collaborative Protection Profile for Full Drive Encryption - Encryption Engine

February 1, 2019

Version 2.0 + Errata 20190201

Acknowledgements

This collaborative Protection Profile (cPP) was developed by the Full Drive Encryption international Technical Community with representatives from industry, Government agencies, Common Criteria Test Laboratories, and members of academia.

0. Preface

0.1 Objectives of Document

This document presents the Common Criteria (CC) collaborative Protection Profile (cPP) to express the security functional requirements (SFRs) and security assurance requirements (SARs) for a Full Drive Encryption - Encryption Engine. The Evaluation Activities that specify the actions the evaluator performs to determine if a product satisfies the SFRs captured within this cPP are described in *Supporting Document (Mandatory Technical Document) Full Drive Encryption: Encryption Engine, February 2019.*

A complete FDE solution requires both an Authorization Acquisition component and Encryption Engine component. A product may provide the entire solution and claim conformance to this cPP (Full Drive Encryption: Encryption Engine (FDE-EE)), and the Full Drive Encryption: Authorization Acquisition (FDE-AA) cPP.

However, because the FDE AA/EE Protection Profile suite is in its infancy, it is not yet possible to mandate that all dependent products will conform to a cPP. Non-validated dependent products (i.e., EE) may be considered to be an acceptable part of the Operational Environment for the AA TOE/product on a case-by-case basis as determined by the relevant national scheme.

The FDE iTC intends to develop guidance for developers whose products provide both components (i.e., an AA and EE) to aid them in developing a Security Target (ST) that can claim conformance to both FDE cPPs. One important aspect to note is:

Note to ST authors: **There is a selection in the ASE_TSS that must be completed. One cannot simply reference the SARs in this cPP.**

0.2 Scope of Document

The scope of the cPP within the development and evaluation process is described in the Common Criteria for Information Technology Security Evaluation. In particular, a cPP defines the IT security requirements of a technology specific type of TOE and specifies the functional and assurance security requirements to be met by a compliant TOE.

0.3 Intended Readership

The target audiences of this cPP are developers, CC consumers, system integrators, evaluators and schemes.

0.4 Related Documents

Protection Profiles

[FDE – AA] collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition, Version 2.0 + Errata 20190201, February 1, 2019

1 Common Criteria[1]

[CC1] Common Criteria for Information Technology Security Evaluation,
Part 1: Introduction and General Model,
CCMB-2017-04-001, Version 3.1 Revision 5, April 2017.

[CC2] Common Criteria for Information Technology Security Evaluation,
Part 2: Security Functional Components,
CCMB-2017-04-002, Version 3.1 Revision 5, April 2017.

[CC3] Common Criteria for Information Technology Security Evaluation,
Part 3: Security Assurance Components,
CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.

[CEM] Common Methodology for Information Technology Security Evaluation,
Evaluation Methodology,
CCMB-2017-04-004, Version 3.1, Revision 5, April 2017.

[SD] Supporting Document (Mandatory Technical Document), Full Drive Encryption: Encryption Engine, February 2019.

2

[1] For details see http://www.commoncriteriaportal.org/

0.5 Revision History

Version	Date	Description
0.1	August 26, 2014	Initial Release for iTC review
0.2	September 5, 2014	Draft published for Public review
0.13	October 17, 2014	Incorporated comments received from the Public review
1.0	January 26, 2015	Incorporated comments received from the CCDB review
1.5	September 2, 2015	Revised based on additional use cases developed by iTC
2.0	September 09, 2016	Incorporated comments received from the public review, and also updated the Key Destruction section and AVA_VAN.
2.0 + Errata 20190201	February 1, 2019	Updated to reflect CC Part 3 evaluation findings and FDE Interpretation Team [FIT] rulings

Contents

Figures / Tables

1. PP Introduction

1.1 PP Reference Identification

PP Reference: collaborative Protection Profile for Full Drive Encryption - Encryption Engine

PP Version: 2.0 + Errata 20190201

PP Date: February 1, 2019

1.2 Introduction to the FDE Collaborative Protection Profiles (cPPs) Effort

The purpose of the first set of Collaborative Protection Profiles (cPPs) for *Full Drive Encryption (FDE): Authorization Acquisition (AA)* and *Encryption Engine (EE)* is to provide requirements for Data-at-Rest protection for a lost device that contains storage. These cPPs allow FDE solutions based in software and/or hardware to meet the requirements. The form factor for a storage device may vary, but could include: system hard drives/solid state drives in servers, workstations, laptops, mobile devices, tablets, and external media. A hardware solution could be a Self-Encrypting Drive or other hardware-based solutions; the interface (USB, SATA, etc.) used to connect the storage device to the host machine is outside the scope.

Full Drive Encryption encrypts all data (with certain exceptions) on 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 user or authorization data.

Since the FDE cPPs support a variety of solutions, two cPPs describe the requirements for the FDE components shown in Figure 1.

Figure 1: FDE Components

The *FDE cPP - Authorization Acquisition* describes the requirements for the Authorization Acquisition piece and details the necessary security requirements and assurance activities necessary to interact with a user and result in the availability of a Border Encryption Value (BEV).

The *FDE cPP - Encryption Engine* describes the requirements for the Encryption Engine piece and details the necessary security requirements and assurance activities for the actual encryption/decryption of the data by the DEK. Each cPP will also have a set of core requirements for management functions, proper handling of cryptographic keys, updates performed in a trusted manner, audit and self-tests.

This TOE description defines the scope and functionality of the Encryption Engine, and the Security Problem Definition describes the assumptions made about the operating environment and the threats to the EE that the cPP requirements address.

1.3 Implementations

Full Disk Encryption solutions vary with implementation and vendor combinations.

Therefore, vendors will evaluate products that provide both components of the Full Disk Encryption Solution (AA and EE) against both cPPs – could be done in a single evaluation with one ST. A vendor that provides a single component of a FDE solution would only evaluate against the applicable cPP. The FDE cPP is divided into two documents to allow labs to independently evaluate solutions tailored to one cPP or the other. When a customer acquires an FDE solution, they will either obtain a single vendor product that meets the AA + EE cPPs or two products, one of which meets the AA and the other of which meets the EE cPPs.

The table below illustrates a few *examples* for certification.

Table 1: Examples of cPP Implementations

Implementation	cPP	Description
Host	AA	Host software provides the interface to a self-encrypting drive
Self-Encrypting Drive (SED)	EE	A self-encrypting drive used in combination with separate host software
Software FDE	AA + EE	A software full drive encryption solution
Hybrid	AA + EE	A single vendor's combination of hardware (e.g. hardware encryption engine or cryptographic co-processor) and software

1.4 Target of Evaluation (TOE) Overview

The target of evaluation for this cPP is either the Encryption Engine or a combined evaluation of the set of cPP's for FDE (Authorization Acquisition and Encryption Engine).

The following sections provide an overview of the functionality of the FDE EE cPP as well as the security capabilities.

1.4.1 Encryption Engine Introduction

The Encryption Engine cPP objectives focus on data encryption, policy enforcement, and key management. The EE is responsible for the generation, update, archival, recovery, protection, and destruction of the DEK and other intermediate keys under its control. The EE receives a BEV from the AA. The EE uses that BEV for the decryption of the DEK although other intermediate keys may exist in between those two points. Key encryption keys (KEKs) wrap other keys, notably the DEK or other intermediary keys which chain to the DEK. Key releasing keys (KRKs) authorize the EE to release either the DEK or other intermediary keys which chain to the DEK. These keys only differ in the functional use.

The EE determines whether to allow or deny a requested action based on the KEK or KRK provided by the AA. Possible requested actions include but are not limited to changing of encryption keys, decryption of data, and key sanitization of encryption keys (including the

DEK). The EE may offer additional policy enforcement to prevent access to ciphertext or the unencrypted portion of the storage device. Additionally the EE may provide encryption support for multiple users on an individual basis.

Figure 2 illustrates the components within EE and its relationship with AA.

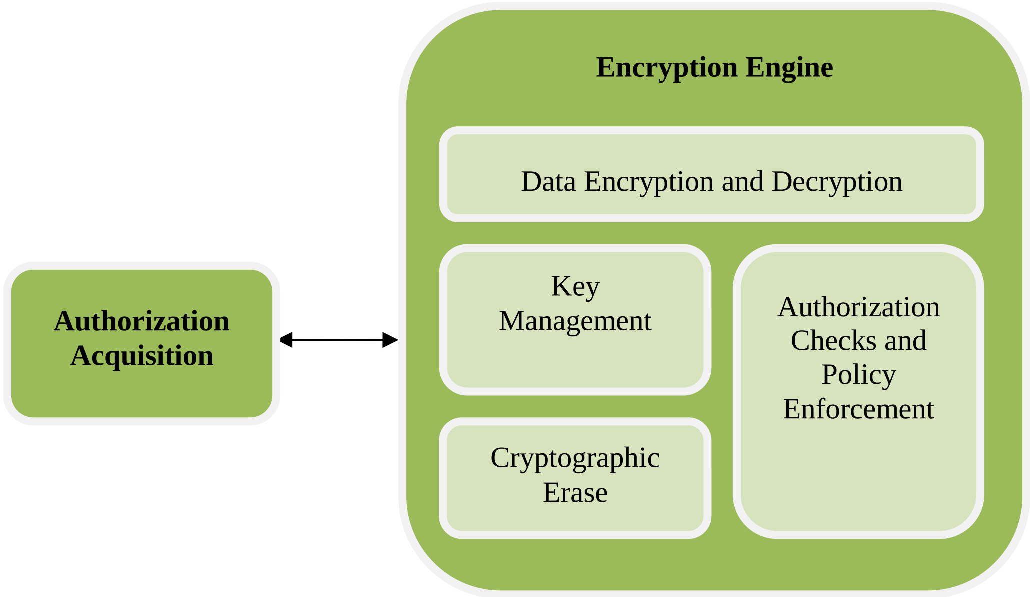


Figure 2: Encryption Engine Details

1.4.2 Encryption Engine Security Capabilities

The Encryption Engine is ultimately responsible for ensuring that the data is encrypted using a prescribed set of algorithms. The EE manages the decryption of the data on the storage device through decryption of the DEK based on the validity of the BEV provided by the AA. It also manages administrative functions, such as changing the DEK, managing the BEVs required for decrypting or releasing the DEK, managing the intermediate wrapping keys under its control, and performing a key sanitization.

The EE may provide key archiving and recovery functionality. The EE may manage the archiving and recovery itself, or interface with the AA to perform this function. It may also offer configurable features, which restricts the movement of keying material and disables recovery functionality.

The foremost security objective of encrypting storage devices is to force an adversary to perform an exhaustive search against a prohibitively large key space in order to recover the DEK or other intermediate keys. The EE uses approved cryptography to generate, handle, and protect keys to force an adversary who obtains an unpowered lost or stolen platform without the authorization factors or intermediate keys to exhaust the encryption key space of intermediate keys or DEK to obtain the data. The EE randomly generates DEKs and – in some cases - intermediate keys. The EE uses DEKs in a symmetric encryption algorithm in an appropriate mode along with appropriate initialization vectors for that mode to encrypt storage units (e.g. sectors or blocks) on the storage device. The EE either encrypts the DEK with a KEK or an intermediate key.

This version of the cPP includes additional security features, included advanced power saving requirements and firmware signing requirements.

1.4.3 The TOE and the Operational/Pre-Boot Environments

The environment in which the EE functions may differ depending on the boot stage of the platform in which it operates; see Figure 3. Aspects of initialization, and perhaps authorization may be performed in the Pre-Boot environment, while provisioning, encryption, decryption and management functionality are likely performed in the Operating System environment. Some of these aspects may occur in both envrioments.

The Operating System environment may make a full range of services available to the Encryption Engine, including hardware drivers, cryptographic libraries, and perhaps other services external to the TOE.

The Pre-Boot environment is much more constrained with limited capabilities. This environment turns on the minimum number of peripherals and loads only those drivers necessary to bring the platform from a cold start to executing a fully functional operating system with running applications.

The EE TOE may include or leverage features and functions within the operational environment.

