1) |
| | Security objectives for the operational environment (ASE_OBJ.1) |
| | Stated security requirements (ASE_REQ.1) |
| | Security Problem Definition (ASE_SPD.1) |
| | TOE summary specification (ASE_TSS.1) |
| Development (ADV) | Basic functional specification (ADV_FSP.1) |
| Guidance documents (AGD) | Operational user guidance (AGD_OPE.1) |
| | Preparative procedures (AGD_PRE.1) |
| Life cycle support (ALC) | Labeling of the TOE (ALC_CMC.1) |
| | TOE CM coverage (ALC_CMS.1) |
| Tests (ATE) | Independent testing –sample (ATE_IND.1) |
| Vulnerability assessment (AVA) | Vulnerability survey (AVA_VAN.1) |

### 6.2.1 Extended Security Assurance Requirements

These requirements are taken directly from the cPP.

#### 6.2.1.1 Class ASE: Security Target

The ST is evaluated as per ASE activities defined in the CEM. In addition, there may be Evaluation Activities specified within the SD that call for necessary descriptions to be included in the TSS that are specific to the TOE technology type.

The SFRs in this cPP allow for conformant implementations to incorporate a wide range of acceptable key management approaches as long as basic principles are satisfied. Given the criticality of the key management scheme, this cPP requires the developer to provide a detailed description of their key management implementation. This information can be submitted as an appendix to the ST and marked proprietary, as this level of detailed information is not expected to be made publicly available. See Appendix E for details on the expectation of the developer's Key Management Description.

In addition, if the TOE includes a random bit generator Appendix D provides a description of the information expected to be provided regarding the quality of the entropy.

The TOE summary specification shall describe how the TOE meets each SFR, including a proprietary Key Management Description (Appendix E), and Entropy Essay.

#### 6.2.1.1.1 Conformance Claims (ASE_CCL.1)

The table below indicates the actions to be taken for particular ASE_CCL.1 elements in order to determine exact compliance with a cPP.

Table 6: Conformance Claims

| ASE_CCL.1 Element | Evaluator Action |
|---|---|
| ASE_CCL.1.8C | The evaluator shall check that the statements of security problem definition in the PP and ST are identical. |
| ASE_CCL.1.9C | The evaluator shall check that the statements of security objectives in the PP and ST are identical. |
| ASE_CCL.1.10C | The evaluator shall check that the statements of security requirements in the ST include all the mandatory SFRs in the cPP, and all of the selection-based SFRs that are entailed by selections made in other SFRs (including any SFR iterations added in the ST). The evaluator shall check that if any other SFRs are present in the ST (apart from iterations of SFRs in the cPP) then these are taken only from the list of optional SFRs specified in the cPP (the cPP will not necessarily include optional SFRs, but may do so). If optional SFRs from the cPP are included in the ST then the evaluator shall check that any selection-based SFRs entailed by the optional SFRs adopted are also included in the ST. |

### 6.2.1.2 Class ADV: Development

The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any additional information required by this cPP that is not to be made public (e.g., Entropy Essay).

#### 6.2.1.2.1 Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TOE Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this cPP will necessarily have interfaces to the Operational Environment that are not directly invokable by TOE users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional “functional specification” documentation is necessary to satisfy the Evaluation Activities specified in the SD.

The Evaluation Activities in the SD are associated with the applicable SFRs; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

**Evaluation Activity:**

The evaluator shall check the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g., perform updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent, is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

**Evaluation Activity:**

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any supplementary information required by the cPP for aspects such as entropy analysis or cryptographic key management architecture[1]: no additional "functional specification" documentation is necessary to satisfy the Evaluation Activities. The interfaces that need to be evaluated are also identified by reference to the assurance activities listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any supplementary information required by the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the Evaluation Activities for each SFR also means that the tracing required in ADV_FSP.1.2D is treated as implicit, and no separate mapping information is required for this element.

However, if the evaluator is unable to perform some other required Evaluation Activity because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

#### 6.2.1.3 Class AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verify that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel.

