collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition

Version 0.2

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 – Authorization Acquisition. 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 the *Supporting Document (Mandatory Technical Document) Full Drive Encryption: Authorization Acquisition September 2014.*

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 [CC]. In particular, a cPP defines the IT security requirements of a generic type of TOE and specifies the functional and assurance security measures to be offered by that TOE to meet stated requirements [CC1, Section C.1].

0.3 Intended Readership

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

0.4 Related Documents

Common Criteria[1]

[CC1]	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, CCMB-2012-09-001, Version 3.1 Revision 4, September 2012.
[CC2]	Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, CCMB-2012-09-002, Version 3.1 Revision 4, September 2012.

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

[CC3] Common Criteria for Information Technology Security Evaluation,
Part 3: Security Assurance Components,
CCMB-2012-09-003, Version 3.1 Revision 4, September 2012.

[CEM] Common Methodology for Information Technology Security Evaluation,
Evaluation Methodology,
CCMB-2012-09-004, Version 3.1, Revision 4, September 2012.

[SD] Supporting Document (Mandatory Technical Document), Full Drive Encryption: Authorization Acquisition September 2014.

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

Contents

Acknowledgements 2

0. Preface 3
   - 0.1 Objectives of Document 3
   - 0.2 Scope of Document 3
   - 0.3 Intended Readership 3
   - 0.4 Related Documents 3
   - 0.5 Revision History 5

1. PP Introduction 9
   - 1.1 PP Reference Identification 9
   - 1.2 Introduction to the FDE Collaborative Protection Profiles (cPPs) Effort 9
   - 1.3 Implementations 10
   - 1.4 Target of Evaluation (TOE) Overview 11
     - 1.4.1 Authorization Acquisition Introduction 11
     - 1.4.2 Authorization Acquisition Security Capabilities 12
     - 1.4.3 Interface/Boundary 12
   - 1.5 The TOE and the Operational/Pre-Boot Environments 12
   - 1.6 Functionality Deferred until Next cPP Version 13

2. CC Conformance Claims 14

3. Security Problem Definition 15
   - 3.1 Threats 15
   - 3.2 Assumptions 17
   - 3.3 Organizational Security Policy 18

4. Security Objectives 19
   - 4.1 Security Objectives for the Operational Environment 19

5. Security Functional Requirements 21
   - 5.1 Class: Cryptographic Support (FCS) 21
     - FCS_AFA_EXT.1 Authorization Factor Acquisition 21
     - FCS_KYC_EXT.1 (Key Chaining) 22
     - FCS_PCC_EXT.1 Cryptographic Password Construct and Conditioning 23
     - FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction 23
     - FCS_CKM.4 Cryptographic key destruction 23
   - 5.2 Specification of Management Functions (FMT_SMF) 25
     - FMT_SMF.1 Specification of Management Functions 25
   - 5.3 Class: Protection of the TSF (FPT) 25
     - FPT_KYP_EXT.1 Extended: Protection of Key and Key Material 25
     - FPT_TUD_EXT.1 Trusted Update 25

6. Security Assurance Requirements 27
   - 6.1 ASE: Security Target 27
   - 6.2 ADV: Development 28
     - 6.2.1 Basic Functional Specification (ADV_FSP.1) 28
   - 6.3 AGD: Guidance Documentation 28
     - 6.3.1 Operational User Guidance (AGD_OPE.1) 29
     - 6.3.2 Preparative Procedures (AGD_PRE.1) 29
   - 6.4 Class ALC: Life-cycle Support 29
     - 6.4.1 Labelling of the TOE (ALC_CMC.1) 29
     - 6.4.2 TOE CM Coverage (ALC_CMS.1) 29
   - 6.5 Class ATE: Tests 30
     - 6.5.1 Independent Testing – Conformance (ATE_IND.1) 30
   - 6.6 Class AVA: Vulnerability Assessment 30
     - 6.6.1 Vulnerability Survey (AVA_VAN.1) 30

A. Optional Requirements 31
A.1 Class: Cryptographic Support (FCS) 31
FCS_CKM.1 Cryptographic Key Generation(Asymmetric Keys) 31
FCS_SMC_EXT.1 Submask Combining 32
FCS_VAL.EXT.1 Validation 32
FCS_COP.1(a) Cryptographic Operation (Signature Verification) 33
FCS_COP.1(b) Cryptographic operation (Hash Algorithm) 34
FCS_COP.1(c) Cryptographic operation (Keyed Hash Algorithm) 34
A.2 Class: Protection of the TSF (FPT) 34
FPT_TST_EXT.1 Extended: TSF Testing 34
B. Selection-Based Requirements 35
B.1 Class: Cryptographic Support (FCS) 35
FCS_COP.1(f) Cryptographic operation (AES Data Encryption/Decryption) 35
FCS_COP.1(e) Cryptographic operation (Key Wrapping) 35
FCS_KDF_EXT.1 Cryptographic Key Derivation 35
FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation) 36
C. Extended Component Definitions 37
C.1 Background and Scope 37
C.2 Extended Component Definitions 37
FCS_KYC_EXT.1 Key Chaining 39
FCS_PCC_EXT.1 Cryptographic Password Construction and Conditioning 40
FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction 41
FCS_SNI_EXT.1 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) 42
FPT_KYP_EXT.1 Extended: Protection of Key and Key Material 43
FCS_SMC_EXT.1 Submask Combining 44
FCS_VAL_EXT.1 Validation 45
FCS_KDF_EXT.1 Cryptographic Key Derivation 47
D. Entropy Documentation And Assessment 49
D.1 Design Description 49
D.2 Entropy Justification 49
D.3 Operating Conditions 50
D.4 Health Testing 50
E. Key Management Description 51
F. Glossary 53
G. Acronyms 55

Figures / Tables

Figure 1: FDE Components ... 9
Table 1: Examples of cPP Implementations ... 10
Figure 2: Authorization Acquisition Details ... 11
Figure 3: Operational Environment ... 13
Table 2 TOE Security Functional Requirements ... 21
Table 3: Security Assurance Requirements ... 27

1. PP Introduction

1.1 PP Reference Identification

PP Reference: collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition

PP Version: 0.2

PP Date: September 5, 2014

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. 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 of this cPP.

