KLC Group LLC

CipherDriveOne Kryptr 1.1.0

Security Target

Version 1.5

April 2024

Document prepared by

www.lightshipsec.com

Document History

Version	Date	Author	Description
1.0	31 Aug 2023	G Nickel	Release for NIAP Check-In
1.1	12 Oct 2023	G Nickel	Updates to AA:FCS_PCC_EXT.1, Address OR11 and OR13.
1.2	19 Oct 2023	G Nickel	Address OR13v2, Updates to AA/EE:FCS_COP.1(g) and section 2.4.3
1.3	15 Mar 2024	G Nickel	Update CAVP Mappings, Update build version, address evaluator comments.
1.4	04 Apr 2024	G Nickel	Release for Check Out
1.5	23 Apr 2024	G Nickel	Address OR15/ECR Comments

Table of Contents

List of Tables

1 Introduction

1.1 Overview

1 This Security Target (ST) defines the CipherDriveOne Kryptr 1.1.0 Target of Evaluation (TOE) for the purposes of Common Criteria (CC) evaluation.

2 CipherDriveOne Kryptr 1.1.0 is software that provides pre-boot authentication (PBA) and full drive encryption. It applies military strength encryption to protect all locally stored data from unauthorized access, loss, and exposure in the event a protected device is lost or stolen.

1.2 Identification

Table 1: Evaluation identifiers

Target of Evaluation	KLC Group LLC CipherDriveOne Kryptr 1.1.0 Build: 17
Security Target	KLC Group LLC CipherDriveOne Kryptr 1.1.0 Security Target, v1.5

1.3 Conformance Claims

3 This ST supports the following conformance claims:

a) CC version 3.1 revision 5

b) CC Part 2 extended

c) CC Part 3 conformant

d) collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition, v2.0 + Errata 20190201 (referenced within as CPP_FDE_AA)

e) collaborative Protection Profile for Full Drive Encryption – Encryption Engine, v2.0 + Errata 20190201 (referenced within as CPP_FDE_EE)

f) NIAP Technical Decisions per Table 2

Table 2: NIAP Technical Decisions

TD #	Name	Applicability Rationale	Source
TD0458	FIT Technical Decision for FPT_KYP_EXT.1 evaluation activities	Applicable	CPP_FDE_AA CPP_FDE_EE
TD0460	FIT Technical Decision for FPT_PWR_EXT.1 non-compliant power saving states	Applicable	CPP_FDE_EE
TD0464	FIT Technical Decision for FPT_PWR_EXT.1 compliant power saving states	Applicable	CPP_FDE_EE
TD0606	FIT Technical Recommendation for Evaluating a NAS against the FDE AA and FDEE	Not Applicable -	CPP_FDE_AA CPP_FDE_EE

TD #	Name	Applicability Rationale	Source
		TOE is not a NAS	
TD0759	FIT Technical Decision for FCS_AFA_EXT.1.1	Applicable	CPP_FDE_AA
TD0760	FIT Technical Decision for FCS_SNI_EXT.1.3, FCS_COP.1(f)	Applicable	CPP_FDE_AA
TD0764	FIT Technical Decision for FCS_PCC_EXT.1	Applicable	CPP_FDE_AA
TD0765	FIT Technical Decision for FMT_MOF.1	Applicable	CPP_FDE_AA
TD0766	FIT Technical Decision for FCS_CKM.4(d) Test Notes	Applicable	CPP_FDE_AA CPP_FDE_EE
TD0767	FIT Technical Decision for FMT_SMF.1.1	Applicable	CPP_FDE_AA
TD0769	FIT Technical Decision for FPT_KYP_EXT.1.1	Applicable	CPP_FDE_AA CPP_FDE_EE

1.4 Terminology

Table 3: Terminology

Term	Definition
AA	Authorization Acquisition
AES	Advanced Encryption Standard
AK	Authentication Key
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
CDO	CipherDriveOne Kryptr
CPP	Collaborative Protection Profile
DAR	Data At Rest

Term	Definition
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
EFI	Extensible Firmware Interface
EMK	Encrypted Master Key
ESP	EFI System Partition
FIPS	Federal Information Processing Standards
FDE	Full Drive Encryption
FFC	Finite Field Cryptography
GCM	Galois Counter Mode
GPT	GUID Partition Table
GUID	Globally Unique Identifier
HMAC	Keyed-Hash Message Authentication Code
HW	Hardware
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
KLC	KLC Group LLC

Term	Definition
KMD	Key Management Description
KRK	Key Release Key
LKRNG	Linux Kernel Random Number Generator
LUKS	Linux Unified Key Setup
MBR	Master Boot Record
MK	Master Key
NIST	National Institute of Standards and Technology
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PIV-CAC	Personal Identity Verification Common Access Card
PRF	Pseudo Random Function
PXE	Preboot eXecution Environment
RBG	Random Bit Generator
RHEL	Red Hat Enterprise Linux
RNG	Random Number Generator
RSA	Rivest Shamir Adleman Algorithm
RSAEP	RSA Encryption Primitive
RSADP	RSA Decryption Primitive
SAR	Security Assurance Requirements
SHA	Secure Hash Algorithm
SFR	Security Functional Requirements
ST	Security Target
SPD	Security Problem Definition
SPI	Serial Peripheral Interface
TOE	Target of Evaluation

Term	Definition
TPM	Trusted Platform Module
TSF	TOE Security Functionality
TSS	TOE Summary Specification
UEFI	Unified Extensible Firmware Interface
USB	Universal Serial Bus
XOR	Exclusive or
XTS	XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing

2 TOE Description

2.1 Type

4 The TOE is software that provides pre-boot authentication (PBA) and full drive encryption.

2.2 Usage

5 The TOE provides software full disk encryption including pre-boot user authentication, chain-boot to the host Operating System (OS) and management capabilities to control user access and settings. The TOE has two distinct modules:

a) **PBA / Management module.** Provides pre-boot authentication and TOE configuration services. This module satisfies the requirements of CPP_FDE_AA.

b) **Encryption Engine / Driver.** Disk encryption for Linux and Windows Operating Systems. This module satisfies the requirement of CPP_FDE_EE.

2.3 Security Functions / Logical Scope

6 The TOE provides the following security functions:

a) **User Data Protection.** The TOE performs full drive encryption on all storage devices to protect data from unauthorized disclosure in the event of loss or theft of the device. All protected data is encrypted by default without user intervention and with NIST approved algorithms.

b) **Security Management.** The TOE provides dedicated interfaces for the management of its security functions. Access to these management functions can be controlled by way of role-based group assignment to administrative users.

c) **Protection of the TSF.** The TOE ensures the authenticity and integrity of software updates by verifying their digital signatures prior to installation. Various software and cryptographic self-tests are performed at start-up to ensure the secure and correct operation of the TOE. All keying material used for storage encryption is securely generated and protected from disclosure.

d) **Cryptographic Support.** The TOE performs cryptographic operations as shown in Table 12, which includes relevant Cryptographic Algorithm Validation Program (CAVP) certificates. Secure destruction of cryptographic keys and keying material is implemented and occurs during transition to a compliant power saving state, or when the key or keying material is no longer needed.

2.4 Physical Scope

7 The physical boundary of the TOE encompasses the KLC Group LLC CipherDriveOne Kryptr 1.1.0 software (including Linux Kernel 5.15). Users download the software after purchase from KLC's web portal.

2.4.1 Guidance Documents

8 The TOE includes the following guidance documents:

a) KLC Group LLC CipherDriveOne Kryptr 1.1.0 Common Criteria Guide, Version 1.1

b) KLC Group LLC CipherDriveOne Kryptr Administrator Guide, V 1.0.1, 4-18-2024

9 Users can download these guidance documents in PDF file format from the NIAP PCL website.

2.4.2 Non-TOE Components

10 The TOE operates with the following components in the environment:

a) **Protected OS.** The TOE supports protection of the following Linux Operating Systems and Windows Operating Systems:

i) Red Hat Enterprise Linux 8

ii) Red Hat Enterprise Linux 9

iii) Microsoft Windows 10

iv) Microsoft Windows 11

CC Testing was performed using the following operating systems:

- Red Hat Enterprise Linux 9
- Microsoft Windows 11

b) **Computer Hardware.** 64-bit Intel-based UEFI booted systems that supports Intel Secure Key Technology. CC testing was performed using the following CPUs:

i) Intel Core i7-1265U (Alder Lake)

c) **Smartcard and reader.** When dual factor authentication is used, Federal Information Processing Standard (FIPS) 201 Personal Identity Verification Common Access Card (PIV-CAC) compliant smartcards and readers are required.