Guidance must be provided for every operational environment that the product supports as claimed in the ST. For hardware products, the developer may not be aware of all the platforms an integrator choses to use to deliver the product. A description of the commands the integrator would need to issue to properly configure the TOE (i.e., satisfies the SFRs in the ST) would satisfy the intent of "every operational environment the product supports". This guidance includes:

- instructions to successfully install the TSF in that environment; and
- instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
- instructions to provide a protected administrative capability.

Guidance pertaining to particular security functionality must also be provided; requirements on such guidance are contained in the Evaluation Activities specified in the SD.

##### 6.2.1.3.1 Operational User Guidance (AGD_OPE.1)

The operational user guidance does not have to be contained in a single document. Guidance to users, administrators, application developers and integrators can be spread among documents or web pages.

[1] The Security Target and AGD documentation are public documents. Supplementary information may be public or proprietary: the cPP and/or Evaluation Activity descriptions will identify where such supplementary documentation is permitted to be proprietary and non-public.

The developer should review the Evaluation Activities contained in the SD to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

**Evaluation Activity:**

The evaluator shall check the requirements below are met by the operational guidance. It should be noted that operational guidance may take the form of an "integrator's guide", where the TOE developer provides a description of the interface (e.g., commands that the Host Platform may invoke to configure a SED).

Operational guidance documentation shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

Operational guidance must be provided for every Operational Environment that the TOE supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target. This may be contained all in one document.

The contents of the operational guidance will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

a) The operational guidance shall contain instructions for configuring any 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.
b) The operational guidance shall describe how to configure the IT environments that are supported to shut down after an administratively defined period of inactivity.
c) The operational guidance shall identify system "sleeping" states for all supported operating environments and for each environment, provide administrative guidance on how to disable the sleep state. As stated above, the TOE developer may be providing an integrator's guide and "power states" may be an abstraction that SEDs provide at various levels –e.g., may simply provide a command that the Host Platform issues to manage the state of the device, and the Host Platform is responsible for providing a more sophisticated power management scheme.
d) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The operational guidance shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

#### 6.2.1.3.2 Preparative Procedures (AGD_PRE.1)

As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

**Evaluation Activity:**

The evaluator shall check the requirements below are met by the preparative procedures.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

Preparative procedures shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

Preparative procedures must include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target). The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE itself).

Preparative procedures must be provided for every Operational Environment that the TOE supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target. This may be contained all in one document.

The preparative procedures must include

a) instructions to successfully install the TSF in each Operational Environment; and
b) instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
c) instructions to provide a protected administrative capability.

#### 6.2.1.4 Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this cPP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor’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 is a reflection on the information to be made available for evaluation at this assurance level.

##### 6.2.1.4.1 Labeling of the TOE (ALC_CMC.1)

This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a “hard label” (e.g., stamped into the metal, paper label) or a “soft label” (e.g., electronically presented when queried). The evaluator performs the CEM work units associated with ALC_CMC.1

##### 6.2.1.4.2 TOE CM Coverage (ALC_CMS.1)

Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

#### 6.2.1.5 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. For this cPP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

##### 6.2.1.5.1 Independent Testing – Conformance (ATE_IND.1)

Testing is performed to confirm the functionality described in the TSS as well as the operational guidance (includes “evaluated configuration” instructions). The focus of the testing is to confirm that the requirements specified in Section 5 are being met. The Evaluation Activities in the SD identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

**Evaluation Activity:**

The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

**Evaluation Activity:**

The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

**Evaluation Activity:**

The evaluator shall prepare a test plan that covers all of the testing actions for ATE_IND.1 in the CEM and in the SFR-related Evaluation Activities. While it is not necessary to have one test case per test listed in an Evaluation Activity, the evaluator must show in the test plan that each applicable testing requirement in the SFR-related Evaluation Activities is covered.

The test plan identifies the platforms to be tested, and for any 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 and configuration of each platform to be tested, and any setup actions that are 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 any cryptographic engine to be used (e.g. for cryptographic protocols being evaluated).

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives, and the expected results.

The test report (which could just be an updated 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, so that a fix was then installed and then a successful re-run of the test was carried out, then the report would show a “fail” result followed by a “pass” result (and the supporting details), and not just the “pass” result[2].