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.

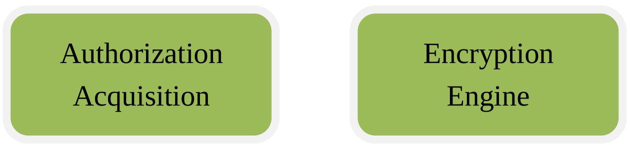


Figure 1: FDE Components

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 data encryption key (DEK).

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 Authorization Acquisition, and the Security Problem Definition describes the assumptions made about the operating environment and the threats to the AA that the cPP requirements address.

1.3 Implementations

Full Drive Encryption solutions vary with implementation and vendor combinations.

Therefore, vendors will evaluate products that provide both components of the Full Drive Encryption Solution (AA and EE) against both cPPs. 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
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, cryptographic co-processor) and software

1.4 Target of Evaluation (TOE) Overview

The Target of Evaluation (TOE) for this cPP (Authorization Acquisition) may be either a Host software solution that manages a HW Encryption Engine (e.g. a SED) or as part of a combined evaluation of this cPP and the Encryption Engine cPP for a vendor that is providing a solution that includes both components.

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

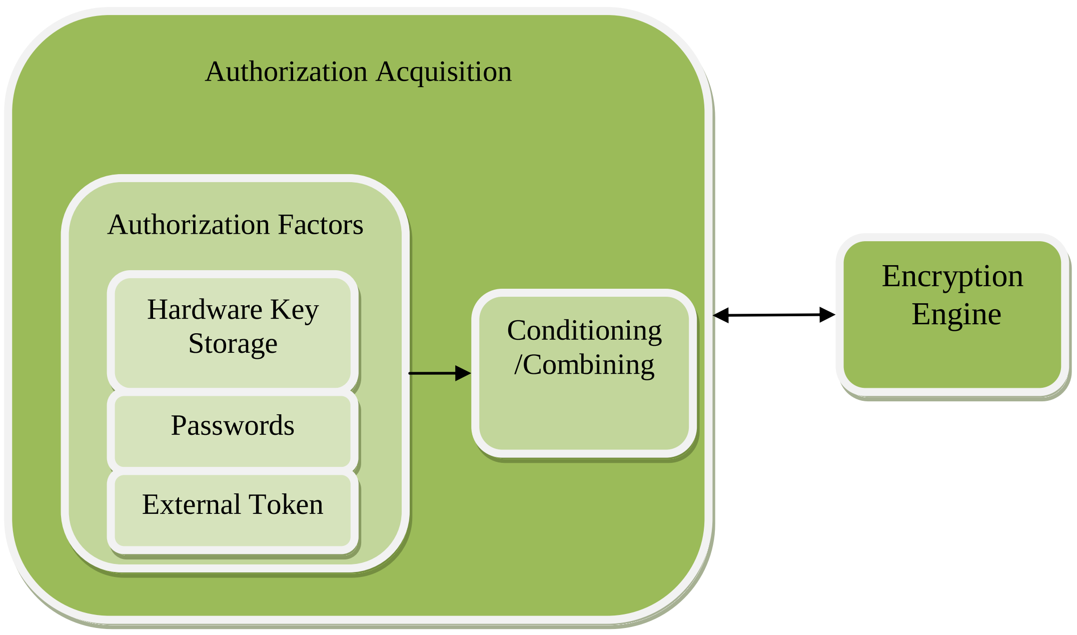


Figure 2: Authorization Acquisition Details

1.4.1 Authorization Acquisition Introduction

The Authorization Acquisition sends a Border Encryption Value (BEV), which could be a Key Encryption Key (KEK), a Key Releasing Key (KRK), or some other type of key to the Encryption Engine. The EE does not have to use this value directly as the key to decrypt or release the DEK. It may use it as part of a scheme that uses other intermediate keys to eventually protect the DEK. A KEK wraps 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. Figure 2 illustrates the components within AA and its relationship with EE.

Authorization factors may be unique to individual users or may be used by a group of individuals. In other words, the EE requires authorization factors from the AA to establish that the possessor of the authorization factors belongs to the community of users authorized to access information stored on the storage device (and does not require specific user authorization). Examples of authorization factors include, but are not limited to, passwords, passphrases, or randomly generated values stored on USB tokens or a pin to release a key on hardware storage media such as a Trusted Platform Module (TPM).

1.4.2 Authorization Acquisition Security Capabilities

The AA collects authorization factors which the EE uses to access data on the storage device and perform a variety of management functions. Depending on the type of authorization factor, the AA may condition them further. For example, it may apply an approved password-based key derivation function (e.g. PBKDF2) on passwords. An external token containing a randomly generated value of sufficient strength may require no further conditioning on the authorization factors. The AA may then combine one or more authorization factors in such a way that maintains the strength of both factors.

The AA serves as the main management interface to the EE. However, the EE may also offer management functionality. The requirements in the EE cPP address how the EE should handle these features. The management functionality should include the ability to send commands to the EE such as changing a DEK, setting up new users, managing KEKs and other intermediate keys, and performing a key sanitization (e.g. overwrite of the DEK). It may also forward commands that partition the drive for use by multiple users. However, this document defers the management of partitions and assumes that administrators and users will only provision and manage the data on whole drives.