2.4.3 Security Functions not included in the TOE Evaluation

11 The evaluation is limited to the security functions identified in section 2.3 only.

12 For additional certainty given the TOE type, the following functions were not tested or addressed as part of this evaluation:

a) **Hypervisor Support.** While the TOE was designed to also support Linux-based hypervisors such as OpenXT and SecureView, they were not tested.

3 Security Problem Definition

13 The Security Problem Definition is reproduced from the claimed PPs.

3.1 Threats

Table 4: CPP_FDE_AA Threats

Identifier	Description
T.UNAUTHORIZED_DATA_ACCESS	The cPP addresses the primary threat of unauthorized disclosure of protected data stored on a storage device. If an adversary obtains a lost or stolen storage device (e.g., a storage device contained in a laptop or a portable external storage device), they may attempt to connect a targeted storage device to a host of which they have complete control and have raw access to the storage device (e.g., to specified disk sectors, to specified blocks).
T.KEYING_MATERIAL_COMPROMISE	Possession of any of the keys, authorization factors, submasks, and random numbers or any other values that contribute to the creation of keys or authorization factors could allow an unauthorized user to defeat the encryption. The cPP considers possession of key material of equal importance to the data itself. Threat agents may look for key material in unencrypted sectors of the storage device and on other peripherals in the operating environment (OE), e.g. BIOS configuration, SPI flash.
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 BEV or otherwise put it in a state in which it discloses protected data to unauthorized users.
T.KEYSPACE_EXHAUST	Threat agents may perform a cryptographic exhaust against the key space. Poorly chosen encryption algorithms and/or parameters allow attackers to exhaust the key space through brute force and give them unauthorized access to the data.
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 access to data.

Table 5: CPP_FDE_EE Threats

Identifier	Description
T.UNAUTHORIZED_DATA_ACCESS	The cPP addresses the primary threat of unauthorized disclosure of protected data stored on a storage device. If an adversary obtains a lost or stolen storage device (e.g., a storage device contained in a laptop or a portable external storage device), they may attempt to connect a targeted storage device to a host of which they have complete control and have raw access to the storage device (e.g., to specified disk sectors, to specified blocks).
T.KEYING_MATERIAL_COMPROMISE	Possession of any of the keys, authorization factors, submasks, and random numbers or any other values that contribute to the creation of

Identifier	Description
	keys or authorization factors could allow an unauthorized user to defeat the encryption. The cPP considers possession of keying material of equal importance to the data itself. Threat agents may look for keying material in unencrypted sectors of the storage device and on other peripherals in the operating environment (OE), e.g. BIOS configuration, SPI flash, or TPMs.
T.AUTHORIZATION_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.
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.
T.KNOWN_PLAINTEXT	Threat agents know plaintext in regions of storage devices, especially in uninitialized regions (all zeroes) as well as regions that contain well known software such as operating systems. A poor choice of encryption algorithms, encryption modes, and initialization vectors along with known plaintext could allow an attacker to recover the effective DEK, thus providing unauthorized access to the previously unknown plaintext on the storage device.
T.CHOSEN_PLAINTEXT	Threat agents may trick authorized users into storing chosen plaintext on the encrypted storage device in the form of an image, document, or some other file. A poor choice of encryption algorithms, encryption modes, and initialization vectors along with the chosen plaintext could allow attackers to recover the effective DEK, thus providing unauthorized access to the previously unknown plaintext on the storage device.
T.UNAUTHORIZED_UPDATE	Threat agents may attempt to perform an update of the product which compromises the security features of the TOE. Poorly chosen update protocols, signature generation and verification algorithms, and parameters may allow attackers to install software that bypasses the intended security features and provides them unauthorized access to data.
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.
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.

3.2 Assumptions

Table 6: CPP_FDE_AA Assumptions

Identifier	Description
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.
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.
A.TRUSTED _CHANNEL	Communication among and between product components (e.g., AA and EE) is sufficiently protected to prevent information disclosure. In cases in which a single product fulfils both cPPs, then the communication between the components does not extend beyond the boundary of the TOE (e.g., communication path is within the TOE boundary). In cases in which independent products satisfy the requirements of the AA and EE, the physically close proximity of the two products during their operation means that the threat agent has very little opportunity to interpose itself in the channel between the two without the user noticing and taking appropriate actions.
A.TRAINED_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.
A.PLATFORM_STATE	The platform in which the storage device resides (or an external storage device is connected) is free of malware that could interfere with the correct operation of the product.
A.SINGLE_USE_ET	External tokens that contain authorization factors are used for no other purpose than to store the external token authorization factors.
A.POWER_DOWN	The user does not leave the platform and/or storage device unattended until all volatile memory is cleared after a power-off, 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., lock screen). Users power the platform and/or storage device down or place it into a power managed state, such as a “hibernation mode”.

Identifier	Description
A.PASSWORD _STRENGTH	Authorized administrators ensure password/passphrase authorization factors have sufficient strength and entropy to reflect the sensitivity of the data being protected.
A.PLATFORM_I&A	The product does not interfere with or change the normal platform identification and authentication functionality such as the operating system login. It may provide authorization factors to the operating system's login interface, but it will not change or degrade the functionality of the actual interface.
A.STRONG_CRYPTO	All cryptography implemented in the Operational Environment and used by the product meets the requirements listed in the cPP. This includes generation of external token authorization factors by a RBG.
A.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.

Table 7: CPP_FDE_EE Assumptions

Identifier	Description
A.TRUSTED _CHANNEL	Communication among and between product components (e.g.,AA and EE) is sufficiently protected to prevent information disclosure. In cases in which a single product fulfils both cPPs, then the communication between the components does not extend beyond the boundary of the TOE (e.g., communication path is within the TOE boundary). In cases in which independent products satisfy the requirements of the AA and EE, the physically close proximity of the two products during their operation means that the threat agent has very little opportunity to interpose itself in the channel between the two without the user noticing and taking appropriate actions.
A.INITIAL_DRIVE _STATE	Users enable Full Drive Encryption on a newly provisioned storage device free of protected data in areas not targeted for encryption. It is also assumed that data intended for protection should not be on the targeted storage media until after provisioning. The cPP does not intend to include requirements to find all the areas on storage devices that potentially contain protected data. In some cases, it may not be possible - for example, data contained in "bad" sectors. While inadvertent exposure to data contained in bad sectors or unpartitioned 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.

Identifier	Description
A.TRAINED_USER	Users follow the provided guidance for securing the TOE and authorization factors. This includes conformance with authorization factor strength, using external token authentication factors for no other purpose and ensuring external token authorization factors are securely stored separately from the storage device and/or platform. The user should also be trained on how to power off their system.
A.PLATFORM_STATE	The platform in which the storage device resides (or an external storage device is connected) is free of malware that could interfere with the correct operation of the product.
A.POWER_DOWN	The user does not leave the platform and/or storage device unattended until 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”.
A.STRONG_CRYPTO	All cryptography implemented in the Operational Environment and used by the product meets the requirements listed in the cPP. This includes generation of external token authorization factors by a RBG.
A.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.

3.3 Organizational Security Policies

14 None defined.

4 Security Objectives

15 The security objectives are reproduced from the claimed PPs.

Table 8: CPP_FDE_AA Security Objectives for the Operational Environment

Identifier	Description
OE.TRUSTED_CHANNEL	Communication among and between product components (i.e., AA and EE) is sufficiently protected to prevent information disclosure.
OE.INITIAL_DRIVE_STATE	The OE provides a newly provisioned or initialized storage device free of protected data in areas not targeted for encryption.
OE.PASSPHRASE_STRENGTH	An authorized administrator will be responsible for ensuring that the passphrase authorization factor conforms to guidance from the Enterprise using the TOE.

Identifier	Description
OE.POWER_DOWN	Volatile memory is cleared after power-off so memory remnant attacks are infeasible.
OE.SINGLE_USE_ET	External tokens that contain authorization factors will be used for no other purpose than to store the external token authorization factor.
OE.STRONG _ENVIRONMENT _CRYPTO	The Operating Environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE and Appendix A.
OE.TRAINED_USERS	Authorized users will be properly trained and follow all guidance for securing the TOE and authorization factors.
OE.PLATFORM _STATE	The platform in which the storage device resides (or an external storage device is connected) is free of malware that could interfere with the correct operation of the product.
OE.PLATFORM_I&A	The Operational Environment will provide individual user identification and authentication mechanisms that operate independently of the authorization factors used by the TOE.
OE.PHYSICAL	The Operational Environment will provide a secure physical computing space such than an adversary is not able to make modifications to the environment or to the TOE itself.