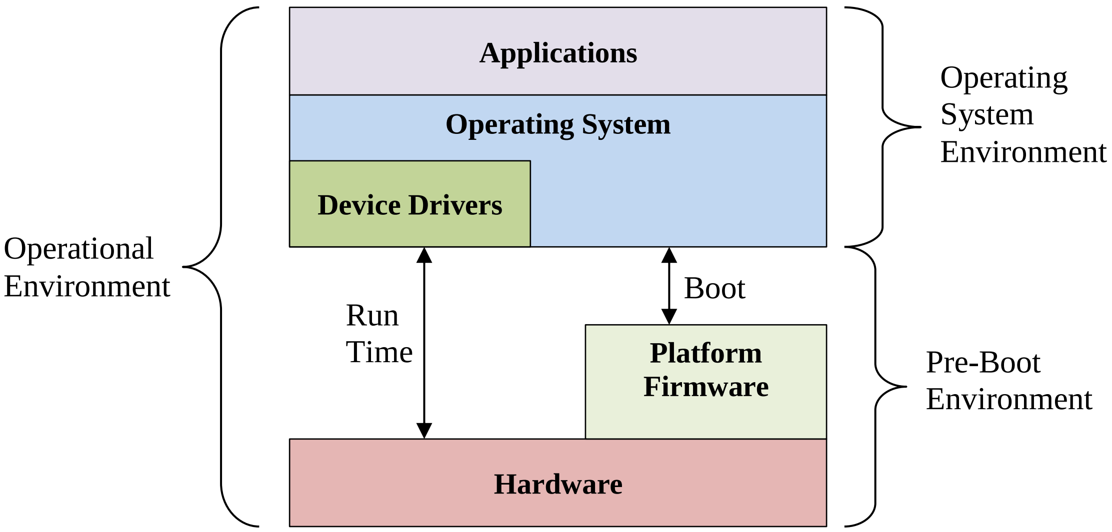


Figure 3: Operational Environment

1.5 Functionality Deferred until the Next cPP

Due to time constraints, this cPP defers requirements for some important functionality until the next version of the cPP. These include requirements for partition/volume management.

1.6 TOE Use Case

The use case for a product conforming to the FDE cPPs is to protect data at rest on a device that is lost or stolen while powered off without any prior access by an adversary. The use case where an adversary obtains a device that is in a powered state and is able to make modifications to the environment or the TOE itself (e.g., evil maid attacks) is not addressed by these cPPs (i.e., FDE-AA and FDE- EE).

2. CC Conformance Claims

As defined by the references [CC1], [CC2], and [CC3], this cPP conforms to the requirements of Common Criteria v3.1, Release 5. This cPP is CC Part 2 extended and CC Part 3 conformant. Extended component definitions can be found in Appendix C.

The methodology applied for the cPP evaluation is defined in [CEM].

This cPP satisfies the following Assurance Families: APE_CCL.1, APE_ECD.1, APE_INT.1, APE_OBJ.1, APE_REQ.1 and APE_SPD.1.

This cPP does not claim conformance to another cPP.

In order to be conformant to this cPP, a TOE must demonstrate *Exact Conformance*. *Exact Conformance* is defined as the ST containing all of the requirements in section 5 of this cPP, and potentially requirements from Appendix A or Appendix B of this cPP. While iteration is allowed, no additional requirements (from the CC parts 2 or 3) are allowed to be included in the ST. Further, no requirements in section 5 of this cPP are allowed to be omitted.

An ST can claim conformance with the FDE-AA in combination with the FDE-EE and still maintain exact conformance.

The FDE Enterprise Management (EM) PP-Module can specify the FDE-AA and FDE-EE as a set of Base-PPs for use with the FDE-EM in a PP-configuration.

3. Security Problem Definition

3.1 Threats

This section provides a narrative that describes how the requirements mitigate the mapped threats. A requirement may mitigate aspects of multiple threats. A requirement may only mitigate a threat in a limited way. Some requirements are optional, either because the TSF fully mitigates the threat without the additional requirement(s) being claimed or because the TSF relies on its Operational Environment to provide the functionality that is described by the optional requirement(s).

A threat consists of a threat agent, an asset and an adverse action of that threat agent on that asset. The threat agents are the entities that put the assets at risk if an adversary obtains a lost or stolen storage device. Threats drive the functional requirements for the target of evaluation (TOE). For instance, one threat below is T.UNAUTHORIZED_DATA_ACCESS. The threat agent is the possessor (unauthorized user) of a lost or stolen storage device. The asset is the data on the storage device, while the adverse action is to attempt to obtain those data from the storage device. This threat drives the functional requirements for the encrypted storage device (TOE) to authorize who can use the TOE to access the hard disk and encrypt/decrypt the data. Since possession of the KEK, DEK, intermediate 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, this SPD considers keying material equivalent to the data in importance and they appear among the other assets addressed below.

It is important to reemphasize at this point that this collaborative Protection Profile does not expect the product (TOE) to defend against the possessor of the lost or stolen hard disk who can introduce malicious code or exploitable hardware components into the Target of Evaluation (TOE) or the Operational Environment. It assumes that the user physically protects the TOE and that the Operational Environment provides sufficient protection against logical attacks. One specific area where a conformant TOE offers some protection is in providing updates to the TOE; other than this area, though, this cPP mandates no other countermeasures. Similarly, these requirements do not address the "lost and found" hard disk problem, where an adversary may have taken the hard disk, compromised the unencrypted portions of the boot device (e.g., MBR, boot partition), and then made it available to be recovered by the original user so that they would execute the compromised code.

(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).

[FMT_SMF.1, FPT_PWR_EXT.1, FPT_PWR_EXT.2, FCS_SNI_EXT.1, FCS_VAL_EXT.1, FDP_DSK_EXT.1, FPT_TST_EXT.1, FCS_COP.1(f)]

Rationale: FDP_DSK_EXT.1 ensures the TOE performs full drive encryption, which includes all protected data. "Full Drive Encryption" defined in Appendix F refers to "partitions of logical blocks of user accessible data as defined by the file system that indexes and partitions and an operating system that maps authorization to read or write data to blocks in these partitions," with the exception of the MBR and other AA/EE pre-authentication software. This ensures that protected data is unexposed even if the device is lost.

FPT_PWR_EXT.1 defines what power states are compliant for the TOE. FPT_PWR_EXT.2 defines conditions in which the TOE will enter a compliant power state. These requirements ensure the device is secure if lost in a compliant power state.

FMT_SMF.1 ensures the TSF provides the functions necessary to manage important aspects of the TOE including requests to change and erase the DEK. The correct behaviour of all cryptographic functionality is verified through the use of self-tests [FPT_TST_EXT.1]. FCS_VAL_EXT.1 verifies correct authentication and limits attempts to decrypt the data. FCS_SNI_EXT.1 ensures proper nonces and IVs are used in the encryption of the data. FCS_COP.1(f) defines proper AES encryption.

(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.

[FCS_CKM.4(a), FCS_CKM.4(b), FCS_CKM_EXT.4(a), FCS_CKM_EXT.4(b), FCS_KYC_EXT.2, FMT_SMF.1, FPT_KYP_EXT.1, FPT_PWR_EXT.1, FPT_PWR_EXT.2, FCS_CKM.1(c), FCS_SNI_EXT.1, FCS_VAL_EXT.1, FPT_TST_EXT.1, FCS_CKM.1(a), FCS_CKM.1(b), FCS_COP.1(b), FCS_COP.1(c), FCS_COP.1(d), FCS_COP.1(e), FCS_COP.1(f), FCS_COP.1(g), FCS_KDF_EXT.1, FCS_RBG_EXT.1, FCS_SMC_EXT.1]

Rationale: The keying material that threat agents may attempt to compromise are generated as specified by FCS_CKM.1(a), (b), and FCS_CKM.1(c), all of which are generated properly via FCS_RBG_EXT.1. One or more submasks may be combined [FCS_SMC_EXT.1] and/or chained [FCS_KYC_EXT.2] to protect the DEK. The key chain can be maintained by several methods, including:

- Key derivation [FCS_KDF_EXT.1]
- Key wrapping [FCS_COP.1(d)]
- Key combining [FCS_SMC_EXT.1]
- Key transport [FCS_COP.1(e)]
- Key encryption [FCS_COP.1(g)]

These requirements ensure the BEV is properly protected. Proper generation of salts, nonces, and IVs [FCS_SNI_EXT.1] is performed to support cryptographic functions requiring their use (such as symmetric key generation and AES encryption and decryption using Galois/Counter Mode [GCM]). This may involve the use of a random bit generator [FCS_RBG_EXT.1]. FCS_VAL_EXT.1 defines methods for validation of the BEV such as hashing [FCS_COP.1(b)], keyed-hash message authentication [FCS_COP.1(c)], and decrypting a known value with the keying material [FCS_COP.1(f)]. Key data can also be protected using submask combining [FCS_SMC_EXT.1] which can also be done using a hash function. The correct behavior of all cryptographic functionality is verified through the use of self-tests [FPT_TST_EXT.1].

FPT_KYP_EXT.1 ensures unwrapped key material is not stored in non-volatile memory and FCS_CKM_EXT.4(a) along with FCS_CKM.4(a) ensures proper key material destruction; minimizing the exposure of plaintext keys and key material.

Secure power management is essential to ensuring that power saving states cannot be used by an attacker to access plaintext keying material. The TSF defines Compliant power saving states [FPT_PWR_EXT.1] that encrypt or destroy [FCS_CKM.4(b), FCS_CKM_EXT.4(b)] all keying material when entered by various conditions [FPT_PWR_EXT.2]. This material is not decrypted until the BEV is validated [FCS_VAL_EXT.1].

FMT_SMF.1 ensures the TSF provides the functions necessary to manage important aspects of the TOE including modifying and erasing cryptographic data.

(T.AUTHORIZATION_GUESSING) 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.

[FCS_SNI_EXT.1, FCS_VAL_EXT.1]

Rationale: [FCS_VAL_EXT.1] requires several options for enforcing validation, such as key sanitization of the DEK or when a configurable number of failed validation attempts is reached within a 24 hour period. This mitigates brute force attacks against authorization factors such as passwords and pins. Salts according to [FCS_SNI_EXT.1] may be used which will prevent pre-computed attacks.

(T.KEYSPACE_EXHAUST) Threat agents may perform a cryptographic exhaust against the key space. Poorly chosen encryption algorithms and/or parameters allow attackers to exhaust the key space through brute force and give them unauthorized access to the data.

[FCS_KYC_EXT.2, FCS_CKM.1(a), FCS_CKM.1(b), FCS_CKM.1(c), FCS_RBG_EXT.1]

Rationale: [FCS_CKM.1(a), (b), and (c)] and [FCS_RBG_EXT.1] ensure cryptographic keys are random and of an appropriate strength/length to make exhaustion attempts cryptographically difficult and cost prohibitive. [FCS_KYC_EXT.2] ensures all keys protecting the DEK are of the same strength.

(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.

[FCS_COP.1(f) (optional), FCS_SNI_EXT.1]

Rationale: FCS_COP.1(f) ensures the proper choice of encryption algorithm and mode. FCS_SNI_EXT.1 ensures proper handling of salts, nonces and initialization vectors.

(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.

[FCS_COP.1(f) (optional), FCS_SNI_EXT.1]

Rationale: FCS_COP.1(f) ensures the proper choice of encryption algorithm and mode. FCS_SNI_EXT.1 ensures proper handling of salts, nonces and initialization vectors.

(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 that bypasses the intended security features and provides them unauthorized access to data.

[FCS_COP.1(a) (optional), FMT_SMF.1, FPT_TUD_EXT.1]

Rationale: FPT_TUD_EXT.1 provides authorized users the ability to query the current version of the TOE software, initiate updates, and verify updates prior to installation using a manufacturer digital signature. FCS_COP.1(a) defines the signature function that is used to verify updates.

FMT_SMF.1 ensures the TSF provides the functions necessary to manage important behaviour of the TOE which includes the initiation of system software updates.

(T.UNAUTHORIZED_FIRMWARE_UPDATE) An attacker attempts to replace the firmware on the SED via a command from the AA or from the host platform with a malicious firmware update that may compromise the security features of the TOE.

[FCS_COP.1(a) (optional), FCS_COP.1(b) (optional), FMT_SMF.1, FPT_FUA_EXT.1(optional), FPT_TUD_EXT.1, FPT_FAC_EXT.1(optional), FPT_RBP_EXT.1(optional)]

Rationale: FPT_TUD_EXT.1 defines a secure mechanism for updating the TOE firmware, initiated by a management function provided by FMT_SMF.1. FCS_COP.1(a) and FCS_COP.1(b) define the cryptographic functions that can be used to validate the authenticity and integrity of firmware updates as defined by FPT_FUA_EXT.1. FPT_FAC_EXT.1 provides additional security by only allowing an update to be initiated if the initiator can provide information that would only be known to a trusted administrator. FPT_RBP_EXT.1 protects against a malicious or inadvertent downgrade of the firmware to an earlier version that may have security flaws not present in the more recent version.