1.4.3 Interface/Boundary

The interface and boundary between the AA and the EE will vary based on the implementation. If one vendor provides the entire FDE solution, then it is may choose to not implement an interface between the AA and EE components. If a vendor are provides a solution for one of the components, then the assumptions below state that the channel between the two components is sufficiently secure. Although standards and specifications exist for the interface between AA and EE components, the cPP does not require vendors to follow the standards in this version.

1.5 The TOE and the Operational/Pre-Boot Environments

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

In the Operating System environment, the Authorization Acquisition has the full range of services available from the operating system (OS), 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 AA TOE may include or leverage features and functions within the operational environment.

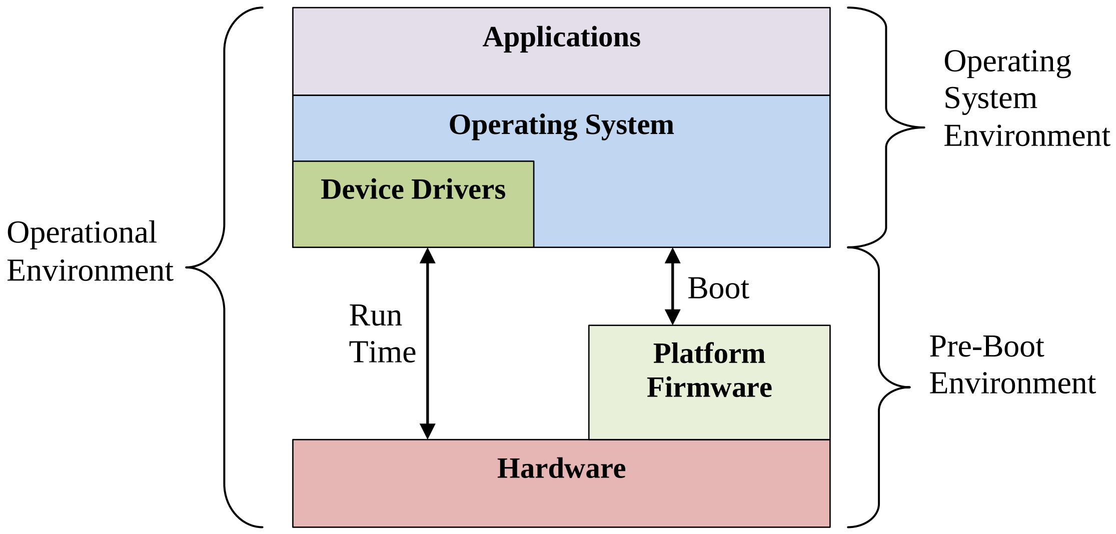


Figure 3: Operational Environment

1.6 Functionality Deferred until Next cPP Version

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, remote management, and power management (requirements for power state protection).

2. CC Conformance Claims

As defined by the references [CC1], [CC2] and [CC3], this cPP conforms to the requirements of Common Criteria v3.1, Revision 4. This cPP is conformant to CC 3.1, CC Part 2 extended and CC Part 3 conformant. Extended component definitions can be found in **Extended Component Definitions**

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 PP.

In order to be conformant to this cPP, a TOE must demonstrate *Exact Compliance*. *Exact Compliance*, as a subset of *Strict Compliance* as defined by the CC, is defined as the ST containing all of the requirements in section 5 of the 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.

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.

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 storage device encryption (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).

> [FCS_AFA.EXT.1.1, FCS_KYC_EXT.1.1, FCS_KYC_EXT.1.2, FCS_PCC_EXT.1, FMT_SMF.1.1]
>
> Rationale: FCS_KYC_EXT.1.2 requires a BEV be provided to the EE to access encrypted protected data on the drive. The BEV minimally is a conditioned password

[FCS_PCC_EXT.1] that may be chained [FCS_KYC_EXT.1.1] with one or more submasks [FCS_AFT_EXT.1.1].This set of requirements ensures the BEV is properly generated and protected, preventing unauthorized disclosure of encrypted protected data. FMT_SMF.1.1 ensures the TSF provides the functions necessary to manage important aspects of the TOE including requests to change and erase the DEK.

(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_PCC_EXT.1, FCS_KYC_EXT.1.1, FCS_AFT_EXT.1.1, FPT_ KYP _EXT.1.1, FCS_CKM_EXT.4, FCS_CKM.4.1, FCS_CKM.1.1, FCS_SMV.EXT.1.1, FMT_SMF.1.1]

Rationale: The BEV minimally is a conditioned password [FCS_PCC_EXT.1] that may be chained [FCS_KYC_EXT.1.1] with one or more submasks [FCS_AFT_EXT.1.1]. These requirements ensure the BEV is properly generated and protected. FPT_KYP_EXT.1.1 ensures unwrapped key material is not stored in volatile memory and FCS_CKM_EXT.4 along with FCS_CKM.4.1 ensures proper key material destruction; minimizing the exposure of plaintext keys and key material.

FMT_SMF.1.1 ensures the TSF provides the functions necessary to manage important aspects of the TOE including generating and configuring authorization factors.

(T.UNAUTHORIZED_UPDATE) Threat agents may attempt to perform an update of the product which compromises the security features of the TOE. Poorly chosen update protocols, signature generation and verification algorithms, and parameters may allow attackers to install software and/or firmware that bypasses the intended security features and provides them unauthorized to access to data.

[FPT_TUD_EXT.1.1, FPT_TUD_EXT.1.2, FPT_TUD_EXT.1.3, FMT_SMF.1.1]

Rationale: FPT_TUD_EXT.1.1, FPT_TUD_EXT.1.2, and FPT_TUD_EXT.1.3 provide authorized users the ability to query the current version of the TOE software/firmware, initiate updates, and verify updates prior to installation using a manufacturer digital signature.