Table 9: CPP_FDE_EE Security Objectives for the Operational Environment

Identifier	Description
OE.TRUSTED _CHANNEL	Communication among and between product components (e.g.,AA and EE) is sufficiently protected to prevent information disclosure.
OE.INITIAL_DRIVE _STATE	The OE provides a newly provisioned or initialized storage device free of protected data in areas not targeted for encryption.
OE.PASSPHRASE _STRENGTH	An authorized administrator will be responsible for ensuring that the passphrase authorization factor conforms to guidance from the Enterprise using the TOE.
OE.POWER_DOWN	Volatile memory is cleared after entering a Compliant power saving state or turned off so memory remnant attacks are infeasible.
OE.SINGLE_USE_ET	External tokens that contain authorization factors will be used for no other purpose than to store the external token authorization factor.
OE.STRONG _ENVIRONMENT _CRYPTO	The Operating Environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE and Appendix A.
OE.TRAINED_USERS	Authorized users will be properly trained and follow all guidance for securing the TOE and authorization factors.

Identifier	Description
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.

5 Security Requirements

5.1 Conventions

16 This document uses the following font conventions to identify the operations defined by the CC:

a) **Assignment.** Indicated with italicized text.

b) **Refinement.** Indicated with bold text and strikethroughs.

c) **Selection.** Indicated with underlined text.

d) **Assignment within a Selection:** Indicated with italicized and underlined text.

e) **Iteration.** Indicated by appending 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.

17 **Note:** Operations performed within the Security Target are denoted within brackets []. Operations shown without brackets are reproduced from the PP.

5.2 Extended Components Definition

18 Extended components are defined in CPP_FDE_AA and CPP_FDE_EE.

5.3 Functional Requirements

19 In this ST, SFRs from CPP_FDE_AA are prefixed with 'AA', and SFRs from CPP_FDE_EE are prefixed with 'EE'. SFRs with no prefix are combined from both PPs (because they are the same when operations are completed).

Table 10: Summary of SFRs

Requirement	Title
AA:FCS_AFA_EXT.1	Authorization Factor Acquisition
AA:FCS_AFA_EXT.2	Timing of Authorization Factor Acquisition
AA:FCS_CKM.4(a)	Cryptographic Key Destruction (Power Management)
AA:FCS_CKM.4(d)	Cryptographic Key Destruction (Software TOE, 3rd Party Storage)
AA:FCS_CKM_EXT.4(a)	Cryptographic Key and Key Material Destruction (Destruction Timing)

Requirement	Title
AA:FCS_CKM_EXT.4(b)	Cryptographic Key and Key Material Destruction (Power Management)
AA:FCS_KYC_EXT.1	Key Chaining (Initiator)
AA:FCS_SNI_EXT.1	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
AA:FMT_MOF.1	Management of Functions Behavior
AA:FMT_SMF.1	Specification of Management Functions
AA:FMT_SMR.1	Security Roles
AA:FPT_KYP_EXT.1	Protection of Key and Key Material
AA:FPT_PWR_EXT.1	Power Saving States
AA:FPT_PWR_EXT.2	Timing of Power Saving States
AA:FPT_TUD_EXT.1	Trusted Update
EE:FCS_CKM.1(c)	Cryptographic Key Generation (Data Encryption Key)
EE:FCS_CKM.4(a)	Cryptographic Key Destruction (Power Management)
EE:FCS_CKM_EXT.4(a)	Cryptographic Key and Key Material Destruction (Destruction Timing)
EE:FCS_CKM_EXT.4(b)	Cryptographic Key and Key Material Destruction (Power Management)
EE:FCS_CKM_EXT.6	Cryptographic Key Destruction Types
EE:FCS_KYC_EXT.2	Key Chaining (Recipient)
EE:FCS_SNI_EXT.1	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
EE:FCS_VAL_EXT.1	Validation
EE:FDP_DSK_EXT.1	Protection of Data on Disk
EE:FMT_SMF.1	Specification of Management Functions
EE:FPT_KYP_EXT.1	Protection of Key and Key Material
EE:FPT_PWR_EXT.1	Power Saving States
EE:FPT_PWR_EXT.2	Timing of Power Saving States

Requirement	Title
EE:FPT_TST_EXT.1	TSF Testing
EE:FPT_TUD_EXT.1	Trusted Update
Selection based	
AA:FCS_CKM.1(b)	Cryptographic Key Generation (Symmetric Keys)
AA/EE:FCS_COP.1(a)	Cryptographic Operation (Signature Verification)
AA:FCS_COP.1(b)	Cryptographic Operation (Hash Algorithm)
AA:FCS_COP.1(c)	Cryptographic Operation (Keyed Hash Algorithm)
AA:FCS_COP.1(g)	Cryptographic Operation (Key Encryption)
AA:FCS_KDF_EXT.1	Cryptographic Key Derivation
AA:FCS_PCC_EXT.1	Cryptographic Password Construct and Conditioning
AA:FCS_RBG_EXT.1	Cryptographic Operation (Random Bit Generation)
AA:FCS_SMC_EXT.1	Submask Combining
EE:FCS_CKM.4(d)	Cryptographic Key Destruction (Software TOE, 3rd Party Storage)
EE:FCS_COP.1(b)	Cryptographic Operation (Hash Algorithm)
EE:FCS_COP.1(c)	Cryptographic Operation (Keyed Hash Algorithm)
EE:FCS_COP.1(f)	Cryptographic Operation (AES Data Encryption/Decryption)
EE:FCS_COP.1(g)	Cryptographic Operation (Key Encryption)
EE:FCS_KDF_EXT.1	Cryptographic Key Derivation
EE:FCS_RBG_EXT.1	Cryptographic Operation (Random Bit Generation)

5.3.1 Cryptographic Support (FCS)

AA:FCS_AFA_EXT.1 Authorization Factor Acquisition

AA: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 **AA:**FCS_PCC_EXT.1,
- an external Smartcard factor that is protecting a submask that is [generated by the TOE (using the RBG as specified in **AA:**FCS_RBG_EXT.1)] protected using [RSA with key size [3072

bits]], with user presence proved by presentation of the smartcard and [an OE defined PIN].

].

Application Note: This SFR has been modified by TD0759.

AA:FCS_AFA_EXT.2 Timing of Authorization Factor Acquisition

AA:FCS_AFA_EXT.2.1 The TSF shall reacquire the authorization factor(s) specified in **AA:**FCS_AFA_EXT.1 upon transition from any Compliant power saving state specified in **AA:**FPT_PWR_EXT.1 prior to permitting access to plaintext data.

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

AA:FCS_CKM.4.1(a) **Refinement:** The TSF **shall [erase] cryptographic keys and key material from volatile memory when transitioning to a Compliant power saving state as defined by AA:FPT_PWR_EXT.1** that meets the following: *a key destruction method specified in **AA:**FCS_CKM.4(d)*.

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

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

- **For volatile memory, the destruction shall be executed by a [**
  - **single overwrite consisting of [**
    - **zeroes,**

      ],

    ];
- **For non-volatile storage that consists of the invocation of an interface provided by the underlying platform that [**
  - **instructs the underlying platform to destroy the abstraction that represents the key]**

]

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

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

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

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

AA:FCS_CKM_EXT.4.1(b) **Refinement:** 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 AA:FPT_PWR_EXT.1.**

AA:FCS_KYC_EXT.1 Key Chaining (Initiator)

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

- intermediate keys originating from one or more submask(s) to the BEV using the following method(s): [
  - key derivation as specified in **AA:**FCS_KDF_EXT.1,
  - key combining as specified in **AA:**FCS_SMC_EXT.1,
  - key encryption as specified in **AA:**FCS_COP.1(g)]]

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

Application Note: Keys are combined per **AA:**FCS_SMC_EXT.1 to maintain an effective strength of 256 bits along the key chain.

AA:FCS_KYC_EXT.1.2 The TSF shall provide at least a [128 bit, 256 bit] BEV to [*the protected OS drive*] [

- without validation taking place].

Application Note: The TOE may provide either 128-bit or 256-bit BEVs depending on the specific license entitlement applied to the TOE during installation. The keychain remains the same in either case.

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

AA:FCS_SNI_EXT.1.1 The TSF shall [use salts that are generated by a [DRBG as specified in **AA:**FCS_RBG_EXT.1]].

AA:FCS_SNI_EXT.1.2 The TSF shall use [no nonces].

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

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

Application Note: This SFR has been modified by TD0760.

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

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

AA/EE:FCS_COP.1(a) Cryptographic Operation (Signature Verification)

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

- **RSA Digital Signature Algorithm with a key size (modulus) of [3072-bit];**

]

that meet the following: [

- ***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 29 9796-2, Digital signature scheme 2 or Digital Signature scheme 3, for RSA schemes*].**

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

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

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

AA:FCS_COP.1.1(c) **Refinement:** The TSF shall perform cryptographic *keyed-hash message authentication* in accordance with a specified cryptographic algorithm [**HMAC-SHA-384**] and cryptographic key sizes [***384 bits***] that meet the following: *ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"*.