[2] It is not necessary to capture failures that were due to errors on the part of the tester or test environment. The intention here is to make absolutely clear when a planned test resulted in a change being required to the originally specified test configuration in the test plan, to the evaluated configuration identified in the ST and operational guidance, or to the TOE itself.

#### 6.2.1.6 Class AVA: Vulnerability Assessment

For the first generation of this cPP, the iTC is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products and provide that content into the AVA_VAN discussion. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. This information will be used in the development of future protection profiles.

##### 6.2.1.6.1 Vulnerability Survey (AVA_VAN.1)

Appendix A in the companion Supporting Document provides a guide to the evaluator in performing a vulnerability analysis.

**Evaluation Activity:**

The evaluator shall document their analysis and testing of potential vulnerabilities with respect to this requirement. This report could be included as part of the test report for ATE_IND, or could be a separate document.

The evaluator performs a search of public information to determine the vulnerabilities that have been found in products representing the relevant TOE type (including vulnerabilities related to aspects such as components used in the TOE and the communication protocols that it uses) as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided for ATE_IND) to confirm the vulnerability, if suitable.

See Appendix A for more information on vulnerability assessment.

# 7 TOE Summary Specification

This section provides evaluators and potential consumers of the TOE with a high-level description of each SFR, thereby enabling them to gain a general understanding of how the TOE is implemented. These descriptions are intentionally not overly detailed, thereby disclosing no proprietary information. These sections refer to SFRs defined in Section 6, Security Requirements.

The TOE consists of the following Security Functions:

- Cryptographic Support
- User Data Protection
- Security Management
- Protection of the TSF

## 7.1 Cryptographic Support

### 7.1.1 Cryptographic Algorithms

The TSF implements the following cryptographic algorithms:

Table 7: Cryptographic Algorithms

| SFR | Description | Library/Implementation | CAVP Cert. |
|---|---|---|---|
| FCS_COP.1(a) | ECDSA P-256 with SHA-256 Signature Verification (POST FW Integrity Check) | Secure Boot Loader ECDSA/SHA Version 1747 | 990 |
| FCS_COP.1(a) | ECDSA P-256 with SHA-256 Signature Verification (Trusted Update: FW Authentication)<br>ECDSA P-384 with SHA-256 Signature Verification (Trusted Update: FW Authentication Certificate Verification) | OpenSSL ECDSA/SHA Version 1.0.1r | 991 |
| FCS_COP.1(b) | SHA-256 Hashing (POST FW Integrity Check) | Secure Boot Loader ECDSA/SHA Version 1747 | 3487 |
| FCS_COP.1(b) | SHA-256 Hashing (Trusted Update) | OpenSSL ECDSA/SHA Version 1.0.1r | 3488 |
| FCS_COP.1(b) | SHA-256 Hashing (HMAC Primitive) | OpenSSL-FIPS Version d40d43f6e6f28cdaf549e0fbc0f3d4a45b003e10 | 3489 |
| FCS_COP.1(c) | HMAC-SHA-256 (PBKDF: 4-4096 byte key, 512-bit block, 256-bit MAC) | OpenSSL-FIPS Version d40d43f6e6f28cdaf549e0fbc0f3d4a45b003e10 | 2789 |
| FCS_COP.1(d) | AES Key Wrap according to SP 800-38F Section 6.2 using AES-256 | SAIFE Library Key Wrap Version 158ff199ab1c013547289147b7bef44a8bd3f15a | 4251 |
| FCS_COP.1(f) | AES 256 in XTS mode | CoolEngine AES-XTS Version 1.7 | 4250 |
| FCS_PCC_EXT.1 | SP 800-132 password based key derivation: HMAC-SHA-256, 1500 iterations, 32 byte salt | | N/A |
| FCS_RBG_EXT.1 | SP 800-90A CTR_DRBG using AES-256 | OpenSSL-FIPS Version d40d43f6e6f28cdaf549e0fbc0f3d4a45b003e10 | 1329 |
| FCS_SNI_EXT.1 | Salts: 32 bytes from the DRBG (Used for PBKDF)<br>Nonce: N/A | | N/A |

| Table 7: Cryptographic Algorithms | | | |
|---|---|---|---|
| SFR | Description | Library/Implementation | CAVP Cert. |
| | CBC IV: N/A<br>XTS Tweak: The tweak value is initialized to zero, then incremented per 512-octet data unit processed. The Data Block Number is the number of the AES codebook size data block (16 octets) being processed within a 512-octet data unit. | | |