FMT_SMF.1.1 ensures the TSF provides the functions necessary to manage important aspects of the TOE including the initiation of system firmware/software updates.

3.2 Assumptions

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

(A. INITIAL_DRIVE_STATE) Users enable Full Drive Encryption on a newly provisioned or initialized storage device free of protected data in areas not targeted for encryption. 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.SECURE_STATE) Upon the completion of proper provisioning, the drive is only assumed secure when in a powered off state up until it is powered on and receives initial authorization.

[OE.MEMORY_REMNANCE]

(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 it assumes that the communication between the components does not breach the boundary of the TOE. 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_AUTHORIZED_USER) Authorized users follow all provided user guidance, including keeping password/passphrases and external tokens securely stored separately from the storage device and/or platform.

[OE.PASSPHRASE_STRENGTH, OE.MEMORY_REMNANCE,
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.SINGLE_USE_ET) External tokens that contain authorization factors are used for no other purpose than to store the external token authorization factors.

[OE.SINGLE_USE_ET]

(A.MEMORY_REMNANCE) The user does not leave the platform and/or storage device unattended until all volatile memory is cleared after a power-off, so memory remnant attacks are infeasible.

Authorized users do not leave the platform and/or storage device in a mode where sensitive information persists in non-volatile storage (e.g., Lockscreen). Users power the platform and/or storage device down or place it into a power managed state, such as a “hibernation mode”.

[OE.MEMORY_REMNANCE]

(A.PLATFORM_I&A) The product does not interfere with or change the normal platform identification and authentication functionality such as the operating system login. It may provide authorization factors to the Operating system's login interface, but it will not change or degrade the functionality of the actual interface.

[OE.PLATFORM_I&A]

(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]

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 must 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 to 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.MEMORY_REMNANCE) Volatile memory is cleared after power-off so memory remnant attacks are infeasible.

> Rationale: Users are properly trained [A_TRAINED_USER] to not leave the storage device unattended until powered down or placed in a managed power state such as “hibernation mode”. A. MEMORY_REMNANCE stipulates that such memory remnant attacks are infeasible given the device is in a powered-down or “hibernation mode” state.

(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 and not store external token authorization factors with the device.

(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.

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

(OE.PLATFORM_I&A) The Operational Environment will provide individual user identification and authentication mechanisms that operate independently of the authorization factors used by the TOE.

> Rationale: While the product may provide authorization factors to the Operating system's login interface, it must not change or degrade the functionality of the actual interface. A.PLATFORM_I&A requires that the product not interfere or change the normal platform I&A functionality.

5. Security Functional Requirements

The individual security functional requirements are specified in the sections below.

Functional Class	Functional Components
Cryptographic support Class (FCS)	FCS_AFA_EXT.1 Authorization Factor Acquisition
Cryptographic support Class (FCS)	FCS_KYC_EXT.1 (Key Chaining)
Cryptographic support Class (FCS)	FCS_PCC_EXT.1 Cryptographic Password Construct and Conditioning
Cryptographic support Class (FCS)	FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction
Cryptographic support Class (FCS)	FCS_CKM.4 Cryptographic key destruction
Cryptographic support Class (FCS)	FCS_SNI_EXT.1 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
Security management Class (FMT)	FMT_SMF.1 Specification of management functions
Protection of the TSF Class (FPT)	FPT_KYP_EXT.1 Extended: Protection of Key and Key Material
Protection of the TSF Class (FPT)	FPT_TUD_EXT.1 Trusted Update

Table 2 TOE Security Functional Requirements

5.1 Class: Cryptographic Support (FCS)

FCS_AFA_EXT.1 Authorization Factor Acquisition

FCS_AFA.EXT.1.1 The TSF shall accept the following authorization factors: a submask derived from a password authorization factor conditioned as defined in FCS_PCC_EXT.1, and [selection of:

- a submask generated by the host platform and stored outside of the TOE,
- a submask generated by the TOE and stored on the TOE,

- a submask generated by the TOE and stored outside of the TOE,
- a submask generated outside of the TOE and stored outside of the TOE,
- no other authorization factor]

that are [selection: 128 bits, 256 bits] in length.

Application Note*: This requirement specifies what authorization factors the TOE accepts from the user. A password entered by the user is a mandatory authorization factor that the TOE must be able to condition, as specified in FCS_PCC_EXT.1. The TOE may accept authorization factors in addition to the password, and these are categorized as "submasks". A submask is simply a bit string that can be generated and stored in a number of ways – it is considered an authorization factor. The ST Author selects the additional authorization factors they support, and there may be multiple methods for a selection. For example, a submask that is generated outside of the TOE and stored outside the TOE, may be generated by a smartcard and stored on a smartcard, as well as, generated by a random number generator provided by the environment, and stored on a USB token. The first item in the selection and the fourth item are similar, the distinction being that the host platform where the FDE resides is generating the submask, where in the fourth case, the submask could be generated by a server or some other external device.*
Use of multiple authorization factors is preferable; if more than one authorization factor is used, the submasks produced must be combined using FCS_SMC_EXT.1 specified in Appendix A.

FCS_KYC_EXT.1 (Key Chaining)

FCS_KYC_EXT.1.1 The TSF shall maintain a key chain of: [selection: one, using a submask as the BEV; intermediate keys originating from a submask 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.2]] while maintaining an effective strength of [selection: AES 128 or AES 256].

FCS_KYC_EXT.1.2 The TSF shall provide a [selection: 128 bit, 256 bit] BEV to the EE [selection: only after the TSF has successfully performed the validation process, without validation taking place].