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

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

AA:FCS_KDF_EXT.1 Cryptographic Key Derivation

AA:FCS_KDF_EXT.1.1 The TSF shall accept [a conditioned password submask] to derive an intermediate key, as defined in [

- NIST SP 800-132],

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

AA:FCS_PCC_EXT.1 Cryptographic Password Construct and Conditioning

AA:FCS_PCC_EXT.1.1 A password used by the TSF to generate a password authorization factor shall enable up to [*128*] characters in the set of {upper case characters, lower case characters, numbers, and *[“!”,“”“,“#”,“$”,“%”,“&”,“‘”,“(”,“)”,“*”,“+”,“,”,“.”,“/”,“:”,“;”,“<”,“=”,“>”,“?”,“@”,“[”,“]”,“^”,“_”,“`”,“{”,“|”,“}”,“~”,“-”]*} and shall perform Password-Based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC- [SHA-384], with [*[100,000]* iterations], and output cryptographic key sizes [256 bits] that meet the following: [NIST SP 800-132].

Application Note: This SFR has been modified by TD0764.

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

AA:FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with [NIST SP 800-90A] using [CTR_DRBG (AES)].

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

- [*1*] hardware-based noise source(s),]

with a minimum of [256 bits] of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 “Security Strength Table for Hash Functions”, of the keys and hashes that it will generate.

AA:FCS_SMC_EXT.1 Submask Combining

AA:FCS_SMC_EXT.1.1 The TSF shall combine submasks using the following method [exclusive OR (XOR)] to generate an *intermediary key or BEV*.

Application Note: Submask combining is used for dual factor authentication.

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

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

- **generate a DEK using the RBG as specified in EE:FCS_RBG_EXT.1,**

]

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

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

EE:FCS_CKM.4.1(a) **Refinement:** The TSF **shall [instruct the Operational Environment to clear] cryptographic keys and key material from volatile memory when transitioning to a Compliant power saving state as defined by EE:FPT_PWR_EXT.1** that meets the following: *a key destruction method specified in **EE:**FCS_CKM_EXT.6*.

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

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

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

EE: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 **EE:**FPT_PWR_EXT.1.

EE:FCS_CKM_EXT.6 Cryptographic Key Destruction Types

EE:FCS_CKM_EXT.6.1 The TSF shall use **[EE:FCS_CKM.4(d)]** key destruction methods.

EE:FCS_KYC_EXT.2 Key Chaining (Recipient)

EE:FCS_KYC_EXT.2.1 The TSF shall accept a BEV of at least [128 bits, 256 bits] from *the AA*.

EE: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): [

- key derivation as specified in **EE:**FCS_KDF_EXT.1,
- key encryption as specified in **EE:**FCS_COP.1(g)]

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

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

EE:FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with [*NIST SP 800-90A*] using [HMAC_DRBG (any)].

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

- [*1*] hardware-based noise source(s),]

with a minimum of [256 bits] of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 “Security Strength Table for Hash Functions”, of the keys and hashes that it will generate.

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

EE:FCS_SNI_EXT.1.1 The TSF shall use [use salts that are generated by a [DRBG provided by the host platform].

EE:FCS_SNI_EXT.1.2 The TSF shall use [no nonces].

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

- XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer].

EE:FCS_VAL_EXT.1 Validation

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

- decrypt a known value using the [intermediate key] as specified in FCS_COP.1(f) and compare it against a stored known value;

]

EE: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*.

EE:FCS_VAL_EXT.1.3 The TSF shall [

- require power cycle/reset the TOE after [*an administrator configurable number between 1 and 20*] of consecutive failed validation attempts].

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

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

- For volatile memory, the destruction shall be executed by a [
  - single overwrite consisting of [
    - zeroes]]
- For non-volatile storage that consists of the invocation of an interface provided by the underlying platform that [
  - instructs the underlying platform to destroy the abstraction that represents the key]

]

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

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

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

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

EE:FCS_COP.1.1(c) **Refinement:** The TSF shall perform cryptographic *message authentication* in accordance with a specified cryptographic algorithm [**HMAC-SHA-256, HMAC-SHA-384**] and cryptographic key sizes [***256, 384 bits* used in** [**HMAC**]] that meet the following: [**ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"**].

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

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

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

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

EE:FCS_KDF_EXT.1 Cryptographic Key Derivation

EE:FCS_KDF_EXT.1.1 The TSF shall accept [imported submask] to derive an intermediate key, as defined in [

- NIST SP 800-132],

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

5.3.2 User Data Protection (FDP)

EE:FDP_DSK_EXT.1 Protection of Data on Disk

EE:FDP_DSK_EXT.1.1 The TSF shall perform Full Drive Encryption in accordance with **EE:**FCS_COP.1(f), such that the drive contains no plaintext protected data.

EE:FDP_DSK_EXT.1.2 The TSF shall encrypt all protected data without user intervention.

5.3.3 Security Management (FMT)

AA:FMT_MOF.1 Management of Functions Behavior

AA:FMT_MOF.1.1 The TSF shall restrict the ability to modify the behaviour of the functions *use of Compliant power saving state* to *authorized users*.

AA:FMT_SMF.1 Specification of Management Functions

AA:FMT_SMF.1.1 **Refinement:** 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 values or set of authorization values used within the supported authorization method,

d) initiate TOE firmware/software updates,

e) **[configure authorization factors, disable key recovery functionality]**.

Application Note: This SFR has been modified by TD0767.

AA:FMT_SMR.1 Security Roles

AA:FMT_SMR.1.1 The TSF shall maintain the roles *authorized user*.

AA:FMT_SMR.1.2 The TSF shall be able to associate users with roles.

EE:FMT_SMF.1 Specification of Management Functions

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

*a) change the DEK, as specified in **EE:**FCS_CKM.1, when re-provisioning or when commanded,*

*b) erase the DEK, as specified in **EE:**FCS_CKM.4(a),*

c) *initiate TOE firmware/software updates,*

d) **[no other functions,]**.

5.3.4 Protection of the TSF (FPT)

AA:FPT_KYP_EXT.1 Protection of Key and Key Material

AA:FPT_KYP_EXT.1.1 The TSF shall [

- only store keys in non-volatile memory when wrapped, as specified in **AA:**FCS_COP.1(d), or encrypted, as specified in **AA:**FCS_COP.1(g) or FCS_COP.1(e)]

AA:FPT_PWR_EXT.1 Power Saving States

AA:FPT_PWR_EXT.1.1 The TSF shall define the following Compliant power saving states: [S4, G2(S5), G3]

Application Note: The S4 power saving state is only supported on Windows platforms.

AA:FPT_PWR_EXT.2 Timing of Power Saving States

AA:FPT_PWR_EXT.2.1 For each Compliant power saving state defined in **AA:**FPT_PWR_EXT.1.1, the TSF shall enter the Compliant power saving state when the following conditions occur: user-initiated request, [*[as prompted by the protected OS]*].

AA:FPT_TUD_EXT.1 Trusted Update

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

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

AA:FPT_TUD_EXT.1.3 **Refinement:** The TSF shall verify updates to the TOE software using a digital signature **as specified in AA:FCS_COP.1(a)** by the manufacturer prior to installing those updates.

EE:FPT_KYP_EXT.1 Protection of Key and Key Material

EE:FPT_KYP_EXT.1.1 The TSF shall [

- only store keys in non-volatile memory when wrapped, as specified in FCS_COP.1(d), or encrypted, as specified in **EE:**FCS_COP.1(g) or FCS_COP.1(e)].

EE:FPT_PWR_EXT.1 Power Saving States

EE:FPT_PWR_EXT.1.1 The TSF shall define the following Compliant power saving states: [S4, G2(S5), G3]

Application Note: The S4 power saving state is only supported on Windows platforms.

EE:FPT_PWR_EXT.2 Timing of Power Saving States

EE:FPT_PWR_EXT.2.1 For each Compliant power saving state defined in **EE:**FPT_PWR_EXT.1.1, the TSF shall enter the Compliant power saving state when the following conditions occur: user-initiated request, [*[as prompted by the protected OS]*].

EE:FPT_TST_EXT.1 TSF Testing

EE:FPT_TST_EXT.1.1 The TSF shall run a suite of the following self-tests [during initial start-up (on power on)] to demonstrate the correct operation of the TSF: [*software integrity test, cryptographic module tests*].

EE:FPT_TUD_EXT.1 Trusted Update

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

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

EE:FPT_TUD_EXT.1.3 **Refinement:** The TSF shall verify updates to the TOE [**software**] using a [digital signature **as specified in EE:FCS_COP.1(a)**] by the manufacturer prior to installing those updates.