The DRBG is seeded by a third party RNG provided by the host platform. The TSF receives 37 bytes of seed data from the host platform and uses this data to directly seed the DBBG. The TSF assumes that this third party entropy source provides 7 bits of entropy per byte of seed data, so the TSF is assumed to be seeded with 259 bits of entropy.

FCS_RBG_EXT.1

### 7.1.2 Cryptographic Key Management

The TSF automatically generates the DEK and key encrypting key (KEK) during the provisioning process (preparative procedures). These keys are generated using the DRBG described in Section 7.1.1 by calling the DRBG's generate function.

FCS_CKM.1, FCS_CKM.1(c)

The TSF utilizes the following cryptographic keys, critical security parameters, and submasks:

1. Password
2. Password Key
3. BEV
4. DEK
5. DRBG State

The TSF stores all of these values as plaintext in RAM during operation. The values in RAM are considered to be no longer needed and zeroized when the TSF is powered off, the TSF is zeroized, the user locks the TSF (logs out), or the TSF detects the session with the handset is idle (30 seconds of inactivity).

The encrypted BEV and encrypted DEK are the only values stored in flash. The Password Split in non-volatile memory is zeroized when the TSF is zeroized or the user changes his or her Password. The DRBG state is zeroized when the TSF is zeroized. The DEK is zeroized when the TSF is zeroized or the user changes the DEK.

FCS_CKM_EXT.4

The TSF zeroizes RAM and flash by performing an overwrite with zeros. The TSF then reads the memory location back. If the memory is not read to be zeros, the TSF repeats the zeroization process.

FCS_CKM.4

### 7.1.3 Key Chaining

The TSF receives a password based authorization factor (Password) from the environment to begin the authentication and unlocking process. The authorization factor can be between 4 and 4096 bytes. The TSF performs password based key derivation as described in Section 7.1.1 to transform the authorization factor into the 256-bit Password Key. The TSF unwraps the 256 bit BEV using AES KW and the Password Key. The TSF validates whether the BEV is correct by using it to AES 256 CBC decrypt a known value. If the decrypted value is correct, the TSF uses the BEV to unwrap the DEK using ASE KW. Each step in the chain only proceeds if key validation succeeds though successful AES KW or decrypting the expected known value. Upon 10 consecutive validation attempts, the TSF blocks additional validation attempts by zeroizing the DEK and resetting itself to factory defaults. Once the DEK is successfully unwrapped, the TSF is considered to be unlocked and User Data Partition access can be performed as described in Section 7.2.1

FCS_AFA_EXT.1, FCS_PCC_EXT.1, FCS_KYC_EXT.1, FCS_KYC_EXT.2, FCS_SMV_EXT.1

## 7.2 User Data Protection

### 7.2.1 Protection of Data on Disk

The only physical interface to the TSF is the microSD card bus. The TSF proxies all communication over the MicroSD card bus and presents the host device with one partition, the Plaintext Partition. Internally, the TSF also maintains a User Data Partition that contains encrypted User Data; however, this partition is not presented to the host device. The TSF accepts microSD writes to the Plaintext Partition memory addresses, but does not pass them to the physical flash memory. The iTraffic Control file is a virtual file, as described below, whose memory address is within the Plaintext Partition address space. The TSF passes on all read requests for the Plaintext Partition to the physical flash memory, with the exception of reads from the iTraffic Control file. This arrangement prevents any user data from being written to the Plaintext Partition. Only factory defined formatting can be read from the Plaintext Partition.