(T.UNAUTHORIZED_FIRMWARE_MODIFY) An attacker attempts to modify the firmware in the SED via a command from the AA or from the host platform that may compromise the security features of the TOE.

[FPT_FUA_EXT.1 (optional), FPT_TUD_EXT.1]

Rationale: FPT_FUA_EXT.1 ensures that the existing firmware cannot be modified unless it is to replace it with a valid update that was initiated as part of FPT_TUD_EXT.1.

3.2 Assumptions

Assumptions that must remain true in order to mitigate the threats appear below:

(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.

[OE.TRUSTED_CHANNEL]

(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, un-partitioned space, and areas that must contain unencrypted code (e.g., MBR and AA/EE pre-authentication software) contain no protected data.

[OE.INITIAL_DRIVE_STATE]

(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.

[OE.PASSPHRASE_STRENGTH, OE. POWER_DOWN, OE.SINGLE_USE_ET, OE.TRAINED_USERS]

(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.

[OE.PLATFORM_STATE]

(A.POWER_DOWN) The user does not leave the platform and/or storage device unattended until the device is in a Compliant power saving state or has fully powered off. This properly clears memories and locks down the device. Authorized users do not leave the platform and/or storage device in a mode where sensitive information persists in non-volatile storage (e.g., lock screen 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".

[OE.POWER_DOWN]

(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.

[OE.STRONG_ENVIRONMENT_CRYPTO]

(A.PHYSICAL) The platform is assumed to be physically protected in its Operational Environment and not subject to physical attacks that compromise the security and/or interfere with the platform's correct operation.

[OE.PHYSICAL]

3.3 Organizational Security Policy

There are no organizational security policies addressed by this cPP.

4. Security Objectives

4.1 Security Objectives for the Operational Environment

The Operational Environment of the TOE implements technical and procedural measures to assist the TOE in correctly providing its security functionality. This part wise solution forms the security objectives for the Operational Environment and consists of a set of statements describing the goals that the Operational Environment should achieve.

(OE.TRUSTED_CHANNEL) Communication among and between product components (e.g., AA and EE) is sufficiently protected to prevent information disclosure.

> Rationale: In situations where there is an opportunity for an adversary to interpose themselves in the channel between the AA and the EE, a trusted channel should be established to prevent exploitation. [A.TRUSTED_CHANNEL] assumes the existence of a trusted channel between the AA and EE, except for when the boundary is within and does not breach the TOE or is in such close proximity that a breach is not possible without detection.

(OE.INITIAL_DRIVE_STATE) The OE provides a newly provisioned or initialized storage device free of protected data in areas not targeted for encryption.

> Rationale: Since the cPP requires all protected data be encrypted, A. INITIAL_DRIVE_STATE assumes that the initial state of the device targeted for FDE is free of protected data in those areas of the drive where encryption will not be invoked (e.g., MBR and AA/EE pre-authentication software). Given this known start state, the product (once installed and operational) ensures partitions of logical blocks of user accessible data is protected.

(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.

> Rationale: Users are properly trained [A.TRAINED_USER] to create authorization factors that conform to administrative guidance.

(OE.POWER_DOWN) Volatile memory is cleared after entering a Compliant power saving state or turned off so memory remnant attacks are infeasible.

> Rationale: Users are properly trained [A_TRAINED_USER] to not leave the storage device unattended until it is in a Compliant power saving state or fully turned off.

(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.

> Rationale: Users are properly trained [A.TRAINED_USER] to use external token authorization factors as intended and for no other purpose.

(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.

Rationale: All cryptography implemented in the Operational Environment and used by the product meets the requirements listed in this cPP [A.STRONG_CRYPTO].

(OE.TRAINED_USERS) Authorized users will be properly trained and follow all guidance for securing the TOE and authorization factors.

Rationale: Users are properly trained [A.TRAINED_USER] to create authorization factors that conform to guidance, not store external token authorization factors with the device, and power down the TOE when required (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.

A platform free of malware [A.PLATFORM_STATE] prevents an attack vector that could potentially interfere with the correct operation of the product.

(OE.PHYSICAL) The Operational Environment will provide a secure physical computing space such that an adversary is not able to make modifications to the environment or to the TOE itself.

Rationale: As stated in section 1.6, the use case for this cPP is to protect data at rest on a device where the adversary receives it in a powered off state and has no prior access.

5. Security Functional Requirements

The individual security functional requirements are specified in the sections below. Based on selections made in these SFRs it will also be necessary to include some of the selection-based SFRs in Appendix B. Additional optional SFRs may also be adopted from those listed in Appendix A for those functions that are provided by the TOE instead of its Operational Environment.

The Evaluation Activities defined in [SD] describe actions that the evaluator will take in order to determine compliance of a particular TOE with the SFRs. The content of these Evaluation Activities will therefore provide more insight into deliverables required from TOE Developers.

5.1 Conventions

The conventions used in descriptions of the SFRs are as follows:

- Assignment: Indicated with *italicized text*;
- Refinement made by PP author: Indicated with **bold text** or ~~strikethroughs~~ for text that is added to or removed from the original SFR—when the refinement operation substitutes text, only the added text is included;
- Selection: Indicated with underlined text;
- Assignment within a Selection: Indicated with *italicized and underlined text*;
- Iteration: Indicated by appending the SFR with parentheses that contain a letter that is unique for each iteration, e.g. (a), (b), (c) and/or with a slash (/) followed by a descriptive string for the SFR's purpose, e.g. /Server.

SFR text that is bold, italicized, and underlined indicates that the original SFR defined an assignment operation but the PP author completed that assignment by redefining it as a selection operation, which is also considered to be a refinement of the original SFR.

If the selection or assignment is to be completed by the ST author, it is preceded by 'selection:' or 'assignment:'. If the selection or assignment has been completed by the PP author and the ST author does not have the ability to modify it, the proper formatting convention is applied but the preceding word is not included. The exception to this is if the SFR definition includes multiple options in a selection or assignment and the PP has excluded certain options but at least two remain. In this case, the selection or assignment operations that are not permitted by this PP were removed without applying additional formatting and the 'selection:' or 'assignment:' text is preserved to show that the ST author still has the ability to choose from the reduced set of options.

Extended SFRs (i.e. those SFRs that are not defined in CC Part 2) are identified by having a label '_EXT' at the end of the SFR name.

5.2 SFR Architecture

The following table lists the SFRs that are mandated by this cPP.

Table 2 TOE Security Functional Requirements

Functional Class	Functional Components
Cryptographic Support (FCS)	FCS_CKM.1(c) Cryptographic Key Generation (Data Encryption Key)
	FCS_CKM.4(a) Cryptographic Key Destruction (Power Management)
	FCS_CKM_EXT.4(a) Cryptographic Key and Key Material Destruction (Destruction Timing)
	FCS_CKM_EXT.4(b) Cryptographic Key and Key Material Destruction (Power Management)
	FCS_CKM_EXT.6 Cryptographic Key Destruction Types
	FCS_KYC_EXT.2 Key Chaining (Recipient)
	FCS_SNI_EXT.1 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
	FCS_VAL_EXT.1 Validation
User Data Protection (FDP)	FDP_DSK_EXT.1 Protection of Data on Disk
Security Management (FMT)	FMT_SMF.1 Specification of Management Functions
Protection of the TSF (FPT)	FPT_KYP_EXT.1 Protection of Key and Key Material
	FPT_PWR_EXT.1 Power Saving States
	FPT_PWR_EXT.2 Timing of Power Saving States
	FPT_TST_EXT.1 TSF Testing
	FPT_TUD_EXT.1 Trusted Update

5.3 Class: Cryptographic Support (FCS)

FCS_CKM.1(c) Cryptographic Key Generation (Data Encryption Key)

FCS_CKM.1.1(c) Refinement: The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation ~~algorithm~~ **method [selection:**

- **generate a DEK using the RBG as specified in FCS_RBG_EXT.1,**
- **accept a DEK that is generated by the RBG provided by the host platform,**
- **accept a DEK that is wrapped as specified in FCS_COP.1(d)]**

and specified cryptographic key sizes [***selection: 128 bits, 256 bits***] ~~that meet the following: [*assignment: list of standards*]~~.

Application Note: *This SFR is iterated because additional iterations are defined as optional requirements in Appendix A. Iteration (c) was chosen specifically to ensure consistency between the FDE cPPs.*

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.

FCS_CKM.4(a) Cryptographic Key Destruction (Power Management)

FCS_CKM.4.1(a) Refinement: The TSF shall **[selection: instruct the Operational Environment to clear, erase] cryptographic keys and key material from volatile memory**

when transitioning to a Compliant power saving state as defined by FPT_PWR_EXT.1 that meets the following: [*a key destruction method specified in FCS_CKM_EXT.6*].

Application Note: *In some cases, erasure of keys from volatile memory is only supported by the Operational Environment, in which case the Operational Environment must expose a well-documented mechanism or interface to invoke the memory clearing operation.*

Self-encrypting drives do not store keys in the Operational Environment and cannot instruct the Operational Environment to perform functionality so they are not expected to select "instruct the Operational Environment to clear".

FCS_CKM_EXT.4(a) Cryptographic Key and Key Material Destruction (Destruction Timing)

FCS_CKM_EXT.4.1(a) The TSF shall destroy all keys and keying material when no longer needed.

Application Note: *Keys, including intermediate keys and key material that are no longer needed are destroyed by using an approved method, FCS_CKM_EXT.6. Examples of keys are intermediate keys, submasks, and BEV. 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.*

FCS_CKM_EXT.4(b) Cryptographic Key and Key Material Destruction (Power Management)

FCS_CKM_EXT.4.1(b) The TSF shall destroy all key material, BEV, and authentication factors stored in plaintext when transitioning to a Compliant power saving state as defined by FPT_PWR_EXT.1.

Application Note: *The TOE may end up in a non-Compliant power saving state indistinguishable from a Compliant power state (e.g. as result of sudden and/or unexpected power loss). Guidance documentation must state what conditions may result in clear text keys or key materials to stay in volatile memory and identify mitigation measures that result in clearing of volatile memory.*

FCS_CKM_EXT.6 Cryptographic Key Destruction Types

***FCS_CKM*_EXT.6.1** The TSF shall use **[*selection: FCS_CKM.4(b), FCS_CKM.4(c), FCS_CKM.4(d)*]** key destruction methods.

Application Note: *If multiple selections are made, the TSS should identify which keys are destroyed according to which selections.*

FCS_KYC_EXT.2 Key Chaining (Recipient)

FCS_KYC_EXT.2.1 The TSF shall accept a BEV of at least [selection: 128 bits, 256 bits] 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): [selection:

- asymmetric key generation as specified in FCS_CKM.1(a),
- symmetric key generation as specified in FCS_CKM.1(b),
- key derivation as specified in FCS_KDF_EXT.1,
- key wrapping as specified in FCS_COP.1(d),
- key combining as specified in FCS_SMC_EXT.1,
- key transport as specified in FCS_COP.1(e),
- key encryption as specified in FCS_COP.1(g)]

while maintaining an effective strength of [selection: 128 bits, 256 bits] for symmetric keys and an effective strength of [selection: not applicable, 112 bits, 128 bits, 192 bits, 256 bits] for asymmetric keys.

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).*

The BEV is considered to be equivalent to keying material and therefore additional checksums or similar values are not the BEV, even if they are sent with the BEV.

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 Appendix E for more information.

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

FCS_SNI_EXT.1.1 The TSF shall use [selection: use no salts, use salts that are generated by a [selection: DRBG as specified in FCS_RBG_EXT.1, DRBG provided by the host platform].

FCS_SNI_EXT.1.2 The TSF shall use [selection: no nonces, unique nonces with a minimum size of [*64*] bits].

FCS_SNI_EXT.1.3 The TSF shall create IVs in the following manner [selection:

- CBC: IVs shall be non-repeating and unpredictable;

- CCM: Nonce shall be non-repeating and unpredictable;
- 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].