5.4 Assurance Requirements

20 The TOE security assurance requirements are summarized in Table 11.

Table 11: Assurance Requirements

Assurance Class	Components	Description
Security Target Evaluation	ASE_CCL.1	Conformance Claims
	ASE_ECD.1	Extended Components Definition
	ASE_INT.1	ST Introduction
	ASE_OBJ.1	Security Objectives for the operational environment
	ASE_REQ.1	Stated Security Requirements
	ASE_SPD.1	Security Problem Definition
	ASE_TSS.1	TOE Summary Specification
Development	ADV_FSP.1	Basic Functional Specification
Guidance Documents	AGD_OPE.1	Operational User Guidance
	AGD_PRE.1	Preparative Procedures
Life Cycle Support	ALC_CMC.1	Labelling of the TOE
	ALC_CMS.1	TOE CM Coverage
Tests	ATE_IND.1	Independent Testing - sample
Vulnerability Assessment	AVA_VAN.1	Vulnerability Survey

21 In accordance with section 6.1 of the CPP_FDE_AA, the following refinement is made to ASE:

a) **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 [Entropy Essay].**

6 TOE Summary Specification

22 The following sections describe how the TOE fulfils each SFR included in section 5.3.

6.1 Context

6.1.1 Core TOE Concepts

23 The following are core concepts and TOE components relevant to understanding the TSS:

a) **Installers.** The TOE is installed in two steps – first the Encryption Engine / Driver is installed on the Host OS that is to be protected. Second, the PBA is installed from a bootable device such as a USB drive, DVD or from a network share (such as executing via PXE boot). After installation the system will boot to the PBA logon screen and the user will enter the user's credentials and log into the PBA which after successful authentication will initiate chain-boot to the Protected OS.

b) **GUI.** The TOE provides a local GUI for PBA (drive unlock via username/password and/or smartcard) and TOE / user management.

c) **User Management.** The TOE enforces role-based access control with the following roles defined:

i) **Admin.** Can unlock the drive, add other users and update TOE firmware.

ii) **Security Officer.** Can unlock the drive, perform wipe-disk function and delete logs.

iii) **Login User.** Can unlock the drive.

iv) **Helpdesk.** Can view logs and reset user passwords.

d) **Protected OS.** The Host OS or Hypervisor environment on the drive that is booted after successful PBA authentication.

6.1.2 Key Management

24 The following sections describe the fundamental key management aspects of the TOE. The figures below depict the resulting keychains designed with sufficient strength to protect a 256-bit DEK. All key outputs are of a strength at least that of the BEV.

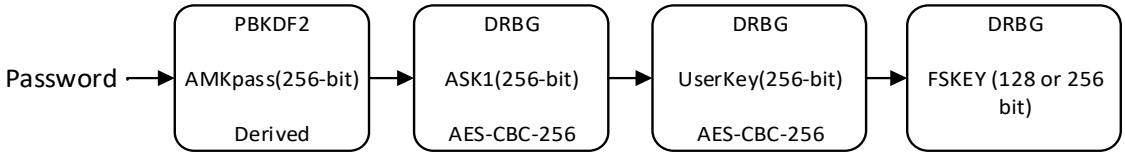

Figure 1: Keychain for Password Authorization

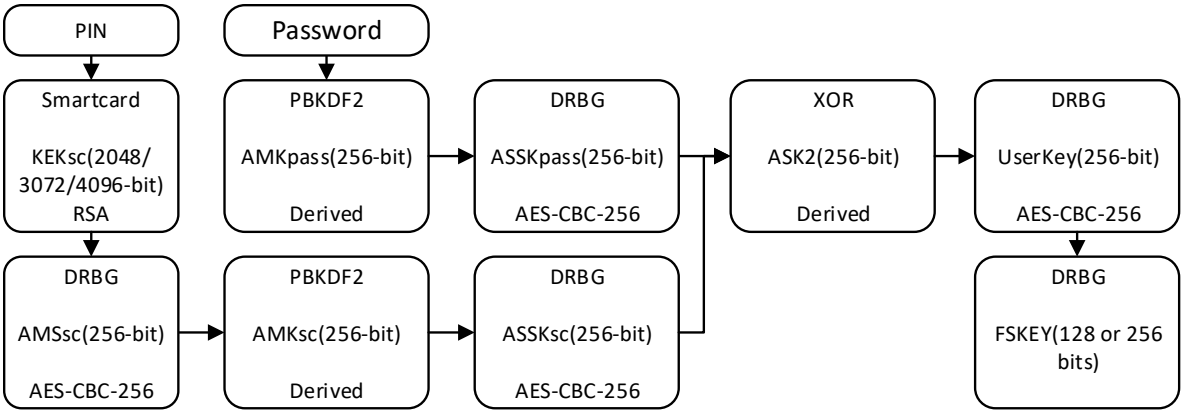

Figure 2: Keychain for Dual Factor Authorization

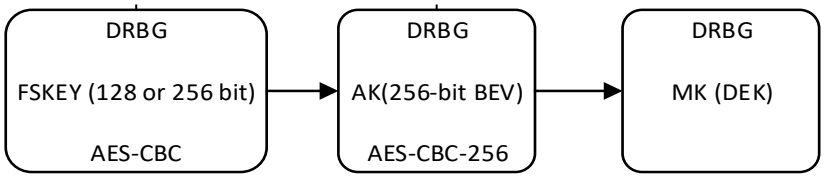

Figure 3: Keychain for EMK

6.1.2.1 Authentication Keys (BEV)

25 The TOE generates and manages the Authentication Keys (AKs) used to unlock a FDE (AKs are the BEV referred to by the CPP_FDE_AA):

a) **ADMIN AK.** This is the Administrative AK on FDE. It manages administrative logins and the locking and unlocking of the FDE.

b) **LOGIN USER AK.** This is the Locking AK. It manages the locking and unlocking of the FDE.

26 During installation, the TOE generates 256-bit keys. Depending on the license entitlement applied to the TOE (which affects configuration of ADMIN and LOGIN user accounts), 128-bit keys may also be used. The AKs are encrypted using AES and stored in the TOE's database.

6.1.2.2 KEKs

27 As shown in Figure 1, Figure 2, and Figure 3 the TOE uses a chain of KEKs to the BEV (AK):

a) **FSKEY.** 256-bit or 128-bit AES key generated by the TOE and used to encrypt the AK.

b) **ASK.** 256-bit AES key used to encrypt UserKey. This key is derived differently depending on the authorization factors in use:

i) **Username and Password.** 256-bit submask generated by the TOE. ASSKpass is encrypted using AES-CBC-256 and stored in the TOE database.

ii) **Dual Factor.** ASK is a combined key which is XOR of:

- **ASSKpass.** 256-bit submask generated by the TOE.

- **ASSKsc.** 256-bit submask generated by the TOE. ASSKsc is encrypted using AES-CBC-256 and stored in the TOE database.

c) **AMKpass.** 256-bit derived from the user's password via PBKDF2. AMKpass is KEK for ASSKpass.

d) **AMKsc.** 256-bit derived from the AMSsc via PBKDF2 256-bit. AMKsc is KEK for ASSKsc

e) **AMSsc.** 256-bit generated by the TOE. AMSsc is key material for AMKsc. AMSsc is encrypted by KEKsc and stored in the database.
Note: (KTS-OAEP) encryption is performed outside of the smartcard using a public key from the smartcard. Decryption is performed using the private key within the smartcard.

f) **KEKsc.** RSA private key stored inside a smartcard and used (by the smartcard) to encrypt AMSsc.

28 KEKs are further delineated depending on the type of user.

6.1.2.3 DEK

29 The Data Encryption Key (DEK) is the Master Key (MK). DEK is encrypted with the AK (BEV) as shown in Figure 3 to create encrypted MK (EMK). The encrypted MK is stored in the metadata using anti-forensic stripes.

6.1.3 Authentication / Drive Unlock Flow

30 At a high-level, the basic start-up and authentication flow is as follows for both Linux and Windows:

a) When the TOE starts up, it boots from the PBA code partition, launches the PBA GUI, copies and de-obfuscates the database from the PBA data partition and mounts it in RAM. The user enters their username and password, and, if dual factor authentication is configured, presents a smartcard and PIN.

b) Depending on the authentication method:

i) Validate username against the database

ii) For smartcard, authenticate PIN against the smartcard and pass encrypted AMSsc to the smartcard for decryption.

c) The TOE derives AMKpass and AMKsc and decrypts ASSKpass and ASSKsc

d) The TOE calculates ASK from ASSKpass and ASKsc

e) UserKey is decrypted by ASK

f) UserKey decrypts FSKEY.

g) The TOE uses the FSKEY to decrypt the appropriate AK based on the user's role.

h) The TOE uses the AK to decrypt BEV

i) The BEV is then passed to FDE-LUKS (with relevant cryptsetup commands). The BEV is used to decrypt EMK and after validating the correctness of MK (DEK), the drive containing the host OS is unlocked, allowing chain-boot to the host OS.

j) The BEV is then passed to FDE – Windows (with relevant KLC CDOCryptsetup commands). The BEV is used to decrypt EMK and after validating the correctness of MK (DEK), the drive containing the host OS is unlocked, allowing chain-boot to the host OS.