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., 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).*

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

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

FCS_PCC_EXT.1 Cryptographic Password Construct and Conditioning

FCS_PCC_EXT.1.1 The TSF shall accept passwords that are used to generate a submask that contain at least [assignment: positive integer of 64 or more] characters in the set of {upper case characters, lower case characters, and [assignment: other supported characters]} and shall be conditioned [selection:

- using [selection: SHA-256, SHA-512];
- using NIST SP 800-132 with an iteration count of [assignment: number greater than or equal to 1000], and HMAC as specified in FCS_COP.1.(c) using [selection: HMAC-SHA-256, HMAC-SHA-512];

] such that the submask resulting from the conditioning function is equal to the size (in number of bits) of the BEV.

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

FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction

FCS_CKM_EXT.4 The TSF shall destroy all plaintext keys and plaintext keying material when no longer needed.

Application Note*: Keys, including intermediate keys and key material that are no longer needed are destroyed in volatile memory by using an approved method, FCS_CKM.4.1. Examples of keys are intermediate keys, submasks, and BEV.*

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 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 single direct overwrite [selection: consisting of a pseudo-random pattern using the TSF’s RBG, consisting of

a pseudo-random pattern using the host environment's RBG, consisting of zeroes] following by a read-verify.
    - If read-verification of the overwritten data fails, the process shall be repeated again,
- For non-volatile EEPROM, the destruction shall be executed by a single direct overwrite consisting of [selection: a pseudo random pattern using the TSF's RBG (as specified in FCS_RBG_EXT.1, consisting of a pseudo-random pattern using the host environment's RBG], followed a read-verify.
    - If read-verification of the overwritten data fails, the process shall be repeated again,
- For non-volatile flash memory, the destruction shall be executed [selection: by a single direct overwrite consisting of zeros followed by a read-verify, by a block erase followed by a read-verify].
    - If read-verification of the overwritten data fails, the process shall be repeated again,
- For non-volatile memory other than EEPROM and flash, the destruction shall be executed by overwriting three or more times with a random pattern that is changed before each write

] that meets the following: [NIST SP800-88].

Application Note*: Keys, including intermediate keys and key material that are no longer needed are destroyed in volatile memory by using one of these approved methods. There may be instances where keys or key material that are contained in persistent storage are no longer needed and require destruction. In these cases, the destruction method conforms to one of methods specified in this requirement.FCS_SNI_EXT.1 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)*

FCS_SNI_EXT.1.1 The TSF shall only use salts that are generated by a [selection: RNG as specified in FCS_RBG_EXT.1, RNG provided by the host platform]

FCS_SNI_EXT.1.2 The TSF shall only use unique nonces with a minimum size of 64 bits.

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

- CBC: IVs shall be non-repeating,
- 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 unless an implementation only uses 96-bit IVs (default length).

Application Note: *This requirement covers several important factors – the salt must be random, but the nonces only have to be unique. FCS_SNI_EXT.1.3 specifies how the IV should be handled for each encryption mode.*

5.2 Specification of Management Functions (FMT_SMF)

FMT_SMF.1 Specification of Management Functions

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

a) forwarding requests to change the DEK to the EE,
b) forwarding requests to cryptographically erase the DEK to the EE**,**
c) allowing authorized users to change authorization factors,
d) initiate system firmware/software updates,
e) [selection: no other functions, [selection: generate external authorization factors using the TSF RNG, change default authorization factors, configure cryptographic functionality, disable key recovery functionality, [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 (e) is used to specify functionality that may be included in the TOE, but is not required to conform to the cPP. In item e, if no other management functions are provided (or claimed), then "no other functions" should be selected.*
For the purposes of this document, key sanitization means to destroy the DEK, using one of the approved destruction methods. In some implementations, changing the DEK could be the same functionality as cryptographically erasing the DEK.

5.3 Class: Protection of the TSF (FPT)

FPT_KYP_EXT.1 Extended: Protection of Key and Key Material

FPT_ KYP _EXT.1.1 The TSF shall only store keys in non-volatile memory when wrapped, as specified in FCS_COP.1(d).

Application Note: *When stored, the BEV is always encrypted (wrapped) and only exists in plaintext form, in volatile memory, when it is being used to encrypt or decrypt data.*

FPT_TUD_EXT.1 Trusted Update

FPT_TUD_EXT.1.1 The TSF shall provide authorized users the ability to query the current version of the TOE software/firmware.

FPT_TUD_EXT.1.2 The TSF shall provide authorized users the ability to initiate updates to TOE software/firmware.

FPT_TUD_EXT.1.3 The TSF shall verify updates to the system software/firmware using a digital signature by the manufacturer prior to installing those updates.

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

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. Individual Evaluation Activities to be performed are specified in *Supporting Document (Mandatory Technical Document) Full Drive Encryption: Authorization Acquisition September 2014*.

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.

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)

Table 3: Security Assurance Requirements

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 0f 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 Key Management Description (Appendix E), and [selection: Entropy Essay, 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 will necessarily have interfaces to the Operational Environment that are not directly invokable by TOE users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional "functional specification" documentation is necessary to satisfy the Evaluation Activities specified in the SD.

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

6.3 AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies 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. 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 application developers 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.

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

6.6.1 Vulnerability Survey (AVA_VAN.1)

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

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 types of requirements specified in Appendices A and B.

The first type (in this Appendix) is 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 type (in Appendix B) is requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements in that appendix will need to be included in the body of the ST (e.g., cryptographic protocols selected in a trusted channel requirement).

Some of the requirements in this section are iterated, but since the ST Author is responsible for incorporating the appropriate requirements from the appendices into the body of their ST, the correct iteration numbering is left to the ST Author.