Application Note: *This SFR does not prescribe when salts, nonces, and IVs must be used, only that when they are used they must be generated in a certain manner. The ST author is expected to document each claimed SFR that requires the use of salts, nonces, and/or IVs (such as symmetric key generation as defined by FCS_CKM.1(b) and AES encryption/decryption as defined by FCS_COP.1(f)). If the TSF does not use salts, nonces, or IVs for any function, then this SFR is considered to be vacuously satisfied.*

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 referred to as Starting Variable (SV) in ISO/IEC 19772.

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

FCS_VAL_EXT.1 Validation

FCS_VAL_EXT.1.1 The TSF shall perform validation of the [BEV] using the following method(s): [selection:

- key wrap as specified in FCS_COP.1(d);
- hash the [BEV] as specified in [selection: FCS_COP.1(b), FCS_COP.1(c)] and compare it to a stored hashed [**value**];
- decrypt a known value using the [selection: intermediate key, BEV] as specified in FCS_COP.1(f) and compare it against a stored known value]

FCS_VAL_EXT.1.2 The TSF shall require the validation of the [BEV] prior to [*allowing access to TSF data after exiting a Compliant power saving state*].

FCS_VAL_EXT.1.3 The TSF shall [selection:

- perform a key sanitization of the DEK upon a [selection: configurable number, [*assignment: ST author specified number*]] of consecutive failed validation attempts,
- institute a delay such that only [*assignment: ST author specified number of attempts*] can be made within a 24 hour period,
- block validation after [*assignment: ST author specified number of attempts*] of consecutive failed validation attempts,
- require power cycle/reset the TOE after [*assignment: ST author specified number of attempts*] of consecutive failed validation attempts].

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 cracker). The cryptographic functions (i.e., hash, decryption) performed are those specified in FCS_COP.1(b) and FCS_COP.1(f).

5.4 Class: User Data Protection (FDP)

This family is used to mandate the encryption of all protected data written to a drive.

FDP_DSK_EXT.1 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.

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 Operational Environment. Note that if this is the case, it is assumed that the environmental implementation of AES is consistent with the behavior described in FCS_COP.1(f). If the TOE provides the cryptographic functions to encrypt/decrypt the data, the ST author includes FCS_COP.1(f) as defined in Appendix A in the main body of the ST.

5.5 Class: Security Management (FMT)

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 Refinement: The TSF shall be capable of performing the following management functions: [

- *a) change the DEK, as specified in FCS_CKM.1, when re-provisioning or when commanded,*
- *b) erase the DEK, as specified in FCS_CKM.4(a),*
- *c) initiate TOE firmware/software updates,*
- d) ***[selection: no other functions, configure a password for firmware update, import a wrapped DEK, configure cryptographic functionality, disable key recovery functionality, securely update the public key needed for trusted update, configure the number of failed validation attempts required to trigger corrective behavior,***

configure the corrective behavior to issue in the event of an excessive number of failed validation attempts, [assignment: other management functions provided by the TSF]].

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 (d) 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 (d), if no other management functions are provided (or claimed), then "no other functions" should be selected. Default Authorization factors are the initial values that are used to manipulate the drive.*

For the purposes of this document, key sanitization means to destroy the DEK, using one of the approved destruction methods. This applies to instances of the protected key that exist in non-volatile storage.

5.6 Class: Protection of the TSF (FPT)

FPT_KYP_EXT.1 Protection of Key and Key Material

FPT_KYP_EXT.1.1 The TSF shall [selection:

- not store keys in non-volatile memory
- 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)
- only store plaintext keys that meet any one of the following criteria [selection:
    - the plaintext key is not part of the key chain as specified in FCS_KYC_EXT.2,
    - the plaintext key will no longer provide access to the encrypted data after initial provisioning,
    - the plaintext key is a key split that is combined as specified in FCS_SMC_EXT.1, and the other half of the key split is [selection:
        - wrapped as specified in FCS_COP.1(d),
        - encrypted as specified in FCS_COP.1(g) or FCS_COP.1(e),
        - derived and not stored in non-volatile memory],
    - the non-volatile memory the key is stored on is located in an external storage device for use as an authorization factor,
    - the plaintext key is [selection:
        - used to wrap a key as specified in FCS_COP.1(d),
        - used to encrypt a key as specified in FCS_COP.1(g) or FCS_COP.1(e)]

      that is already [selection:
        - wrapped as specified in FCS_COP.1(d),
        - encrypted as specified in FCS_COP.1(g) or FCS_COP.1(e)]]].

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 are 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.

FPT_PWR_EXT.1 Power Saving States

FPT_PWR_EXT.1.1 The TSF shall define the following Compliant power saving states: [selection: choose at least one of: S3, S4, G2(S5), G3, D0, D1, D2, D3 [*assignment: other power saving states*]].

Application Note: *Power saving states S3, S4, G2(S5), G3, D0, D1, D2, D3 are defined by the Advanced Configuration and Power Interface (ACPI) standard.*

FPT_PWR_EXT.2 Timing of Power Saving States

FPT_PWR_EXT.2.1 For each Compliant power saving state defined in FPT_PWR_EXT.1.1, the TSF shall enter the Compliant power saving state when the following conditions occur: user-initiated request, [selection: shutdown, user inactivity, request initiated by remote management system, [*assignment: other conditions*], no other conditions].

Application Note: *If volatile memory is not cleared as part of an unexpected power shutdown sequence then guidance documentation must define mitigation activities (e.g. how long users should wait after an unexpected power-down before volatile memory can be considered cleared).*

FPT_TST_EXT.1 TSF Testing

FPT_TST_EXT.1.1 The TSF shall run a suite of the following self-tests [selection: during initial start-up (on power on), at the conditions [*before the function is first invoked*]] to demonstrate the correct operation of the TSF: [*assignment: list of self-tests run by the TSF*].

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 should describe the known-answer self-tests for those functions.

The evaluator is expected to 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 should determine that all of the identified functions/components are adequately tested on start-up.

FPT_TUD_EXT.1 Trusted Update

FPT_TUD_EXT.1.1 Refinement: The TSF shall provide [*authorized users*] the ability to query the current version of the TOE **[selection: software, firmware]** ~~software/firmware~~.

FPT_TUD_EXT.1.2 Refinement: The TSF shall provide [*authorized users*] the ability to initiate updates to TOE **[selection: software, firmware]** ~~software/firmware~~.

FPT_TUD_EXT.1.3 Refinement: The TSF shall verify updates to the TOE [**selection: software, firmware**] using a [selection: digital signature **as specified in FCS_COP.1(a), authenticated firmware update mechanism as described in FPT_FUA_EXT.1**] by the manufacturer prior to installing those updates.

Application Note: *"Authorized users" refers to an individual who has rightful physical possession of the device.*

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.

If the TOE is a software product, the ST author selects 'digital signature'. If the TOE is a hardware product, the ST author selects 'authenticated firmware update mechanism as described in FPT_FUA_EXT.1'.

The secure firmware update mechanism is used for verifying the authenticity and integrity of the new update package and for ensuring that it is protected from modification outside of the secure update process. The authenticated firmware update mechanism should be protected from unintended or malicious modification by a mechanism that is at least as strong as that protecting the RTU and the firmware.

The intent of this requirement is to ensure that an authenticated firmware update mechanism will be provided. Authentication verifies that the firmware package was generated by an authentic source and is unaltered. All updates to the existing firmware should go through an authenticated update mechanism as described in FPT_FUA_EXT.1.

6. Security Assurance Requirements

This cPP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

Note to ST authors: **There is a selection in the ASE_TSS that must be completed. One cannot simply reference the SARs in this cPP.**

This section lists the set of SARs from CC part 3 that are required in evaluations against this cPP. Individual Evaluation Activities to be performed are specified in *Supporting Document (Mandatory Technical Document) Full Drive Encryption: Encryption Engine, February 2019.*

The general model for evaluation of TOEs against STs written to conform to this cPP is as follows: after the ST has been approved for evaluation, the ITSEF will obtain the TOE, supporting environmental IT (if required), and the administrative/user guides for the TOE. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The ITSEF also performs the Evaluation Activities contained within the SD, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The Evaluation Activities that are captured in the SD also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the cPP.

Table 3: Security Assurance Requirements

Assurance Class	Assurance Components
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)

6.1 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.

ASE_TSS.1.1C Refinement: The TOE summary specification shall describe how the TOE meets each SFR**, including a proprietary Key Management Description (Appendix E), and [selection: Entropy Essay, list of all of 3rd party software libraries (including version numbers), 3rd party hardware components (including model/version numbers), no other cPP specified proprietary documentation].**

6.2 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 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 may have interfaces to the Operational Environment that are not directly invoked 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.

6.3 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.

6.3.1 Operational User Guidance (AGD_OPE.1)

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

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.

6.3.2 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.

6.4 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.

6.4.1 Labelling 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.

6.4.2 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.

6.5 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.

6.5.1 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.

6.6 Class AVA: Vulnerability Assessment

For the current 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.

6.6.1 Vulnerability Survey (AVA_VAN.1)

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

Appendix A: Optional Requirements

As indicated in the introduction to this cPP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this cPP. Additionally, there are two other sets of requirements specified in Appendices A and B.

The first set (in this Appendix) are requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this cPP. The second set (in Appendix B) are requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements in that appendix would need to be included in the body of the ST (e.g., cryptographic protocols selected in a trusted channel requirement).

A.1 Internal Cryptographic Implementation

As indicated in the body of this cPP, it is acceptable for the TOE to either directly implement cryptographic functionality that supports the drive encryption/decryption process, or to use that functionality in the Operational Environment (for example, calling an Operating System's cryptographic provider interface; a third-party cryptographic library; or a hardware cryptographic accelerator). However, each one of these SFRs that can optionally be implemented by the Operational Environment are also considered to be ‘selection-based’ SFRs due to the fact that their functionality is contingent on the ST author make certain selections in other SFRs. Because of this, these SFRs have been placed in Appendix B. Note however that there is still an expectation that some of these functions may be provided by the Operational Environment, in which case it is acceptable to omit the SFRs in question so long as the ST author can provide evidence that the Operational Environment will include a cryptographic interface to the TSF that allows for secure usage of these functions, and that the functions have been validated to the same level of rigor as is described in [SD].

If all of the cryptographic functionality is implemented by the TSF and the TOE does not rely on its Operational Environment to provide any cryptographic services, the ST author shall omit OE.STRONG_ENVIRONMENT_CRYPTO and its corresponding assumption since the environment does not need to satisfy the objective in this case.

A.2 Firmware Update Validation

The TOE can either be a software or a hardware product. The ST author will select this through completion of FPT_TUD_EXT.1. If the TOE is a hardware product (i.e. a SED), this cPP defines several selection-based requirements that are intended to provide guarantees of the authenticity and integrity of firmware updates. However, some additional security measures may be provided by a conformant TOE in addition to these mechanisms. If the TSF provides these security measures, the ST author may optionally include one or more of the SFRs described below.

FPT_FAC_EXT.1 Firmware Access Control

FPT_FAC_EXT.1.1 The TSF shall require [selection: a password, a known unique value printed on the device, a privileged user action] before the firmware update proceeds.

Application Note: *Before an update takes place, the drive owner will authorize the update by providing either a known unique value (for example, a serial number) that is printed on the* *drive, a password (which should be administratively configurable as defined in FMT_SMF.1) or perform the operation as a privileged user. It is assumed that physical presence to the drive is limited to authorized personnel. If the correct value is not provided, the update will not take place. The values are intended to be unique per drive so they cannot be easily exhausted.*

The same requirements for cleaning up a password still apply.

FPT_RBP_EXT.1 Rollback Protection

FPT_RBP_EXT.1.1 The TSF shall verify that the new firmware package is not downgrading to a lower security version number by [*assignment: method of verifying the security version number is the same as or higher than the currently installed version*].

FPT_RBP_EXT.1.2 The TSF shall generate and return an error code if the attempted firmware update package is detected to be an invalid version.

Application Note: *This requirement prevents an unauthorized rollback of the firmware to an earlier authentic version. This mitigates against unknowing installation of an earlier authentic firmware version that may have a security weakness. It is expected that vendors will increase security version numbers with each new update package.*

For FPT_RBP_EXT.1.1 the purpose is to verify that the new package has a security version number equal to or larger than the security version number of currently installed firmware package.

The administrator guidance would include instructions for the administrator to configure the rollback prevention mechanism, if appropriate.

A.3 Cryptographic Key Destruction

The TOE can optionally choose to destroy keys by destroying the parent key through the key destruction requirements. This method of destruction mirrors the data destruction method commonly referred to as Secure Erase. Although the ST author still has to do the traditional key destruction requirements, this is an additional option that expands the methods of key destruction.