6.2 Cryptographic Support (FCS)

6.2.1 AA:FCS_AFA_EXT.1 Authorization Factor Acquisition

31 The TOE supports the use of username/password and smartcards (dual factor).

6.2.1.1 Username / Password

32 The password authentication process is as follows for both Linux and Windows:

a) The user enters their username and password

b) The TOE will attempt to locate the username in the database

c) The TOE will return a generic authentication error if the username is not found

d) The TOE will compare a SHA384 hash of username + password. If there is a mismatch with value computed at enrolment, the TOE will return a generic authentication error

e) The TOE will perform PBKDF2 on the password and derive AMKpass

f) The TOE will use AMKpass to decrypt ASSKpass

g) The TOE will use ASSKpass as ASK1 to decrypt UserKey

h) The TOE will use UserKey to decrypt FSKEY

i) The TOE will use FSKEY to decrypt the AK

j) The TOE will use AK to decrypt DEK

k) The TOE will use the BEV to unlock a FDE-LUKS or FDE-Windows

l) The BEV is used to decrypt FDE-LUKS EMK. A stored MK (DEK) digest is used to verify with the digest computed using PBKDF2 of MK with a different set of parameters. Thus, the correctness of MK is determined to unlock FDE-LUKS.

m) If FDE-LUKS unlock fails, the TOE will return a generic authentication error. If FDE-LUKS unlock succeeds, the user is authenticated / authorized.

n) If FDE-Windows unlock fails, the TOE will return a generic authentication error. If FDE-Windows unlock succeeds, the user is authenticated / authorized.

6.2.1.2 Dual Factor

33 The TOE supports an external smartcard factor that is at least the same bit-length as the DEK (256-bit) - ASKsc is 256-bits in length.

34 The TOE generates ASSKpass using the RBG as specified in FCS_RBG_EXT.1 and stores it in encrypted form using AES-CBC-256.

35 The TOE generates ASSKsc using the RBG as specified in FCS_RBG_EXT.1 and stores it in encrypted form using AES-CBC-256.

36 ASSKpass and ASSKsc are submasks for combining into ASK2 for dual factor authentication.

37 The dual factor authentication process is as follows for both Linux and Windows:

a) The user enters their username and password

b) The user presents a smartcard by inserting it into the FIPS 201 PIV-CAC compliant reader and will be prompted to enter the smartcard PIN

c) The TOE will attempt to locate the username in the database

d) The TOE will return a generic authentication error if the username is not found

e) The TOE will compare a SHA384 hash of username + password. If there is a mismatch with value computed at enrolment, the TOE will return a generic authentication error.

f) The smartcard will verify the PIN, if verification fails, the TOE will return a generic authentication error

g) The TOE will generate AMKpass via PBKDF2 which is used to decrypt ASSKsc

h) If PIN verification succeeds, the TOE will pass the encrypted AMSsc to smartcard for decryption with KEKsc

i) The smartcard returns the decrypted AMSsc to the TOE

j) The TOE generates AMKsc via PBKDF2 which is used to decrypt ASSKsc

k) The TOE combines ASSKpass and ASSKsc via XOR to form ASK2

l) The TOE decrypts UserKey with ASK2

m) The TOE decrypts FSKEY with UserKey

n) The TOE will use FSKEY to decrypt the AK (BEV)

o) The TOE will use AK (BEV) to decrypt DEK

p) The TOE will use the BEV to unlock FDE-LUKS or FDE-Windows

q) The BEV is used to decrypt FDE-LUKS EMK. A stored MK (DEK) digest is used to verify with the digest computed using PBKDF2 of MK with a different set of parameters. Thus, the correctness of MK is determined to unlock FDE-LUKS (same for both Linux and Windows).

r) If FDE-LUKS unlock fails, the TOE will return a generic authentication error. If FDE-LUKS unlock succeeds, the user is authenticated / authorized.

s) If FDE-Windows unlock fails, the TOE will return a generic authentication error. If FDE-Windows unlock succeeds, the user is authenticated / authorized.

6.2.2 AA:FCS_AFA_EXT.2 Timing of Authorization Factor Acquisition

38 The user must authenticate via password or dual factor to gain access to user data after the TOE entered a Compliant power saving state described by FPT_PWR_EXT.1 below.

6.2.3 AA:FCS_CKM.1(b) Cryptographic Key Generation (Symmetric Keys)

39 The TOE leverages the CTR_DRBG provided by BoringCrypto to generate the following 256-bit AES keys:

a) All Admin and User AK’s

b) FSKEY (128-bit or 256-bit)

c) ASSKpass

d) ASSKsc

40 The TOE can generate 256-bit or 128-bit MK/BEVs depending on the license entitlement applied during installation.

6.2.4 AA:FCS_CKM.4(a) Cryptographic Key Destruction (Power Management)

41 The TOE erases cryptographic keys and key material from volatile memory when transitioning to a compliant power saving state as defined by FPT_PWR_EXT.1 which meets the key destruction method specified in AA:FCS_CKM.4(d). Users initiate a request to enter a power saving state by interacting with the Host OS and occurs immediately, as prompted by the protected OS, and without delay.

42 Temporary keys are tracked by allocated memory throughout their usage lifecycle until destruction.

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

43 The TOE destroys cryptographic keys in volatile memory according to the execution of a single overwrite operation consisting of zeroes. For non-volatile storage, the TOE destroys cryptographic keys via the invocation of an interface provided by the underlying platform which instructs the underlying platform to destroy the abstraction that represents the key.

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

44 The TOE performs cryptographic key destruction operations that destroy all keys and key material when they are no longer needed. Additional information on how these operations are performed can be found in Section 6.2.5

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

45 When the TOE transitions to a power saving state as defined by FPT_PWR_EXT.1, all key material, BEV, and authentication factors stored in plaintext are destroyed.

6.2.8 AA/EE:FCS_COP.1(a) Cryptographic Operation (Signature Verification)

46 The TOE performs signature verification using RSA 3072 with SHA-384 for trusted updates as follows:

a) TOE updates are signed with the KLC code signing private key

b) The obfuscated public key is embedded in the TOE binary

c) When the user triggers the TOE update from the GUI, the TOE verifies the digital signature using the embedded public key

d) If the digital signature verification succeeds, the upgrade process is carried out

e) If the digital signature verification fails, the upgrade process is aborted, and an error is displayed to the user.

6.2.9 AA:FCS_COP.1(b) Cryptographic Operation (Hash Algorithm)

47 The TOE makes use of SHA-384 for digital signature verification.

48 The TOE makes use of SHA-384 for PBKDF.

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

49 The TOE implements HMAC-SHA-384 with the following characteristics:

a) **Key length.** 384 bits.

b) **Block size.** 1024 bits.

c) **MAC length.** 384 bits.

6.2.11 AA:FCS_COP.1(g) Cryptographic Operation (Key Encryption)

50 The TOE performs key encryption using AES-CBC-128 or AES-CBC-256.

6.2.12 AA:FCS_KDF_EXT.1 Cryptographic Key Derivation

51 Passwords are conditioned via PBKDF2 using HMAC-SHA-384 with 100,000 iterations, resulting in a 256-bit key in accordance with NIST SP 800-132.

6.2.13 AA:FCS_KYC_EXT.1 Key Chaining (Initiator)

52 The TOE key chain is described at section 6.1.2.

53 The TOE maintains a key chain of intermediate keys originating from one or more submasks to the BEV while maintaining an effective key strength of 128 bits or 256 bits for symmetric keys.

54 The TOE supports a default BEV (AK) size of 256 bits, however specific licensing entitlements (by request to KLC) can support 128-bit keys.

6.2.14 AA:FCS_PCC_EXT.1 Cryptographic Password Construct and Conditioning

55 The TOE implements a configurable password policy with the following options:

a) Minimum Length (8 – 128)

b) Require at least one uppercase

c) Require at least one lowercase

d) Require at least one numeric

e) Require at least one special character
(“!”,“”“,“#”,“$”,“%”,“&”,“’”,“(”,“)”,“*”,“+”,“,”,“-”,“.”,“/”,“:”,“;”,“<”,“=”,“>”,“?”,“@”,“[”,“]”,“^”,“_”,“`”,“{”,“|”,“}”,“~”,“-”)

f) History (Can repeat same password after configurable number of times)

g) Number of consecutive failed validation attempts before reboot is required

56 Passwords are conditioned via PBKDF2 using HMAC-SHA-384 with 100,000 iterations, resulting in a 256-bit key in accordance with NIST SP 800-132.