A.1 Class: Cryptographic Support (FCS)

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). The requirements in this section specify the cryptographic functionality that must be present either in the TOE or the Operational Environment in order for the TOE to satisfy its security objectives. If the functionality is present in the TOE, then these requirements will be moved by the ST Author to the body of the ST.

If the functionality is merely used by the TOE and provided by the Operational Environment, then the developer will identify those functions in each Operational Environment listed in the ST. This identification should be such that an evaluator can use the information in the TSS (which requires that the method by which each operation is invoked is identified) coupled with the information on the functions in the Operational Environment to perform activities to validate that each Operational Environment listed for the TOE is able to meet the requirements in this section. The evaluator checks the Operational Environment to make sure they supply those functions and that the interfaces exist in the Operational Environment documentation.

FCS_CKM.1 Cryptographic Key Generation(Asymmetric Keys)

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

- ***RSA schemes*** *using cryptographic key sizes of* ***2048-bit or greater*** *that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3;*

- ***ECC schemes*** *using* ***"NIST curves" P-256, P-384 and [selection: P-521, no other curves]*** *that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;*
- ***FFC schemes*** *using cryptographic key sizes of* ***2048-bit or greater*** *that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1*

].

Application Note: *The ST author shall select all key generation schemes used for key establishment. When key generation is used for key establishment, the schemes in FCS_CKM.2.1 and selected cryptographic protocols must match the selection.*

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

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-512] to generate a [selection: intermediary key, BEV].

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) and FCS_COP.1(c).*

FCS_VAL.EXT.1 Validation

FCS_VAL.EXT.1.1 The TSF shall perform validation of the [selection: submask, intermediate key, BEV] using the following methods: [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 forward the BEV to the EE only after validation has occurred

FCS_VAL_EXT.1.3 The TSF shall [selection: issue a key sanitization of the DEK to the EE upon a configurable number of consecutive failed validation attempts, institute a delay such that only 300 attempts can be made within a 24 hour period].

***Application Note**: The purpose of performing secure validation is to not expose any material that might compromise the submask(s).*

The TOE validates the submask(s) (e.g., authorization factor(s)) prior to allowing the user access to the data stored on the drive. When a password is used as an authorization factor, it is conditioned before any attempts to validate. In cases where validation of the authorization factor(s) fails, the product will not forward a BEV to EE.

When the key wrap in FCS_COP.1(d) is used, the validation is performed inherently.

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

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

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

- <u>RSA Digital Signature Algorithm with a key size (modulus)of 2048 bits or greater,</u>
- <u>Elliptic Curve Digital Signature Algorithm with a key size of 256 bits or greater</u>

]

that meets the following: [selection:

- <u>FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS2v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3, for RSA schemes</u>
- <u>FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" P-256, P-384, and [selection: P-521, no other curves]; ISO/IEC 14888-3, Section 6.4, for ECDSA schemes</u>

].

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

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

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

Application Note: *The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1(1) and FCS_COP.1(2) (SHA 256 for 128-bit keys, SHA 512 for 256-bit keys). The selection of the standard is made based on the algorithms selected.*

FCS_COP.1(c) Cryptographic operation (Keyed Hash Algorithm)

FCS_COP.1.1(c) The TSF shall perform keyed-hash message authentication in accordance with [selection: HMAC-SHA-256, HMAC-SHA-512] and cryptographic key sizes [assignment: key size (in bits) used in HMAC] that meet the following:[ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”].

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

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

FPT_TST_EXT.1 Extended: TSF Testing

FPT_TST_EXT.1.1 The TSF shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of 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.*

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 will need to be included.

As with those requirements specified in Appendix A, the ST Author will have to ensure they correctly adjust the iteration number, which, of course, is dependent on the optional and selection-based requirements that are pulled into the ST body.

B.1 Class: Cryptographic Support (FCS)

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

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

Application Note*: The intent of this requirement in the context of this cPP is to provide an SFR that expresses the appropriate symmetric encryption/decryption algorithms suitable for use in the TOE. If the ST Author incorporates the validation requirement (FCS_VAL.EXT.1) and chooses to select the option to decrypt a known value and perform a comparison, this is the requirement used to specify the algorithm, modes, and key sizes the ST Author can choose from.*

FCS_COP.1(e) Cryptographic operation (Key Wrapping)

FCS_COP.1.1(e) 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: [ISO/IEC 18033-3 (AES), [selection: NIST SP 800-38F, NIST SP 800-38D, NIST SP 800-38C]].

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.1.*

FCS_KDF_EXT.1 Cryptographic Key Derivation

FCS_KDF_EXT.1.1The 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 NIST SP 800-108, using the hash functions specified in FCS_COP.1(b) and FCS_COP.1(c), such that the output is at least equal to the size (in number of bits) of 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.1.*

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 Extended: Cryptographic Operation (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 using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)]].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: *a software-based noise source*, *hardware-based noise source*] 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-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.*

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.

C.1 Background and Scope

This document provides a definition for all of the extended components used in the *collaborative Protection Profile for Full Drive Encryption—Authorization Acquisition.* These components are identified in the following table:

FCS_AFA_EXT.1	Authorization Factor Acquisition
FCS_KYC_EXT.1	Key Chaining
FCS_PCC_EXT.1	Cryptographic Password Construction and Conditioning
FCS_CKM_EXT.4	Cryptographic Key and Key Material Destruction
FCS_SNI_EXT.1	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
FPT_KYP_EXT.1	Extended: Protection of Key and Key Material
FPT_TUD_EXT.1	Trusted Update
FCS_SMC_EXT.1	Submask Combining
FCS_VAL_EXT.1	Validation
FPT_TST_EXT.1	Extended: TSF Testing
FCS_KDF_EXT.1	Cryptographic Key Derivation
FCS_RBG_EXT.1	Extended: Cryptographic operation (Random Bit Generation)

C.2 Extended Component Definitions

Authorization Factor Acquisition (FCS_AFA_EXT)

Family Behavior

Components in this family address the ability for the TOE to accept a variety of authorization factors.