FCS_CKM.4(e) Cryptographic Key Destruction (Key Cryptographic Erase)

FCS_CKM.4.1(e) The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [*by using the appropriate method to destroy all encryption keys encrypting the key intended for destruction*] that meets the following: [*no standard*].

Application Note: *A key can be considered destroyed by destroying the key that protects the key. If a key is wrapped or encrypted it is not necessary to "overwrite" that key, overwriting the key that is used to wrap or encrypt the key used to encrypt/decrypt data, using the appropriate method for the memory type involved, will suffice. For example, if a product uses a Key Encryption Key (KEK) to encrypt a Data Encryption Key (DEK), destroying the KEK using one of the methods in FCS_CKM.EXT.6.1 is sufficient, since the DEK would no longer be usable (of course, presumes the DEK is still encrypted.*

Appendix B: Selection-Based Requirements

As indicated in the introduction to this cPP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this cPP. There are additional requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements below may need to be included.

Note that many of these selection-based SFRs could also be implemented by cryptographic services in the TOE's Operational Environment. If this is the case, it is not necessary to include the SFRs in question so long as the Operational Environment can be shown to provide equivalent functionality.

B.1 Class: Cryptographic Support (FCS)

FCS_CKM.1(a) Cryptographic Key Generation (Asymmetric Keys)

FCS_CKM.1.1(a) Refinement: The TSF shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm: [***selection:***

- ***RSA schemes using cryptographic key sizes of [selection: 2048-bit, 3072-bit, 4096-bit] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3;***
- ***ECC schemes using "NIST curves" of [selection: P-256, P-384, P-521] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;***
- ***FFC schemes using cryptographic key sizes of [selection: 2048-bit, 3072-bit, 4096-bit] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1***

] ~~and specified cryptographic key sizes [*assignment: cryptographic key sizes*] that meet the following: [*assignment: list of standards*]~~.

Application Note: *Asymmetric keys may be used to "wrap" a key or submask. This SFR should be included by the ST author when making the appropriate selection in FCS_COP.*

Asymmetric Keys may also be used for the key chain. Therefore, the ST author should select FCS_CKM.1(a), if Asymmetric key generation is used.

If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

For all schemes (RSA schemes, ECC schemes, FFC schemes), a RBG is needed to a) generate seeds for RSA and to b) generate private keys directly for ECC and FFC. So FCS_RBG_EXT.1 is used together with this SFR. A hash algorithm is also required when the key pair generation algorithm is selected based on either Appendix B.3.2 or B.3.5 of FIPS 186-4. So in such case, FCS_COP.1(d) is used together with this SFR.

FCS_CKM.1(b) Cryptographic Key Generation (Symmetric Keys)

FCS_CKM.1.1(b) Refinement: The TSF shall generate **symmetric** cryptographic keys **using a Random Bit Generator as specified in FCS_RBG_EXT.1** and specified cryptographic key sizes [***selection: 128 bit, 256 bit***] that meet the following: [*no standard*].

Application Note: *The symmetric key generation function may be used to generate keys along the key chain or a DEK. It may also be used to provide inputs for key combining, key encryption, or key wrapping. Therefore, the ST author should select FCS_CKM.1(b), if Symmetric key generation is used.*

FCS_CKM.4(b) Cryptographic Key Destruction (TOE-Controlled Hardware)

FCS_CKM.4.1(b) Refinement: The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **[*selection:***

- ***For volatile memory, the destruction shall be executed by a [selection:***
  - ***single overwrite consisting of [selection:***
    - ***a pseudo-random pattern using the TSF's RBG,***
    - ***zeroes,***
    - ***ones,***
    - ***a new value of a key,***
    - ***[assignment: some value that does not contain any CSP]],***
  - ***removal of power to the memory,***
  - ***destruction of reference to the key directly followed by a request for garbage collection];***
- ***For non-volatile memory [selection:***
  - ***that employs a wear-leveling algorithm, the destruction shall be executed by a [selection:***
    - ***single overwrite consisting of zeroes,***
    - ***single overwrite consisting of ones,***
    - ***overwrite with a new value of a key of the same size,***
    - ***single overwrite consisting of [assignment: some value that does not contain any CSP],***
    - ***block erase];***
  - ***that does not employ a wear-leveling algorithm, the destruction shall be executed by a [selection:***
    - ***[selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of zeros followed by a read-verify,***
    - ***[selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of ones followed by a read-verify, overwrite with a new value of a key of the same size followed by a read-verify,***
    - ***[selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of [assignment: some value that does not contain any CSP] followed by a read-verify,***
    - ***block erase]***

    and if the read-verification of the overwritten data fails, the process shall be repeated again up to [assignment: number of times to attempt overwrite] times, whereupon an error is returned.

]

] that meets the following: [*no standard*].

Application Note**:** *In the first selection, the ST Author is presented options for destroying a key based on the memory or storage technology where keys are stored within the TOE.*

If non-volatile memory is used to store keys, the ST Author selects whether the memory storage algorithm uses wear-leveling or not. Storage technologies or memory types that use wear-leveling are not required to perform a read verify. The selection for destruction includes block erase as an option, and this option applies only to flash memory. A block erase does not require a read verify, since the mappings of logical addresses to the erased memory locations are erased as well as the data itself.

Within the selections is the option to overwrite a disused key with a new value of a key. The intent is that a new value of a key (as specified in another SFR within the PP) can be used to "replace" an existing key.

If a selection for read verify is chosen, it should generate an audit record upon failures.

Several selections allow assignment of a 'value that does not contain any CSP'. This means that the TOE uses some other specified data not drawn from an RBG meeting FCS_RBG_EXT requirements, and not being any of the particular values listed as other selection options. The point of the phrase 'does not contain any CSP' is to ensure that the overwritten data is carefully selected, and not taken from a general 'pool' that might contain current or residual data that itself requires confidentiality protection.

Key destruction does not apply to the public component of asymmetric key pairs.

FCS_CKM.4(c) Cryptographic Key Destruction (General Hardware)

FCS_CKM.4.1(c) Refinement: The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [***selection:***

- ***For volatile memory, the destruction shall be executed by a [selection:***
    - ***single overwrite consisting of [selection:***
        - ***a pseudo-random pattern using the TSF's RBG,***
        - ***zeroes,***
        - ***ones,***
        - ***a new value of a key,***
        - ***[assignment: some value that does not contain any CSP]],***
    - ***removal of power to the memory,***
    - ***destruction of reference to the key directly followed by a request for garbage collection];***
- ***For non-volatile memory the destruction shall be executed by a [selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of [selection:***
    - ***a pseudo-random pattern using the TSF's RBG,***
    - ***zeroes,***
    - ***ones,***
    - ***a new value of a key of the same size,***

  - ***[assignment: some value that does not contain any CSP], block erase]***

]

] that meets the following: [*no standard*].

Application Note: *In the first selection, the ST Author is presented options for destroying disused cryptographic keys based on whether they are in volatile memory or non-volatile storage within the TOE. The selection of block erase for non-volatile storage applies only to flash memory. A block erase does not require a read-verify, since the reference to the memory location is erased as well as the data itself.*

Within the selections is the option to overwrite the memory location with a new value of a key. The intent is that a new value of a key (as specified in another SFR within the PP) can be used to "replace" an existing key.

Several selections allow assignment of a 'value that does not contain any CSP'. This means that the TOE uses some other specified data not drawn from an RBG meeting FCS_RBG_EXT requirements, and not being any of the particular values listed as other selection options. The point of the phrase 'does not contain any CSP' is to ensure that the overwritten data is carefully selected, and not taken from a general 'pool' that might contain current or residual data that itself requires confidentiality protection.

Key destruction does not apply to the public component of asymmetric key pairs.

FCS_CKM.4(d) Cryptographic Key Destruction (Software TOE, 3rd Party Storage)

FCS_CKM.4.1(d) Refinement: The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [***selection:***

- ***For volatile memory, the destruction shall be executed by a [selection:***
  - ***single overwrite consisting of [selection:***
    - ***a pseudo-random pattern using the TSF's RBG,***
    - ***zeroes,***
    - ***ones,***
    - ***a new value of a key,***
    - ***[assignment: some value that does not contain any CSP]],***
  - ***removal of power to the memory,***
  - ***destruction of reference to the key directly followed by a request for garbage collection];***
- ***For non-volatile storage that consists of the invocation of an interface provided by the underlying platform that [selection:***
  - ***logically addresses the storage location of the key and performs a [selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of [selection:***
    - ***a pseudo-random pattern using the TSF's RBG,***
    - ***zeroes,***
    - ***ones,***
    - ***a new value of a key,***
    - ***[assignment: some value that does not contain any CSP];***
  - ***instructs the underlying platform to destroy the abstraction that represents the key]***

]

that meets the following: [*no standard*].

Application Note: *The interface referenced in the requirement could take different forms, the most likely of which is an application programming interface to an OS kernel. There may be various levels of abstraction visible. For instance, in a given implementation the application may have access to the file system details and may be able to logically address specific memory locations. In another implementation the application may simply have a handle to a resource and can only ask the platform to delete the resource. The level of detail to which the TOE has access will be reflected in the TSS section of the ST.*

Several selections allow assignment of a 'value that does not contain any CSP'. This means that the TOE uses some other specified data not drawn from an RBG meeting FCS_RBG_EXT requirements, and not being any of the particular values listed as other selection options. The point of the phrase 'does not contain any CSP' is to ensure that the overwritten data is carefully selected, and not taken from a general 'pool' that might contain current or residual data that itself requires confidentiality protection.

Key destruction does not apply to the public component of asymmetric key pairs.

FCS_COP.1(a) Cryptographic Operation (Signature Verification)

FCS_COP.1.1(a) Refinement: The TSF shall perform [*cryptographic signature services (verification)*] in accordance with a [***selection:***

- ***RSA Digital Signature Algorithm with a key size (modulus) of [selection: 2048-bit, 3072-bit, 4096-bit];***
- ***Elliptic Curve Digital Signature Algorithm with a key size of 256 bits or greater***

]

that meet the following: [***selection:***

- ***FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1-v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3, for RSA schemes***
- ***FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" [selection: P-256, P-384, P-521]; ISO/IEC 14888-3, Section 6.4, for ECDSA schemes***

].

Application Note: *The selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(a) and quantum resistant recommendations. For example, SHA-256 should be chosen for 2048-bit RSA or ECC with P-256, SHA-384 should be chosen for 3072-bit RSA, 4096-bit RSA, or ECC with P-384, and SHA-512 should be chosen for ECC with P-521. The selection of the standard is made based on the algorithms selected.*

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

FCS_COP.1.1(b) Refinement: The TSF shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm [***selection: SHA-256, SHA-384, SHA-512***] ~~and cryptographic key sizes [*assignment: cryptographic key sizes*]~~ that meet the following: [*ISO/IEC 10118-3:2004*].

Application Note: *The selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(a) and quantum resistant recommendations. For example, SHA-256 should be chosen for 2048-bit RSA or ECC with P-256, SHA-384 should be chosen for 3072-bit RSA, 4096-bit RSA, or ECC with P-384, and SHA-512 should be chosen for ECC with P-521. The selection of the standard is made based on the algorithms selected.*

FCS_COP.1(c) Cryptographic Operation (Message Authentication)

FCS_COP.1.1(c) Refinement: The TSF shall perform cryptographic [*message authentication*] in accordance with a specified cryptographic algorithm **[selection: HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, CMAC-AES-128, CMAC-AES-256]** and cryptographic key sizes [***assignment: key size (in bits) used in [selection: HMAC, AES]***] that meet the following: [***selection: ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2", NIST SP 800-38B***].

Application Note: *If one or more HMAC algorithms are selected, the ST author selects "HMAC" in the second selection and "ISO/IEC 9797-2:2011, Section 7 'MAC Algorithm 2'" in the third selection. For the assignment, the key size [k] 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 and L2 = 256 where* $L2 \leq k \leq L1$.

If one or more CMAC algorithms are selected, the ST author selects "AES" in the second selection and "NIST SP 800-38B" in the third selection. For the assignment, the key size will fall into a range between 128 and 256.