6.2.15 AA:FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

57 The TOE uses a deterministic random bit generator (DRBG) that complies with NIST SP 800-90A for AA cryptographic operations. The BoringSSL cryptographic library is used in support of AA cryptographic operations. BoringSSL implements CTR-

DRBG(AES 256) seeded with 256-bits of entropy. The source of entropy for the TOE is the Intel DRNG (RDRAND).

58 The TOE performs NIST SP 800-90A compliant health testing to ensure that entropy sources continue to operate as expected. The Intel DRNG performs validation tests at system startup in addition to DRNG Online Health Tests (OHTs) and Built-In Self Tests (BISTs) which include entropy source tests and Known Answer Tests (KAT) that comply with NIST SP 800-90A Section 11.3. The BoringSSL cryptographic module performs DRBG Continuous Tests when a random value is requested from the DRBG.

6.2.16 AA:FCS_SMC_EXT.1 Submask Combining

59 As described in Section 6.1.2, the order of submask combining operations is as follows: ASSKpass and ASSKsc are first XORed together which forms the intermediate key ASK that is then used with dual factor authentication configurations. No other ordering requirements are applicable to this process.

60 The output key length is at least that of the BEV as shown in Section 6.1.2.

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

61 The TOE generates 32 byte salts at the time of encryption using the RAND_bytes function provided by the DRBG specified AA:FCS_RBG_EXT.1. The salts are then stored in a database for use during decryption.

62 For DEK, AES-CBC is used and the IV is generated randomly by the DRBG using the RAND_bytes function and are appended to the encrypted data.

63 The TOE does not make use of nonces.

6.2.18 EE:FCS_CKM.1(c) Cryptographic Key Generation (Data Encryption Key

64 The cryptsetup/CDOcryptsetup utility (during CipherDriveOne Kryptr (CDO) installation) invokes the DRBG specified in AA:FCS_RBG_EXT.1 to generate the Master Key (DEK) which is encrypted using AES-CBC cipher and the key used for encryption is PBKDF2 of BEV. The encrypted MK (EMK) is then stored in anti-forensic (AF) stripes. The EE layer encrypts the block device using the decrypted master key (DEK) and AES-XTS cipher.

6.2.19 EE:FCS_CKM.4(a) Cryptographic Key Destruction (Power Management)

65 Transitioning to a compliant power saving state will trigger the destruction of keys or key material in volatile memory. Users initiate a request to enter a power saving state by interacting with the Host OS. The TOE will instruct the Protected OS to enter a compliant power saving state immediately and destroy cryptographic keys and key material from volatile memory without delay. Temporary keys and the allocated memory in which they reside are tracked during the usage lifecycle until destruction.

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

66 The TOE destroys all keys and keying material in volatile memory when no longer needed, including when the TOE transitions into a compliant power-saving state or is shut down, and when keys are overwritten with new ones.

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

67 When the TOE transitions to a compliant power-saving state, all key material, BEV, and authentication factors stored in plaintext are destroyed.

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

68 The TOE destroys cryptographic keys in volatile memory according to the execution of a single overwrite operation consisting of zeros. On Linux platforms, Cryptsetup uses the crypt_safe_free function which zeroes all memory. On Windows platforms, memory is initialized to zeros before destruction.

69 For non-volatile storage, the TOE destroys cryptographic keys via the invocation of an interface provided by the underlying platform which instructs the underlying platform to destroy the abstraction that represents the key.

6.2.23 EE:FCS_CKM_EXT.6 Cryptographic Key Destruction Types

70 The TOE implements the key destruction methods and types described in FCS_CKM.4(d). More information can be found in Section 6.2.22.

6.2.24 EE:FCS_COP.1(b) Cryptographic Operation (Hash Algorithm)

71 The TOE makes use of SHA-384 for digital signature verification and PBKDF2.

72 The TOE makes use of SHA-256 in the HMAC_DRBG implemented by Libgcrypt in support of EE cryptographic operations from the PBA.

73 These algorithms are not configurable within the TOE.

6.2.25 EE:FCS_COP.1(c) Cryptographic Operation (Message Authentication)

74 The TOE implements HMAC-SHA-256 and HMAC-SHA-384 with the following characteristics:

a) **Key length.** 256 bits, 384 bits.

b) **Block size.** 512 bits, 1024 bits.

c) **MAC length.** 256 bits, 384 bits.

75 These algorithms are not configurable within the TOE.

76 In Linux, EE uses the Linux Crypto API for PBKDF2.

77 In Windows, EE leverages the Windows CryptoAPI

78 In PBA, EE leverages libgcrypt library.

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

79 Disk encryption is performed using AES-XTS 256 by default (AES-XTS 128 may be used depending on the specific license entitlement applied to the TOE during installation). The TOE relies on the Operational Environment as follows:

a) **Linux.** CDO performs AES-XTS disk encryption leveraging Linux Crypto API.

b) **Windows.** CDO-encryption/hibernation drivers performs AES-XTS disk encryption leveraging the Windows CryptoAPI.

80 Initial encryption/decryption operations that are performed during install/uninstall for both Linux/Windows protected OS leverage libgcrypt.

6.2.27 EE:FCS_COP.1(g) Cryptographic Operation (Key Encryption)

81 Windows and Linux environments both use the AES-CBC algorithm for MK encryption functions and generates key sizes of 128 bits or 256 bits.

6.2.28 EE:FCS_KDF_EXT.1 Cryptographic Key Derivation

82 The BEV is conditioned via PBKDF2 using HMAC-SHA-384 with 100,000 iterations, resulting in a 256-bit key in accordance with NIST SP 800-132. This process is used by both cryptsetup for Linux and CDOCryptsetup for Windows.

6.2.29 EE:FCS_KYC_EXT.2 Key Chaining (Recipient)

83 The Encryption Engine (EE) accepts a BEV of 128 bits or 256 bits from the AA and maintains a chain of intermediary keys originating from the BEV to the DEK by using key derivation as specified in EE:FCS_KDF_EXT.1, and key encryption as specified in EE:FCS_COP.1(g). These methods ensure an effective strength of 128 bits or 256 bits for symmetric keys is maintained.

6.2.30 EE:FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

84 The TOE uses a deterministic random bit generator (DRBG) that complies with NIST SP 800-90A for EE cryptographic operations.

85 The Libgcrypt cryptographic library is used in support of EE cryptographic operations from the PBA and implements HMAC-DRBG(SHA-256) seeded with 384-bits of entropy. The source of entropy for the TOE is Intel DRNG (RDRAND).

86 The TOE performs NIST SP 800-90A compliant health testing to ensure that entropy sources continue to operate as expected. The Intel DRNG performs validation tests at system startup in addition to DRNG Online Health Tests (OHTs) and Built-In Self Tests (BISTs) which include entropy source tests and Known Answer Tests (KAT) that comply with NIST SP 800-90A Section 11.3. The Libgcrypt cryptographic module performs DRBG Continuous Tests when a random value is requested from the DRBG.

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

87 Salts are generated by the DRBG provided by the host platform and is invoked using the *gcry_randomize* function.

88 Tweak values are required for data encryption using AES-XTS and start from a non-negative integer that is incremented by sector number to maintain sequential order for both Windows and Linux hosts.

89 The TOE does not make use of nonces.

6.2.32 EE:FCS_VAL_EXT.1 Validation

90 The TOE validates the BEV by using it to derive an intermediate key using PBKDF2. The intermediate key is then used to decrypt a known value (using AES-CBC-256, per FCS_COP.1(f)) which is compared to a stored known value. Additional details are provided in the KMD.

91 Successful validation of BEV per above is required prior to allowing decryption of the drive and granting access to any TSF data after exiting a compliant power saving state. If a validation attempt of the BEV is unsuccessful, the TOE increments a failed authentication counter.

92 After a configurable number of failed BEV validation/authentication attempts is reached, the user will be locked out until the system is rebooted at which point the counter is reset. An administrator can set this threshold to a value between 1 and 20 failed attempts.

6.2.33 Cryptographic Libraries and CAVP Mapping

93 The TOE incorporates the following cryptographic libraries:

a) **BoringSSL(fips-20220613) in FIPS mode.** All cryptographic functions performed for AA within the PBA.

b) **Libgcrypt (1.9.4) in FIPS mode.** Provides cryptographic functions in support of EE operations from the PBA.

94 The Operational Environment consists of the following operating systems:

a) **Windows 10 & 11.** For Windows based systems, the TOE makes use of the Windows CryptoAPI for data encryption/decryption operations.

b) **Red Hat Enterprise Linux 8 & 9.** For Linux based systems, the TOE makes use of the Linux Crypto API library for data encryption/decryption operations.