The iTraffic Control file is a virtual file (only exists in RAM) that allows the host device to interact with the TSF. Each write to the iTraffic Control file is treated as a command to the TSF. Once a command has been issued, the TSF allows the host device to read the result from the iTraffic Control file. The TSF implements the iTraffic Control file as two separate FIFO buffers, an input buffer and an output buffer. Data written to the iTraffic Control file cannot subsequently be read back out, and data cannot be read more than one time.

Reads from and writes to the User Data Partition can only be accomplished by issuing a read or write command to the iTraffic Control file while the TSF is initialized and unlocked. When the TSF receives a write command, it applies AES XTS encryption to the data and writes the data to the User Data Partition. When the TSF receives a read command; it reads the encrypted data from the User Data Partition, performs AES XTS decryption, and returns the plaintext data to the user.

FDP_DSK_EXT.1

### 7.2.2 Initialization of the User Data Partition

During the initialization process (also referred to as provisioning), the user must provide entropy as described in Section 7.1.1 and a PIN/passphrase. The TSF seeds its DRBG with the entropy and generates a DEK and BEV. The TSF also performs Password Based Key Derivation to generate a Password Key from the PIN/passphrase. The TSF AES KW encrypts the DEK using the BEV and AES KW encrypts the BEV with the Password Key. The TSF then automatically proceeds to encrypt the User Data Partition using AES XTS and the DEK. The TSF does not allow any User Data Partition access while the initial encryption is in process. Once encryption is complete, the TSF is considered to be unlocked and User Data Partition read/write operations may commence as described in Section 7.2.1.

FDP_DSK_EXT.1

## 7.3 Security Management

### 7.3.1 Specification of Management Functions

The when the User is configuring the TSF, the TSF requires that Passwords are at least 4 characters long. The TSF supports passwords up to 4096 characters long and supports any 8 bit value as an allowable character.

FCS_PCC_EXT.1

The TSF allows the User to change the DEK by cryptographically erasing the DEK and re-provisioning the TOE. The TSF allows the User to cryptographically erase the DEK by issuing the zeroize command or failing 10 consecutive authentication attempts.

The TSF allows the User to initiate updates by issuing a Firmware Update command. The updates are loaded as described in Section 7.4.2.

FMT_SMF.1

## 7.4 Protection of the TSF

### 7.4.1 Protection of Key and Key Material

As describe in Section 7.1.2, the TSF does not store any plaintext keys, key material, or submasks in non-volatile memory.

FPT_KYP_EXT.1

### 7.4.2 Trusted Update

The TSF contains a certificate chain consisting of an ECDSA P-384 root certificate, an ECDSA P-384 intermediate certificate, an ECDSA P-256 intermediate certificate, and an ECDSA P-256 package signing certificate. These certificates are stored within the TSF on flash as part of the MicroCloud Manager package. The vendor maintains control of the associated private keys.

The TSF contains an ECDSA P-256 public key that is used to authenticate MicroCloud Linux updates. The TSF stores this key as part of the unmodifiable bootloader in Serial Peripheral Interface (SPI) flash. The SPI flash is locked in read-only mode, so the Linux update key cannot be changed. The vendor maintains control of the associated private key.

When the user initiates a firmware update, the firmware update is transferred using the iTraffic Control file. The TSF stores the incoming update in a temporary location on flash. Once the transfer is complete, the TSF checks the signature of the loaded each package in the update with the ECDSA P-256 package signing certificate or Linux update key. If the all signatures is valid, the TSF installs the update and may reboot to invoke the update. If any signature is not valid, the TSF deletes the update and return an error.

The TSF also provides the user with to commands to examine the version of the TSF. The TSF returns the HW version when the SD card identification number (CID) when the SD ALL_SEND_CID (CMD2) command is issued over the SD bus during SD bus initialization. The serial number and HW version are included in the CID. The TSF returns the  version of the MicroCloud Linux and version of the MicroCloud Manager when the update process is initiated. The TSF sends the host device a list of the packages and their versions as the first step in the update process.