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

FCS_COP.1.1(d) Refinement: The TSF shall perform [*key wrapping*] in accordance with a specified cryptographic algorithm [*AES*] **in the following modes [selection: KW, KWP, GCM, CCM]** and the cryptographic key size [***selection: 128 bits, 256 bits***] that meet the following: [*AES as specified in ISO/IEC 18033-3****, [selection: NIST SP 800-38F, ISO/IEC 19772, no other standards]***].

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.*

FCS_COP.1(e) Cryptographic Operation (Key Transport)

FCS_COP.1.1(e) Refinement: The TSF shall perform [*key transport*] in accordance with a specified cryptographic algorithm [*RSA in the following modes **[selection: KTS-OAEP, KTS-KEM-KWS]***] and the cryptographic key size [***selection: 2048 bits, 3072 bits***] that meet the following: [*NIST SP 800-56B, Revision 1*].

Application Note: *This requirement is used in the body of the ST if the ST author chooses to use key transport in the key chaining approach that is specified in FCS_KYC_EXT.2.*

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

FCS_COP.1.1(f) Refinement: The TSF shall perform [*data encryption and decryption*] in accordance with a specified cryptographic algorithm [*AES used in* ***[selection: CBC, GCM, XTS]*** *mode*] and cryptographic key sizes [***selection: 128 bits, 256 bits***] that meet the following: [*AES as specified in ISO /IEC 18033-3,* ***[selection: CBC as specified in ISO/IEC 10116, GCM as specified in ISO/IEC 19772, XTS as specified in IEEE 1619]***].

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.

FCS_COP.1(g) Cryptographic Operation (Key Encryption)

FCS_COP.1.1(g) Refinement: The TSF shall perform [*key encryption and decryption*] in accordance with a specified cryptographic algorithm [*AES used in* ***[selection: CBC, GCM]*** *mode*] and cryptographic key sizes [selection: 128 bits, 256 bits] that meet the following: [*AES as specified in ISO /IEC 18033-3,* ***[selection: CBC as specified in ISO/IEC 10116, GCM as specified in ISO/IEC 19772]***].

Application Note: *This requirement is used in the body of the ST if the ST author chooses to use AES encryption/decryption for protecting the keys as part of the key chaining approach that is specified in FCS_KYC_EXT.2.*

FCS_KDF_EXT.1 Cryptographic Key Derivation

FCS_KDF_EXT.1.1 The TSF shall accept [selection: a RNG generated submask as specified in FCS_RBG_EXT.1, a conditioned password submask, imported submask] to derive an intermediate key, as defined in [selection:

- NIST SP 800-108 [selection: KDF in Counter Mode, KDF in Feedback Mode, KDF in Double-Pipeline Iteration Mode],
- NIST SP 800-132],

using the keyed-hash functions specified in FCS_COP.1(c), such that the output is at least of equivalent security strength (in number of bits) to the BEV.

Application Note: *This requirement is used in the body of the ST if the ST author chooses to use key derivation in the key chaining approach that is specified in FCS_KYC_EXT.2.*

This requirement establishes acceptable methods for generating a new random key or an existing submask to create a new key along the key chain.

FCS_RBG_EXT.1 Random Bit Generation

FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with [selection: ISO/IEC 18031:2011, [*NIST SP 800-90A*]] using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from [selection:

- [*assignment: number of software-based sources*] software-based noise source(s),
- [*assignment: number of hardware-based sources*] hardware-based noise source(s)]

with a minimum of [selection: 128 bits, 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.

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-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_DRBG 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 to which the TSF is compliant.

In 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 DRBG and if the DRBG 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. Individual contributions to the entropy pool may be combined to provide the minimum amount of entropy as long as the Entropy Documentation demonstrates that entropy from each of these individual sources is generated independently.*

FCS_SMC_EXT.1 Submask Combining

FCS_SMC_EXT.1.1 The TSF shall combine submasks using the following method [selection: exclusive OR (XOR), SHA-256, SHA-384, SHA-512] to generate an [*intermediary key or DEK*].

Application Note: This requirement specifies the way that a product may combine the various submasks by using either an XOR or an approved SHA-hash. The approved hash functions are captured in FCS_COP.1(b).

B.2 Class: Protection of the TSF (FPT)

FPT_FUA_EXT.1 Firmware Update Authentication

FPT_FUA_EXT.1.1 The TSF shall authenticate the source of the firmware update using the digital signature algorithm specified in FCS_COP.1(a) using the RTU that contains [selection: the public key, hash value of the public key as specified in FCS_COP.1(b)].

FPT_FUA_EXT.1.2 The TSF shall only allow installation of update if the digital signature has been successfully verified as specified in FCS_COP.1(a).

FPT_FUA_EXT.1.3 The TSF shall only allow modification of the existing firmware after the successful validation of the digital signature, using a mechanism as described in FPT_TUD_EXT.1.2.

Application Note: *The firmware portion of the TOE (e.g., RTU (key store and the signature verification algorithm)) is expected to be stored in a write protected area on the TOE. It is expected that the firmware only be modifiable in a post-manufacturing state using the authenticated update mechanism described in FPT_FUA_EXT.1. The TSF is modifiable only by using the mechanisms specified in FPT_TUD_EXT.*

FPT_FUA_EXT.1.4 The TSF shall return an error code if any part of the firmware update process fails.

Application Note: *These requirements are for a SED in an operational state – not a drive in manufacturing.*

The authenticated firmware update mechanism employs digital signatures to ensure the authenticity of the firmware update image. The TSF provides a RTU that contains a signature verification algorithm and a key store that includes the public key needed to verify the signature on the update image. The key store in the RTU should include a public key used to verify the signature on an update image or a hash of the public key if a copy of the public key is provided with the update image. In the latter case, the update mechanism should hash the public key provided with the update image, and ensure that it matches a hash which appears in the key store before using the provided public key to verify the signature on the update image. If the hash of the public key is selected, the ST author may iterate the FCS_COP.1(b) requirement - to specify the hashing functions used.

The intent of this requirement is to specify that the authenticated update mechanism should ensure that the new image has been digitally signed; and that the digital signature can be *verified by using a public key before the update takes place. The requirement also specifies that the authenticated update mechanism only allows installation of updates when the digital signature has been successfully verified by the TSF.*

Appendix C: Extended Component Definitions

This appendix contains the definitions for the extended requirements that are used in the cPP, including those used in Appendices A and B.

Note that several of the extended requirements used for this cPP have dependencies on SFRs that are iterated in the cPP (e.g. FCS_COP.1(d)). The reader is advised that the SFR names for these dependencies may differ if the same extended components are used in other Protection Profiles.

C.1 Background and Scope

This document provides a definition for all of the extended components used in this cPP. These components are identified in the following table:

Table 4: Extended Components

Functional Class	Functional Components
Cryptographic Support (FCS)	FCS_CKM_EXT Cryptographic Key Management
	FCS_KDF_EXT Cryptographic Key Derivation
	FCS_KYC_EXT Key Chaining
	FCS_RBG_EXT Cryptographic Operation (Random Bit Generation)
	FCS_SMC_EXT Submask Combining
	FCS_SNI_EXT Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
	FCS_VAL_EXT Validation of Cryptographic Elements
User Data Protection (FDP)	FDP_DSK_EXT Protection of Data on Disk
Protection of the TSF (FPT)	FPT_FAC_EXT Firmware Access Control
	FPT_FUA_EXT Firmware Update Authentication
	FPT_KYP_EXT Key and Key Material Protection
	FPT_PWR_EXT Power Management
	FPT_RBP_EXT Rollback Protection
	FPT_TST_EXT TSF Testing
	FPT_TUD_EXT Trusted Update

Note that several of the extended components define dependencies on iterated Part 2 SFRs that are defined in this cPP. This definition mandates that these dependencies be included in a PP that claims the SFR but it does not mandate that the dependent SFRs are defined using the same iteration identifiers (e.g. inclusion of FCS_KDF_EXT.1 does not require the dependent SFR for keyed-hash message authentication to be identified specifically as FCS_COP.1(c), only that an FCS_COP.1 iteration exists and defines the same behavior as what this cPP defines as FCS_COP.1(c)).

C.2 Extended Component Definitions

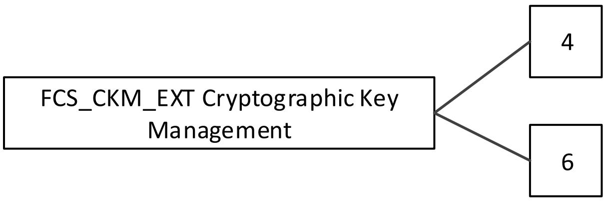
FCS_CKM_EXT Cryptographic Key Management

Family Behavior

Cryptographic keys must be managed throughout their life cycle. This family is intended to support that lifecycle and consequently defines requirements for the following activities: cryptographic key generation, cryptographic key distribution, cryptographic key access and cryptographic key destruction. This family should be included whenever there are functional requirements for the management of cryptographic keys.

The creation of this family is necessary because CC Part 2 provides the ability to specify the method of key destruction but does not define SFRs for the timing of key destruction or the ability to implement multiple key destruction methods.

Component Leveling



FCS_CKM_EXT.4, Key and Key Material Destruction, requires the TSF to specify circumstances when keys are destroyed (as opposed to the actual method of destruction, which is defined in CC Part 2 as FCS_CKM.4). The number 4 was chosen to reflect the similarity between the two SFRs.

FCS_CKM_EXT.6, Cryptographic Key Destruction Types, provides the TOE with the ability to select between multiple methods of key destruction.

Management: FCS_CKM_EXT.4

No specific management functions are identified.

Audit: FCS_CKM_EXT.4

There are no auditable events foreseen.

Management: FCS_CKM_EXT.4

No specific management functions are identified.

Audit: FCS_CKM_EXT.4

There are no auditable events foreseen.

FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction

Hierarchical to: No other components

Dependencies: No dependencies

FCS_CKM_EXT.4.1 The TSF shall destroy all keys and key material when no longer needed.

FCS_CKM_EXT.6 Cryptographic Key Destruction Types

Hierarchical to: No other components

Dependencies: FCS_CKM.4 Cryptographic Key Destruction

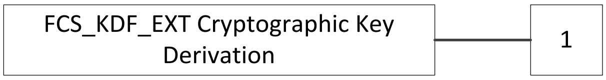
FCS_CKM_EXT.6.1 The TSF shall use [*assignment: one or more iterations of FCS_CKM.4 defined elsewhere in the Security Target*] key destruction methods.

FCS_KDF_EXT Cryptographic Key Derivation

Family Behavior

This family specifies the means by which an intermediate key is derived from a specified set of submasks.

Component Leveling



FCS_KDF_EXT.1, Cryptographic Key Derivation, requires the TSF to derive intermediate keys from submasks using the specified hash functions.

Management: FCS_KDF_EXT.1

No specific management functions are identified.

Audit: FCS_KDF_EXT.1

There are no auditable events foreseen.

FCS_KDF_EXT.1 Cryptographic Key Derivation

Hierarchical to: No other components

Dependencies: FCS_COP.1(c) Cryptographic Operation (Keyed Hash Algorithm)

FCS_KDF_EXT.1.1 The TSF shall accept [selection: a RNG generated submask as specified in FCS_RBG_EXT.1, a conditioned password submask, imported submask] to derive an intermediate key, as defined in [selection:

- NIST SP 800-108 [selection: KDF in Counter Mode, KDF in Feedback Mode, KDF in Double-Pipeline Iteration Mode],
- NIST SP 800-132],

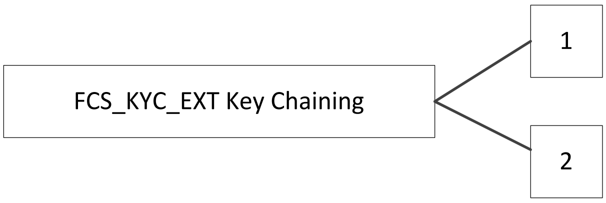
using the keyed-hash functions specified in FCS_COP.1(c), such that the output is at least of equivalent security strength (in number of bits) to the BEV.

FCS_KYC_EXT Key Chaining

Family Behavior

This family provides the specification to be used for using multiple layers of encryption keys to ultimately secure the protected data encrypted on the drive.

Component Leveling



FCS_KYC_EXT.1, Key Chaining (Initiator), requires the TSF to maintain a key chain for a BEV that is provided to a component external to the TOE.

FCS_KYC_EXT.2, Key Chaining (Recipient), requires the TSF to be able to accept a BEV that is then chained to a DEK used by the TSF through some method.