Component leveling

FCS_AFA_EXT Extended: Authorization Factor Acquisition
1

FCS_AFA_EXT.1 Extended: Authorization Factor Acquisition, requires authorization factors to be accepted by the TOE.

Management: FCS_AFA_EXT.1

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

Change the authorization factors to be used

Generate external authorization factors using the TSF RNG

Audit: FCS_AFA_EXT.1

There are no auditable events foreseen.

FCS_AFA_EXT.1 Authorization Factor Acquisition

Hierarchical to: No other components

Dependencies: No other components

FCS_AFA.EXT.1.1 The TSF shall accept the following authorization factors: a submask derived from a password authorization factor conditioned as defined in FCS_PCC_EXT.1, and [selection of:

- **a submask generated by the host platform and stored outside of the TOE,**
- **a submask generated by the TOE and stored on the TOE,**
- **a submask generated by the TOE and stored outside of the TOE,**
- **a submask generated outside of the TOE and stored outside of the TOE,**
- **no other authorization factor]**

that are [selection: 128 bits, 256 bits] in length.

Key Chaining (FCS_KYC_EXT)

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, requires the TSF to maintain a key chain and specifies the characteristics of that chain.

Management: FCS_KYC_EXT.1

No specific management functions are identified

Audit: FCS_KYC_EXT.1

There are no auditable events foreseen.

FCS_KYC_EXT.1 Key Chaining

Hierarchical to: No other components

Dependencies: No other components

FCS_KYC_EXT.1.1 The TSF shall maintain a key chain of: [selection: one, using a submask as the BEV; intermediate keys originating from a submask 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.2]].

FCS_KYC_EXT.1.2 The TSF shall provide a [selection: 128 bit, 256 bit] BEV to the EE [selection: only after the TSF has successfully performed the validation process, without validation taking place].

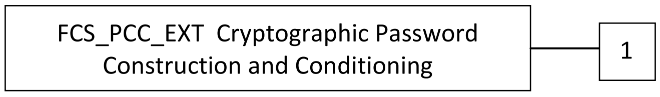
Application Note: *Key Chaining is the method of using multiple layers of encryption keys to ultimately secure the protected data encrypted on the drive. The number of intermediate keys will vary – from one (e.g., taking the conditioned password authorization factor and directly using it as the BEV) 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).*

Cryptographic Password Construction and Conditioning (FCS_PCC_EXT)

Family Behavior

This family ensures that passwords used to produce the BEV are robust (in terms of their composition) and are conditioned to provide an appropriate-length bit string.

Component leveling


FCS_PCC_EXT Cryptographic Password Construction and Conditioning
1

FCS_PCC_EXT.1 Cryptographic Password Construction and Conditioning, requires the TSF to accept passwords of a certain composition and condition them appropriately.

Management: FCS_PCC_EXT.1

No specific management functions are identified

Audit: FCS_PCC_EXT.1

There are no auditable events foreseen.

FCS_PCC_EXT.1 Cryptographic Password Construction and Conditioning

Hierarchical to: No other components

Dependencies: No other components

FCS_PCC_EXT.1.1 The TSF shall accept passwords that are used to generate a submask that contain at least [assignment: *number*] characters in the set of [assignment: supported characters] and shall be conditioned [selection:

- **using [selection: SHA-256, SHA-512];**
- **using NIST SP 800-132 with an iteration count of [assignment: number greater than or equal to 1000], and HMAC as specified in FCS_COP.1.(c) using [selection: HMAC-SHA-256, HMAC-SHA-512];**

] such that the submask resulting from the conditioning function is equal to the size (in number of bits) of the BEV.

Cryptographic Key Management (FCS_CKM)

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.

Component leveling

FCS_CKM_EXT Cryptographic Key and Key Material Destruction
4

FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction, is an extended component under FCS_CKM.4 and contains requirements on the timing 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.

FCS_CKM_EXT.4 Cryptographic Key and Key Material Destruction
Hierarchical to: No other components

Dependencies: No other components

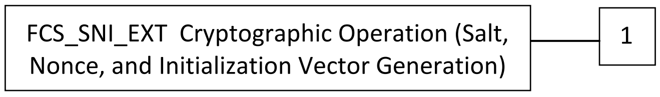
FCS_CKM_EXT.4 The TSF shall destroy all plaintext keys (intermediate keys, submasks, and BEV) and plaintext keying material when no longer needed.

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

Family Behavior

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

Component leveling


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

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: No other components

FCS_SNI_EXT.1.1 The TSF shall only use salts that are generated by a [selection: RNG as specified in FCS_RBG_EXT.1, RNG provided by the host platform].

FCS_SNI_EXT.1.2 The TSF shall only use unique nonces with a minimum size of [assignment: number of bits] bits.

FCS_SNI_EXT.1.3 The TSF shall create IVs in the following manner: [assignment: *list of algorithms/modes that require IVs, and associated requirements on those IVs*].

FCS_SNI_EXT.1.4 The TSF shall ensure that tweak values for AES-XTS are non-negative integers that are assigned consecutively.

Key and Key Material Protection (FPT_KYP_EXT)

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 Extended: 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 Extended: Protection of Key and Key Material

Hierarchical to: No other components

Dependencies: No other components

FPT_ KYP _EXT.1.1 The TSF shall only store keys in non-volatile memory when wrapped, as specified in FCS_COP.1(d).