FPT_TUD_EXT.1, FCS_COP.1(a)

### 7.4.3 TSF Testing

During power-up, the TSF performs a CRC integrity test of the boot loader. Once the integrity of the bootloader has been verified, the TSF performs an ECDSA P-256 with SHA-256 digital signature verification of the MicroCloud Linux using the Linux update key stored in the read-only SPI flash. The TSF verifies the integrity of MicroCloud Manager using the ECDSA P-256 package signing certificate. The package signing certificate can be updated by updating the MicroCloud Manager package.

The TSF performs the following known answer tests (KATs) to verify the correct operation of the cryptographic functions:

1. CTR_DRBG with ASE 256: The TSF instantiates the DRBG with a known value, invokes the generate function, and compares the generated bits to the expected bits. This satisfies the SP 800-90Ar1 Section 11.3 Health Tests by showing the correct operation of the seed, reseed, and generate functions.
2. ECDSA P-256 with SHA-256 Signature Verification: satisfied by the Firmware Integrity signature verification test above.
3. ECDSA P-256 with SHA-256 Signature Verification KAT (Trusted Update Implementation)
4. SHA-256: Hash of a known message
5. HMAC-SHA-256: MAC generation with a known key and message.
6. AES KW 256-bit encrypt and decrypt KATs
7. AES 256 XTS Encrypt/Decrypt: This show the correct operation of AES Encrypt and Decrypt primitive functions with a 256 bit key.

FPT_TST_EXT.1, FCS_COP.1(a)

# 8 Terms and Definitions

Table 8: cPP Glossary

| Term | Description |
|---|---|
| Authorization Factor | A value that a user knows, has, or is (e.g. password, token, etc) submitted to the TOE to establish that the user is in the community authorized to use the hard disk and that is used in the derivation or decryption of the BEV and eventual decryption of the DEK. Note that these values may or may not be used to establish the particular identity of the user. |
| Assurance | Grounds for confidence that a TOE meets the SFRs [CC1]. |
| Border Encryption Value | A value passed from the AA to the EE intended to link the key chains of the two components. |
| Key Sanitization | A method of sanitizing encrypted data by securely overwriting the key that was encrypting the data. |
| Data Encryption Key (DEK) | A key used to encrypt data-at-rest. |
| Full Drive Encryption | Refers to partitions of logical blocks of user accessible data as managed by the host system that indexes and partitions and an operating system that maps authorization to read or write data to blocks in these partitions. For the sake of this Security Program Definition (SPD) and cPP, FDE performs encryption and authorization on one partition, so defined and supported by the OS and file system jointly, under consideration. FDE products encrypt all data (with certain exceptions) on the partition of the storage device and permits access to the data only after successful authorization to the FDE solution. The exceptions include the necessity to leave a portion of the storage device (the size may vary based on implementation) unencrypted for such things as the Master Boot Record (MBR) or other AA/EE pre-authentication software. These FDE cPPs interpret the term "full drive encryption" to allow FDE solutions to leave a portion of the storage device unencrypted so long as it contains no protected data. |
| Intermediate Key | A key used in a point between the initial user authorization and the DEK. |
| Host Platform | The local hardware and software the TOE is running on, this does not include any peripheral devices (e.g. USB devices) that may be connected to the local hardware and software. |
| Key Chaining | The method of using multiple layers of encryption keys to protect data. A top layer key encrypts a lower layer key which encrypts the data; this method can have any number of layers. |
| Key Encryption Key (KEK) | A key used to encrypt other keys, such as DEKs or storage that contains keys. |
| Key Material | Key material is commonly known as critical security parameter (CSP) data, and also includes authorization data, nonces, and metadata. |
| Key Release Key (KRK) | A key used to release another key from storage, it is not used for the direct derivation or decryption of another key. |
| Operating System (OS) | Software which runs at the highest privilege level and can directly control hardware resources. |
| Non-Volatile Memory | A type of computer memory that will retain information without power. |
| Powered-Off State | The device has been shutdown. |
| Protected Data | This refers to all data on the storage device with the exception of a small portion required for the TOE to function correctly. It is all space on the disk a user could write data to and includes the operating system, applications, and user data. Protected data does not include the Master Boot Record or Pre-authentication area of the drive – areas of the drive that are necessarily unencrypted. |
| Submask | A submask is a bit string that can be generated and stored in a number of ways. |
| Target of Evaluation | A set of software, firmware and/or hardware possibly accompanied by guidance. [CC1] |