Note that this cPP does not include FCS_KYC_EXT.1; it is only included here to provide a complete definition of the FCS_KYC_EXT family.

Management: FCS_KYC_EXT.1

No specific management functions are identified.

Audit: FCS_KYC_EXT.1

There are no auditable events foreseen.

Management: FCS_KYC_EXT.2

No specific management functions are identified.

Audit: FCS_KYC_EXT.2

There are no auditable events foreseen.

FCS_KYC_EXT.1 Key Chaining (Initiator)

Hierarchical to:	No other components
Dependencies:	FCS_CKM.1(a) Cryptographic Key Generation (Asymmetric Keys), FCS_CKM.1(b) Cryptographic Operation (Symmetric Keys), FCS_COP.1(d) Cryptographic Operation (Key Wrapping), FCS_COP.1(e) Cryptographic Operation (Key Transport), FCS_COP.1(g) Cryptographic Operation (Key Encryption), FCS_SMC_EXT.1 Submask Combining, FCS_VAL_EXT.1 Validation

FCS_KYC_EXT.1.1 The TSF shall maintain a key chain of: [selection:

- one, using a submask as the BEV;
- intermediate keys generated by the TSF using the following method(s): [selection:
    - asymmetric key generation as specified in FCS_CKM.1(a),

    - symmetric key generation as specified in FCS_CKM.1(b)];
- intermediate keys originating from one or more submask(s) to the BEV using the following method(s): [selection:
    - key derivation as specified in FCS_KDF_EXT.1,
    - key wrapping as specified in FCS_COP.1(d),
    - key combining as specified in FCS_SMC_EXT.1,
    - key transport as specified in FCS_COP.1(e),
    - key encryption as specified in FCS_COP.1(g)]]

while maintaining an effective strength of [selection: 128 bits, 256 bits] for symmetric keys and an effective strength of [selection: not applicable, 112 bits, 128 bits, 192 bits, 256 bits] for asymmetric keys.

FCS_KYC_EXT.1.2 The TSF shall provide a [selection: 128 bit, 256 bit] BEV to [*assignment: one or more external entities*] [selection:

- after the TSF has successfully performed the validation process as specified in FCS_VAL_EXT.1,
- without validation taking place].

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).*

FCS_KYC_EXT.2 Key Chaining (Recipient)

Hierarchical to: No other components

Dependencies: No other components

FCS_KYC_EXT.2.1 The TSF shall accept a BEV of at least [selection: 128 bits, 256 bits] from [*assignment: one or more external entities*].

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): [selection:

- asymmetric key generation as specified in FCS_CKM.1(a),
- symmetric key generation as specified in FCS_CKM.1(b),
- key derivation as specified in FCS_KDF_EXT.1,
- key wrapping as specified in FCS_COP.1(d),
- key combining as specified in FCS_SMC_EXT.1,
- key transport as specified in FCS_COP.1(e),
- key encryption as specified in FCS_COP.1(g)]

while maintaining an effective strength of [selection: 128 bits, 256 bits] for symmetric keys and an effective strength of [selection: not applicable, 112 bits, 128 bits, 192 bits, 256 bits] for asymmetric keys.

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 one (e.g., using the BEV as a key encrypting key (KEK)) 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).*

FCS_RBG_EXT Random Bit Generation

Family Behavior

Components in this family address the requirements for random bit/number generation. This is a new family defined for the FCS class.

Component Leveling

FCS_RBG_EXT.1, Random Bit Generation, requires random bit generation to be performed in accordance with selected standards and seeded by an entropy source.

Management: FCS_RBG_EXT.1

No specific management functions are identified.

Audit: FCS_RBG_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Failure of the randomization process

FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

Hierarchical to: No other components

Dependencies: FCS_COP.1(b) Cryptographic Operation (Hash Algorithm),
FCS_COP.1(c) Cryptographic Operation (Keyed Hash Algorithm)

FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with [selection: ISO/IEC 18031:2011, [*assignment: other RBG standards*]] using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from [selection:

- [*assignment: number of software-based sources*] software-based noise source(s),
- [*assignment: number of hardware-based sources*] hardware-based noise source(s)]

with a minimum of [selection: 128 bits, 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.

Application Note: *ISO/IEC 18031:2011contains three 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-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.*

FCS_SMC_EXT Submask Combining

Family Behavior

This family specifies the means by which submasks are combined, if the TOE supports more than one submask being used to derive or protect the BEV.

Component Leveling

FCS_SMC_EXT.1, Submask Combining, requires the TSF to combine the submasks in a predictable fashion.

Management: FCS_SMC_EXT.1

No specific management functions are identified.

Audit: FCS_SMC_EXT.1

There are no auditable events foreseen.

FCS_SMC_EXT.1 Submask Combining

Hierarchical to: No other components

Dependencies: FCS_COP.1(b) Cryptographic Operation (Hash Algorithm)

FCS_SMC_EXT.1.1 The TSF shall combine submasks using the following method [selection: exclusive OR (XOR), SHA-256, SHA-384, SHA-512] to generate an [*assignment: types of keys*].

FCS_SNI_EXT Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)

Family Behavior

This family ensures that salts, nonces, and IVs are well formed.

Component Leveling

FCS_SNI_EXT.1, Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation), requires the generation of salts, nonces, and IVs to be used by the cryptographic components of the TOE to be performed in the specified manner.

Management: FCS_SNI_EXT.1

No specific management functions are identified.

Audit: FCS_SNI_EXT.1

There are no auditable events foreseen.

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

Hierarchical to: No other components

Dependencies: FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

FCS_SNI_EXT.1.1 The TSF shall [selection: use no salts, use salts that are generated by a [selection: DRBG as specified in FCS_RBG_EXT.1, DRBG provided by the host platform]].

FCS_SNI_EXT.1.2 The TSF shall use [selection: no nonces, unique nonces with a minimum size of [*64*] bits].

FCS_SNI_EXT.1.3 The TSF shall create IVs in the following manner [selection:

- 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].

FCS_VAL_EXT Validation of Cryptographic Elements

Family Behavior

This family specifies the means by which submasks and/or BEVs are determined to be valid prior to their use.

Component Leveling

FCS_VAL_EXT.1, Validation, requires the TSF to validate submasks and BEVs by one or more of the specified methods.

Management: FCS_VAL_EXT.1

No specific management functions are identified.

Audit: FCS_VAL_EXT.1

There are no auditable events foreseen.

FCS_VAL_EXT.1 Validation

Hierarchical to: No other components

Dependencies: FCS_COP.1(b) Cryptographic Operation (Hash Algorithm), FCS_COP.1(c) Cryptographic Operation (Keyed Hash Algorithm), FCS_COP.1(d) Cryptographic Operation (Key Wrapping), FCS_COP.1(f) Cryptographic Operation (AES Data Encryption/Decryption)

FCS_VAL_EXT.1.1 The TSF shall perform validation of the [selection: submask, intermediate key, BEV] using the following method(s): [selection:

- key wrap as specified in FCS_COP.1(d);
- hash the [selection: submask, intermediate key, BEV] as specified in [selection: FCS_COP.1(b), FCS_COP.1(c)] and compare it to a stored hashed [selection: submask, intermediate key, BEV];
- decrypt a known value using the [selection: submask, intermediate key, BEV] as specified in FCS_COP.1(f) and compare it against a stored known value]

FCS_VAL_EXT.1.2 The TSF shall require validation of the [selection: submask, intermediate key, BEV] prior to [*assignment: activity requiring validation*].

FCS_VAL_EXT.1.3 The TSF shall [selection:

- perform a key sanitization of the DEK upon a [selection: configurable number, [*assignment: ST author specified number*]] of consecutive failed validation attempts,
- institute a delay such that only [*assignment: ST author specified number of attempts*] can be made within a 24 hour period,
- block validation after [*assignment: ST author specified number of attempts*] of consecutive failed validation attempts,
- require power cycle/reset the TOE after [*assignment: ST author specified number of attempts*] of consecutive failed validation attempts].

FDP_DSK_EXT Protection of Data on Disk

Family Behavior

This family specifies methods for ensuring that data residing in permanent storage on disk is not subject to unauthorized disclosure.

Component Leveling

FDP_DSK_EXT.1, Validation, requires the TSF to validate submasks and BEVs by one or more of the specified methods.

Management: FDP_DSK_EXT.1

No specific management functions are identified.

Audit: FDP_DSK_EXT.1

There are no auditable events foreseen.

FDP_DSK_EXT.1 Protection of Data on Disk

Hierarchical to:	No other components
Dependencies:	FCS_COP.1(f) Cryptographic Operation (AES Data Encryption/Decryption)

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.

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.*

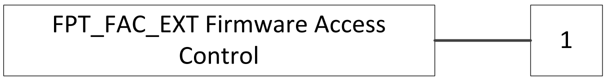
The cryptographic functions that perform the encryption/decryption of the data may be provided by the Operational Environment. Note that if this is the case, it is assumed that the environmental implementation of AES is consistent with the behavior described in FCS_COP.1(f). If the TOE provides the cryptographic functions to encrypt/decrypt the data, the ST author includes FCS_COP.1(f) as defined in Appendix A in the main body of the ST.

FPT_FAC_EXT Firmware Access Control

Family Behavior

This family requires that a valid authentication factor be provided prior to the TSF authorizing an update of its firmware.

Component Leveling



FPT_FAC_EXT.1, Firmware Access Control, requires the TSF to require an authentication factor prior to allowing a firmware update to be performed.

Management: FPT_FAC_EXT.1

The following actions could be considered for the management functions in FMT:

a) management of the password used to authorize the firmware update

Audit: FPT_FAC_EXT.1

There are no auditable events foreseen.

FPT_FAC_EXT.1 Firmware Access Control

Hierarchical to: No other components

Dependencies: No dependencies

FPT_FAC_EXT.1.1 The TSF shall require [selection: a password, a known unique value printed on the device, a privileged user action] before the firmware update proceeds.

FPT_FUA_EXT Firmware Update Authentication

Family Behavior

This family requires that firmware updates be authenticated by the TSF prior to being applied.

Component Leveling

FPT_FUA_EXT.1, Firmware Update Authentication, requires the TSF to authenticate firmware updates using a specified method.

Management: FPT_FUA_EXT.1

No specific management functions are identified.

Audit: FPT_FUA_EXT.1

There are no auditable events foreseen.

FPT_FUA_EXT.1 Firmware Update Authentication

Hierarchical to: No other components

Dependencies: FCS_COP.1(a) Cryptographic Operation (Signature Verification), FCS_COP.1(b) Cryptographic Operation (Hash Algorithm)

FPT_FUA_EXT.1.1 The TSF shall authenticate the source of the firmware update using the digital signature algorithm specified in FCS_COP.1(a) using the RTU that contains [selection: the public key, hash value of the public key as specified in FCS_COP.1(b)].

FPT_FUA_EXT.1.2 The TSF shall only allow installation of firmware updates if the digital signature has been successfully verified as specified in FCS_COP.1(a).

FPT_FUA_EXT.1.3 The TSF shall only allow modification of the existing firmware after the successful validation of the digital signature, using a mechanism as described in FPT_TUD_EXT.1.2.

FPT_FUA_EXT.1.4 The TSF shall return an error code if any part of the firmware update process fails.

FPT_KYP_EXT Key and Key Material Protection

Family Behavior

This family requires that key and key material be protected if and when written to non-volatile storage.

Component Leveling

FPT_KYP_EXT.1, Protection of Key and Key Material, requires the TSF to ensure that no plaintext key or key material are written to non-volatile storage.

Management: FPT_KYP_EXT.1

No specific management functions are identified.

Audit: FPT_KYP_EXT.1

There are no auditable events foreseen.

FPT_KYP_EXT.1 Protection of Key and Key Material

Hierarchical to: No other components

Dependencies: FCS_COP.1(d) Cryptographic Operation (Key Wrapping),
FCS_COP.1(e) Cryptographic Operation (Key Transport),
FCS_COP.1(g) Cryptographic Operation (Key Encryption),
FCS_KYC_EXT.1 Key Chaining (Initiator),

FCS_KYC_EXT.2 Key Chaining (Recipient),
FCS_SMC_EXT.1 Submask Combining

FPT_KYP_EXT.1.1 The TSF shall [selection:

- not store keys in non-volatile memory
- 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)
- only store plaintext keys that meet any one of the following criteria [selection:
    - the plaintext key is not part of the key chain as specified in FCS_KYC_EXT.2,
    - the plaintext key will no longer provide access to the encrypted data after initial provisioning,
    - the plaintext key is a key split that is combined as specified in FCS_SMC_EXT.1, and the other half of the key split is [selection:
        - wrapped as specified in FCS_COP.1(d),
        - encrypted as specified in FCS_COP.1(g) or FCS_COP.1(e),
        - derived and not stored in non-volatile memory],
    - the non-volatile memory the key is stored on is located in an external storage device for use as an authorization factor,
    - the plaintext key is [selection:
        - used to wrap a key as specified in FCS_COP.1(d),
        - used to encrypt a key as specified in FCS_COP.1(g) or FCS_COP.1(e)]

      that is already [selection:
        - wrapped as specified in FCS_COP.1(d),
        - encrypted as specified in FCS_COP.1(g) or FCS_COP.1(e)]]].