Trusted Update (FPT_TUD_EXT)

Family Behavior

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

Component leveling

FPT_TUD_EXT Trusted Update
1

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), or

FCS_COP.1(b) Cryptographic operation (hash algorithm)

FPT_TUD_EXT.1.1 The TSF shall provide [assignment: role or group] the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The TSF shall provide [assignment: role or group] the ability to initiate updates to TOE firmware/software.

FPT_TUD_EXT.1.3 The TSF shall verify updates to the system firmware/software updates to the TOE using a [selection: digital signature mechanism, published hash] by the manufacturer prior to installing those updates.

Submask Combining (FCS_SMC_EXT)

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)

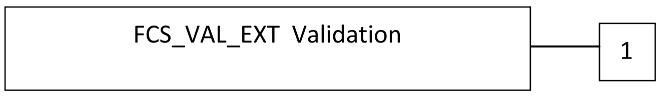
FCS_SMC_EXT.1.1 The TSF shall combine submasks using the following method [selection: exclusive OR (XOR), [assignment: hash function], [assignment: other combining function]] to generate a [selection: intermediary key, BEV].

Validation of Cryptographic Elements (FCS_VAL_EXT)

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(d) Cryptographic Operation (key wrapping)

FCS_VAL.EXT.1.1 The TSF shall perform validation of the [selection: submask, BEV] using the following methods: [assignment: *list of methods used to validate the submask/BEV*].

FCS_VAL_EXT.1.2 The TSF shall forward the BEV to the EE only after validation has occurred.

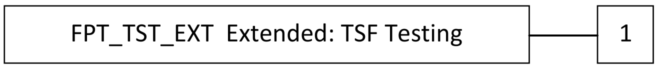
FCS_VAL_EXT.1.3 The TSF shall [selection: issue a key sanitization of the DEK to the EE upon a configurable number of consecutive failed validation attempts, institute a delay such that only 300 failed validation attempts can be made within a 24 hour period].

TSF Self-Test (FPT_TST_EXT)

Family Behavior

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

Component leveling



FPT_TST_EXT.1 Extended: 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

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

No management functions.

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 Extended: TSF Testing

Hierarchical to: No other components.

Dependencies: No other components.

FPT_TST_EXT.1.1 The TSF shall run a suite of self-tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

Key Derivation (FCS_KDF_EXT)

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(b) Cryptographic Operation (hash algorithm)

FCS_COP.1(c) Cryptographic Operation (keyed hash algorithm)

FCS_KDF_EXT.1.1The TSF shall accept [assignment: *types of submasks*] to derive an intermediate key, as defined in NIST SP 800-108, using [assignment: *hash or keyed hash functions specified in FCS_COP.1(b) and FCS_COP.1(c)*] such that the output is at least equal to the size (in number of bits) of the BEV.

Random Bit Generation (FCS_RBG_EXT)

Family Behavior

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

Component leveling

FCS_RBG_EXT Extended: Random Bit Generation
1

FCS_RBG_EXT.1 Extended: 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

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

There are no management activities foreseen

Audit: FCS_RBG_EXT.1

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

Minimal: failure of the randomization process

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

Hierarchical to: No other components

Dependencies: No other components

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

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: *a software-based noise source, a hardware-based noise source*] with minimum of [selection; *128 bits, 192 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-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.*

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>

The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide 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.

D.1 Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, 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 random comes from, where it 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 exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure 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 entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

D.3 Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. 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, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

E. Key Management Description

The documentation of the product's 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.

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

The essay will also describe the following topics:

- A description of all authorization factors that are supported by the product and how each factor is handled, including any conditioning and combining performed.
- If validation is supported, the process for validation shall be described, noting what value 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.
- The authorization process that leads to the ultimate release of the BEV. This section shall detail the key chain used by the product. It shall describe which keys are used in the protection of the BEV and how they meet the derivation, key wrap, or a combination of the two requirements, including the direct chain from the initial authorization to the BEV. 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 evaluator shall examine the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or the initial authorization value and the effective strength of the BEV is maintained throughout the Key Chain.
- The process for destroying keys when they are no longer needed by describing the storage location of all keys and the protection of all keys stored in non-volatile memory.

Diagram:

- The diagram will include all of keys from the initial authorization factor to the BEV and any keys or values that contribute into the chain. It must list the cryptographic strength of each key and explain 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.

1 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 and that is used in the derivation or decryption of the BEV and eventual decryption of the DEK. Note that these values may or may not be used to establish the particular identity of the user.
Assurance	Grounds for confidence that a TOE meets the SFRs [CC1].
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 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. 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 Release Key (KRK)	A key used to release another key from storage, it is not used for the direct derivation or decryption of another key.
Operating System (OS)	Software which runs at the highest privilege level and can directly control hardware resources.
Non-Volatile Memory	A type of computer memory that will retain information without power.

Powered-Off State	The device has been shutdown.
Protected Data	This refers to all data on the hard drive 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.
Submask	A submask is a bit string that can be generated and stored in a number of ways.

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

1 G. Acronyms

Acronym	Meaning
AA	Acquisition Authorization
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
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
ITSEF	IT Security Testing Laboratory
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
PBKDF	Password-Based Key Derivation Function
PRF	Pseudo Random Function
RBG	Random Bit Generator
RNG	Random Number Generator
RSA	Rivest Shamir Adleman Algorithm
SAR	Security Assurance Requirements
SED	Self Encrypting Drive
SHA	Secure Hash Algorithm
SFR	Security Functional Requirements
SPD	Security Problem Definition

Acronym	Meaning
SPI	Security Parameter Index
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

1