| Table 9: CC Abbreviations and Acronyms | |
|---|---|
| Abbreviations/ Acronyms | Description |
| AA | Authorization Acquisition |
| AES | Advanced Encryption Standard |
| BEV | Border Encryption Value |
| BIOS | Basic Input Output System |
| CBC | Cipher Block Chaining |
| CC | Common Criteria |
| CCM | Counter with CBC-Message Authentication Code |
| CEM | Common Evaluation Methodology |
| CPP | Collaborative Protection Profile |
| DEK | Data Encryption Key |
| DRBG | Deterministic Random Bit Generator |
| DSS | Digital Signature Standard |
| ECC | Elliptic Curve Cryptography |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| EE | Encryption Engine |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| FIPS | Federal Information Processing Standards |
| FDE | Full Drive Encryption |
| FFC | Finite Field Cryptography |
| GCM | Galois Counter Mode |
| HMAC | Keyed-Hash Message Authentication Code |
| IEEE | Institute of Electrical and Electronics Engineers |
| IT | Information Technology |
| ITSEF | IT Security Evaluation Facility |
| ISO/IEC | International Organization for Standardization / International Electrotechnical Commission |
| IV | Initialization Vector |
| KEK | Key Encryption Key |
| KMD | Key Management Description |
| KRK | Key Release Key |
| MBR | Master Boot Record |
| NIST | National Institute of Standards and Technology |
| OS | Operating System |
| RBG | Random Bit Generator |
| RNG | Random Number Generator |
| RSA | Rivest Shamir Adleman Algorithm |
| SAR | Security Assurance Requirement |
| SED | Self Encrypting Drive |
| SHA | Secure Hash Algorithm |
| SFR | Security Functional Requirement |
| SPD | Security Problem Definition |
| SPI | Serial Peripheral Interface |
| ST | Security Target |
| TOE | Target of Evaluation |
| TPM | Trusted Platform Module |
| TSF | TOE Security Functionality |
| TSS | TOE Summary Specification |
| USB | Universal Serial Bus |

Table 9: CC Abbreviations and Acronyms

| Abbreviations/ Acronyms | Description |
|---|---|
| XOR | Exclusive or |
| XTS | XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing |

# 9 References

Table 10: TOE Guidance Documentation

| Reference | Description | Version | Date |
|---|---|---|---|
| [1] | Bluechip Systems MicroCloud X4 Operation Manual | 7e4 | |
| [2] | MicroCloud X4 Key Management Description | 2.3 | January 13, 2017 |
| [3] | MicroCloud X4 Entropy Essay | 2.0 | February 27, 2017 |

Table 11: Common Criteria v3.1 References

| Reference | Description | Version | Date |
|---|---|---|---|
| [4] | Common Criteria for Information Technology Security Evaluation<br>Part 1: Introduction and general model CCMB-2009-07-001 | V3.1 R4 | July 2009 |
| [5] | Common Criteria for Information Technology Security Evaluation<br>Part 2: Security functional components CCMB-2009-07-002 | V3.1 R4 | July 2009 |
| [6] | Common Criteria for Information Technology Security Evaluation<br>Part 3: Security assurance components CCMB-2009-07-003 | V3.1 R4 | July 2009 |
| [7] | Common Criteria for Information Technology Security Evaluation<br>Evaluation Methodology CCMB-2009-07-004 | V3.1 R4 | July 2009 |

Table 12: Supporting Documentation

| Reference | Description | Version | Date |
|---|---|---|---|
| [8] | collaborative Protection Profile for Full Drive Encryption – Encryption Engine | 1.0 | January 26, 2015 |
| [9] | Supporting Document Mandatory Technical Document Full Drive Encryption: Encryption Engine | 1.0 | January 2015 |
| [10] | collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition | 1.0 | January 26, 2015 |
| [11] | Supporting Document Mandatory Technical Document Full Drive Encryption: Authorization Acquisition | 1.0 | January 2015 |