FPT_PWR_EXT Power Management

Family Behavior

This family defines secure behavior of the TSF when the TOE supports multiple power saving states. The use of Compliant power saving states (i.e. power saving states that purge security-relevant data upon entry) is essential for ensuring that state transitions cannot be used as attack vectors to bypass TOE self-protection mechanisms.

Component Leveling

FPT_PWR_EXT.1, Power Saving States, defines the Compliant power saving states that are implemented by the TSF.

FPT_PWR_EXT.2, Timing of Power Saving States, describes the situations that cause Compliant power saving states to be entered.

Management: FPT_PWR_EXT.1

The following actions could be considered for the management functions in FMT:

- Enable or disable the use of individual power saving states
- Specify one or more power saving state configurations

Audit: FPT_PWR_EXT.1

There are no auditable events foreseen.

Management: FPT_PWR_EXT.2

There are no management activities foreseen.

Audit: FPT_PWR_EXT.2

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Transition of the TSF into different power saving states

FPT_PWR_EXT.1 Authorization Factor Acquisition

Hierarchical to: No other components

Dependencies: No dependencies

FPT_PWR_EXT.1.1 The TSF shall define the following Compliant power saving states: [selection: choose at least one of: S3, S4, G2(S5), G3, D0, D1, D2, D3 [*assignment: other power saving states*]].

FPT_PWR_EXT.2 Authorization Factor Acquisition

Hierarchical to: No other components

Dependencies: FPT_PWR_EXT.1 Power Saving States

FPT_PWR_EXT.2.1 For each Compliant power saving state defined in FPT_PWR_EXT.1.1, the TSF shall enter the Compliant power saving state when the following conditions occur: user-initiated request, [selection: shutdown, user inactivity, request initiated by remote management system, [*assignment: other conditions*], no other conditions].

FPT_RBP_EXT Rollback Protection

Family Behavior

This family requires that the TSF protects against rollbacks or downgrades to its firmware.

Component Leveling

FPT_RBP_EXT.1, Rollback Protection, requires the TSF to detect and prevent unauthorized rollback.

Management: FPT_KYP_EXT.1

No specific management functions are identified.

Audit: FPT_KYP_EXT.1

There are no auditable events foreseen.

FPT_RBP_EXT.1 Rollback Protection

Hierarchical to: No other components

Dependencies: No dependencies

FPT_RBP_EXT.1.1 The TSF shall verify that the new firmware package is not downgrading to a lower security version number by [*assignment: method of verifying the security version number is the same as or higher than the currently installed version*].

FPT_RBP_EXT.1.2 The TSF shall generate and return an error code if the attempted firmware update package is detected to be an invalid version.

FPT_TST_EXT TSF Testing

Family Behavior

Components in this family address the requirements for self-testing the TSF for selected correct operation.

Component Leveling

FPT_TST_EXT.1, TSF Testing, requires a suite of self-tests to be run during initial start-up in order to demonstrate correct operation of the TSF.

Management: FPT_TST_EXT.1

No specific management functions are identified.

Audit: FPT_TST_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Indication that TSF self-test was completed

FPT_TST_EXT.1 TSF Testing

Hierarchical to: No other components

Dependencies: No dependencies

FPT_TST_EXT.1.1 The TSF shall run a suite of the following self-tests [selection: during initial start-up (on power on), periodically during normal operation, at the request of the authorized user, at the conditions [*assignment: conditions under which self-tests should occur*]] to demonstrate the correct operation of the TSF: [*assignment: list of self-tests run by the TSF*].

FPT_TUD_EXT Trusted Update

Family Behavior

Components in this family address the requirements for updating the TOE firmware and/or software.

Component Leveling

FPT_TUD_EXT.1, Trusted Update, requires the capability to be provided to update the TOE firmware and software, including the ability to verify the updates prior to installation.

Management: FPT_TUD_EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to update the TOE and to verify the updates

Audit: FPT_TUD_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Initiation of the update process.
- Any failure to verify the integrity of the update

FPT_TUD_EXT.1 Trusted Update

Hierarchical to: No other components

Dependencies: FCS_COP.1(a) Cryptographic Operation (Signature Verification), FCS_COP.1(b) Cryptographic Operation (Hash Algorithm)

FPT_TUD_EXT.1.1 The TSF shall provide [*assignment: list of subjects*] the ability to query the current version of the TOE software/firmware.

FPT_TUD_EXT.1.2 The TSF shall provide [*assignment: list of subjects*] 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 [selection: digital signature, published hash] by the manufacturer prior to installing those updates.

Appendix D: Entropy Documentation and Assessment

This is an optional appendix in the cPP, and only applies if the TOE is providing the Random Bit Generator

This appendix describes the required supplementary information for each entropy source used by the TOE.

The documentation of the entropy source(s) should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide sufficient entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS in the public facing ST.

D.1 Design Description

Documentation shall include the design of each entropy source as a whole, including the interaction of all entropy source components. Any information that can be shared regarding the design should also be included for any third-party entropy sources that are included in the product.

The documentation will describe the operation of the entropy source to include, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the entropy comes from, where the entropy output is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

If implemented, the design description shall include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on shall be included.

D.2 Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source delivering sufficient entropy for the uses made of the RBG output (by this particular TOE). This argument will include a description of the expected min-entropy rate (i.e. the minimum entropy (in bits) per bit or byte of source data) and explain that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

The amount of information necessary to justify the expected min-entropy rate depends on the type of entropy source included in the product.

For developer provided entropy sources, in order to justify the min-entropy rate, it is expected that a large number of raw source bits will be collected, statistical tests will be performed, and the min-entropy rate determined from the statistical tests. While no particular statistical tests are required at this time, it is expected that some testing is necessary in order to determine the amount of min-entropy in each output.

For third party provided entropy sources, in which the TOE vendor has limited access to the design and raw entropy data of the source, the documentation will indicate an estimate of the amount of min-entropy obtained from this third-party source. It is acceptable for the vendor to "assume" an amount of min-entropy, however, this assumption must be clearly stated in the documentation provided. In particular, the min-entropy estimate must be specified and the assumption included in the ST.

Regardless of type of entropy source, the justification will also include how the DRBG is initialized with the entropy stated in the ST, for example by verifying that the min-entropy rate is multiplied by the amount of source data used to seed the DRBG or that the rate of entropy expected based on the amount of source data is explicitly stated and compared to the statistical rate. If the amount of source data used to seed the DRBG is not clear or the calculated rate is not explicitly related to the seed, the documentation will not be considered complete.

The entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

D.3 Operating Conditions

The entropy rate may be affected by conditions outside the control of the entropy source itself. For example, voltage, frequency, temperature, and elapsed time after power-on are just a few of the factors that may affect the operation of the entropy source. As such, documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. Similarly, documentation shall describe the conditions under which the entropy source is no longer guaranteed to provide sufficient entropy. Methods used to detect failure or degradation of the source shall be included.

D.4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, TOE behavior upon entropy source failure, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

Appendix E: Key Management Description

The documentation of the product's encryption key management should be detailed enough that, after reading, the evaluator will thoroughly understand the product's key management and how it meets the requirements to ensure the keys are adequately protected. This documentation should include an essay and diagram(s). This documentation is not required to be part of the TSS - it can be submitted as a separate document and marked as developer proprietary.

The following topics may not apply to all products, so a note as to why the details do not apply should be included.

Essay:

The essay will provide the following information for all keys in the key chain:

- The purpose of the key
- If the key is stored in non-volatile memory
- How and when the key is protected
- How and when the key is derived
- The strength of the key
- When or if the key would be no longer needed, along with a justification.

The essay will also describe the following topics:

- The process for validation shall be described, noting what value(s) is used for validation and the process used to perform the validation. It shall describe how this process ensures no keys in the key chain are weakened or exposed by this process. It shall describe the method used to limit the number of consecutively failed authorization attempts.
- The authorization process that leads to the ultimate release of the DEK. This section shall detail the key chain used by the product. It shall describe which keys are used in the protection of the DEK and how they meet the derivation or key wrap. It shall also include any values that add into that key chain or interact with the key chain and the protections that ensure those values do not weaken or expose the overall strength of the key chain.
- The diagram and essay will clearly illustrate 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.
- 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 description should also include the data flow from the device's host interface to the device's persistent media storing the data, 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 description should be detailed enough to verify all platforms to ensure that when the

user enables encryption, the product encrypts all hard storage devices. It should also describe the platform's boot initialization, the encryption initialization process, and at what moment the product enables the encryption.
- The process for destroying keys when they are no longer needed by including the type of storage location of all keys and the destruction method for that storage.

Diagram:

- The diagram will include all keys from the BEV to the DEK and any keys or values that contribute into the chain. It must list the cryptographic strength of each key and illustrate how each key along the chain is protected with either Key Derivation or Key Wrapping (from the allowed options). The diagram should indicate the input used to derive or unwrap each key in the chain.
- 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 for 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 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.

1 Appendix F: Glossary

Term	Meaning
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. This value 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 shut down.

Term	Meaning
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.
Root of Trust for Update	A type of Root of Trust for Verification that verifies the integrity and authenticity of an update payload before initiating the update process.
Root of Trust for Verification	A Root of Trust (RoT) that verifies an integrity measurement against a policy.
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]

1 See [CC1] for other Common Criteria abbreviations and terminology.

1 Appendix G: Acronyms

Acronym	Meaning
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
RoT	Root of Trust
RSA	Rivest Shamir Adleman Algorithm
RTU	Root of Trust for Update
RTV	Root of Trust for Verification
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
XOR	Exclusive or

XTS	XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing

Appendix H: References

National Institute of Standards and Technology (NIST) Special Publication 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, National Institute of Standards and Technology, December 2012.

National Institute of Standards and Technology (NIST) Special Publication 800-56B, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, National Institute of Standards and Technology, August 2009.

National Institute of Standards and Technology (NIST) Special Publication 800-88 Revision 1, Guidelines for Media Sanitization, National Institute of Standards and Technology, December 2014.

National Institute of Standards and Technology (NIST) Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, National Institute of Standards and Technology, January 2012.

National Institute of Standards and Technology (NIST) Special Publication 800-132, Recommendation for Password-Based Key Derivation Part 1: Storage Applications, National Institute of Standards and Technology, December 2010.

Federal Information Processing Standard Publication (FIPS-PUB) 186-4, Digital Signature Standard (DSS), National Institute of Standards and Technology, July 2013.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 9796-2:2010 (3rd edition), Information technology — Security techniques — Digital signature schemes giving message recovery, International Organization for Standardization/International Electrotechnical Commission, 2010.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 9797-2:2011 (2nd edition), Information technology — Security techniques — Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function, International Organization for Standardization/International Electrotechnical Commission, 2011.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 10116:2006 (3rd edition), Information technology — Security techniques — Modes of operation for an n-bit block cipher, International Organization for Standardization/International Electrotechnical Commission, 2006.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 10118-3:2004 (3rd edition), Information technology — Security techniques — Hash-functions – Part 3: Dedicated hash-functions, International Organization for Standardization/International Electrotechnical Commission, 2004.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 14888-3:2006 (2nd edition), Information technology — Security techniques —Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms, International Organization for Standardization/International Electrotechnical Commission, 2006.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 18031:2011 (2nd edition), Information technology — Security techniques — Random bit generation, International Organization for Standardization/International Electrotechnical Commission, 2011.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 18033-3:2011 (3rd edition), Information technology — Security techniques — Encryption algorithms – Part 3: Block ciphers, International Organization for Standardization/International Electrotechnical Commission, 2011.

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 19772:2009, Information technology — Security techniques Authenticated encryption, International Organization for Standardization/International Electrotechnical Commission, 2009.