95 **Note:** Initial EE encryption/decryption operations during installation/uninstallation and lock/unlock of partitions from the PBA use libgcrypt.

Table 12: CAVP Mapping

SFR	Capability	Cryptographic Library	CAVP Certificate
AA/EE:FCS_COP.1(a)	RSA (186-4) SigVer 3072	BoringCrypto	A5166
AA:FCS_COP.1(b)	SHA-384	BoringCrypto	A5166
AA:FCS_COP.1(c)	HMAC-SHA-384	BoringCrypto	A5166
AA:FCS_COP.1(g)	AES-CBC (128, 256)	BoringCrypto	A5166
AA:FCS_RBG_EXT.1	CTR_DRBG (AES-256)	BoringCrypto	A5166

SFR	Capability	Cryptographic Library	CAVP Certificate
EE:FCS_COP.1(b)	SHA-384	Linux Crypto API	A3241
	SHA-256 SHA-384	Libgcrypt (PBA)	A3230
	SHA-384	Windows CryptoAPI	A3231
EE:FCS_COP.1(c)	HMAC-SHA-384	Linux Crypto API	A3241
	HMAC-SHA-256 HMAC-SHA-384	Libgcrypt (PBA)	A3230
	HMAC-SHA-384	Windows CryptoAPI	A3231
EE:FCS_COP.1(f)	AES-XTS (128, 256)	Linux Crypto API	A3241
	AES-XTS (128, 256)	Libgcrypt (PBA)	A3230
	AES-XTS (128, 256)	Windows CryptoAPI	A3231
EE:FCS_COP.1(g)	AES-CBC (128, 256)	Linux Crypto API	A3241
	AES-CBC (128, 256)	Libgcrypt (PBA)	A3230
	AES-CBC (128, 256)	Windows CryptoAPI	A3231
EE:FCS_RBG_EXT.1	HMAC-DRBG (SHA-256)	Libgcrypt (PBA)	A3230

6.3 User Data Protection (FDP)

6.3.1 EE:FDP_DSK_EXT.1 Protection of Data on Disk

96 The TOE performs full drive encryption in accordance with EE:FCS_COP.1(f) on all disk drives specified at installation, such that the entire drive(s) are encrypted and contain no plaintext protected data. The TOE encrypts all protected data using AES without user intervention per sections 6.3.1.1 and 6.3.1.2 below. All protected data is written to disk in blocks per standard operating conditions of the AES algorithm.

97 All partitions of each protected disk are encrypted except for the ESP partition on Red Hat systems. (Note: ESP partition is not used post installation). The integrity of GPT partition structures are maintained.

98 All standard methods of accessing a protected disk drive via the host platforms operating system will pass through these functions without exception.

6.3.1.1 Linux

99 The TOE Linux Encryption Engine makes use of LUKS encryption.

100 The Encryption Engine installer uses the cryptsetup utility with specified encryption parameters for initial encryption. During runtime, dm-crypt which is a transparent disk encryption subsystem in the Linux kernel, performs the disk encryption/decryption.

6.3.1.2 Windows

101 The TOE Windows Encryption Engine (EE) makes use of CipherDriveOne Kryptr (CDO) encryption.

102 The CDO installer uses the CDOCryptsetup utility with specified encryption parameters for initial encryption. During runtime, CDO drivers provide transparent disc encryption subsystem for Windows that performs the disk encryption/decryption.

6.4 Security Management (FMT)

6.4.1 AA:FMT_MOF.1 Management of Functions Behavior

103 The TOE does not allow any modification related to power saving states.

6.4.2 AA:FMT_SMF.1 Specification of Management Functions

104 The TOE sends the request to the Encryption Engine to change the DEK in the following manner:

a) **Linux.** On user's request, luksAddKey and luksKillSlot commands are sent to the CDOCryptsetup engine using the Admin AK.

b) **Windows.** On user's request, CDO sends-ChangeDEK command to CDOCryptsetup engine using the Admin AK.

105 The TOE sends the request to the Encryption Engine to cryptographically erase the DEK in the following manner:

a) **Linux.** On user's request for cryptographic erase of the DEK, luksAddKey command is sent to the cryptsetup engine using the Admin AK with the value of a new random key followed by luksKillSlot for Admin and User slots.

b) **Windows.** On user's request for cryptographic erase of the DEK, CDO sends ChangeDEK command to CDOCryptsetup engine using the Admin AK

Note: Successful erase of the disk will also result in erase of DEK.

106 The TOE GUI may be used by the user to change their password. The TOE GUI may also be used for new smartcard enrollments with a changed PIN.

107 The TOE GUI (maintenance screen) can be used to initiate updates. Key recovery functionality (export configuration or backup database) can be disabled at install time (using '-n noexport' as one of the command-line parameters) or recovery can be administratively disabled at runtime (by setting the appropriate configuration item in the Settings Console as the Security Officer). The settings console can also be used to configure authorization factors.

6.4.3 AA:FMT_SMR.1 Security Roles

108 The TOE restricts access to authorized users.

6.4.4 EE:FMT_SMF.1 Specification of Management Functions

109 The TOE performs changes to the DEK as follows:

a) **Linux.** On user's request, CDO Kryptr sends luksAddKey and luksKillSlot commands to cryptsetup engine using the Admin AK to destroy the keys.

b) **Windows.** On user's request, CDO Kryptr sends ChangeDEK command to CDOCryptsetup engine using the Admin AK.The engine then wipes the key area with random data before deleting it.

110 Descriptions of how the TOE erases the DEK and initiates updates are provided in section 6.4.2.

111 The process for initiating TOE updates is conducted manually via the CDO Kryptr UI.

6.5 Protection of the TSF (FPT)

6.5.1 FPT_KYP_EXT.1 Protection of Key and Key Material (AA & EE)

112 The TOE only stores keys in non-volatile memory when encrypted as specified in FCS_COP.1(g).

113 Additional information on key storage can be found in section 6.1.2

6.5.2 FPT_PWR_EXT.1 Power Saving States (AA & EE)

114 The TOE supports the following Compliant power saving states:

a) **S4.** In this state, the system appears to be off and consumes the lowest power. While transitioning to this state from higher power, it may save the contents of the volatile memory to a file. When the system restarts, it will load the contents of the file for a quick boot only after CipherDriveOne Kryptr is invoked for authentication/authorization. The S4 power saving state is only supported on Windows platforms.

b) **G2(S5).** In this state, the system appears to be off and involves a complete shutdown and boot process and hence PBA will be invoked for authentication/authorization.

c) **G3.** In this state, the system is completely off and does not consume any power. The system returns to the working state only after a complete reboot and hence PBA will be invoked for authentication/authorization.

6.5.3 FPT_PWR_EXT.2 Timing of Power Saving States (AA & EE)

115 The TOE enters a compliant power saving state as prompted by the protected OS and user-initiated requests.

6.5.4 FPT_TUD_EXT.1 Trusted Update (AA & EE)

116 Update files for AA are obtained securely from the KLC customer web portal and are digitally signed via RSA per FCS_COP.1(a) by KLC Group. All update files automatically have their digital signatures verified by the TOE prior to installation using the digital signature provided in the SecurityToken file that is distributed with the downloaded update package. Updates files for EE are also obtained securely from the KLC customer web portal and contain a digital signature embedded within the binary.

6.5.5 EE:FPT_TST_EXT.1 TSF Testing

117 The TOE performs the following self-tests at start-up:

a) **Software Integrity Test.** Root of trust verification is performed through Secure Boot Mechanism for AA. When the system boots with Secure Boot enabled, it verifies the EFI binary by using a public db key that is generated during the installation process. The corresponding private db key is used to sign the EFI binaries. Further, OS Kernels are chain booted after authentication. Until chain booting occurs, EE is not invoked. EE validates the supplied key. If there is no match, the protected OS will not start.

b) **Cryptographic Module Tests.** The libgcrypt module (when initialized in FIPS mode) performs cryptographic self-tests to ensure proper functioning. The following Known Answer Tests (KAT) are conducted at power-on and do not require user intervention to execute:

i) **Symmetric Cipher Algorithm Tests**:

- GCRY_CIPHER_3DES
- GCRY_CIPHER_AES128
- GCRY_CIPHER_AES192
- GCRY_CIPHER_AES256

ii) **Hash Algorithm Tests**:

- GCRY_MD_SHA1
- GCRY_MD_SHA224
- GCRY_MD_SHA256
- GCRY_MD_SHA384
- GCRY_MD_SHA512

iii) **MAC Algorithm Tests**:

- GCRY_MAC_HMAC_SHA1
- GCRY_MAC_HMAC_SHA224
- GCRY_MAC_HMAC_SHA256
- GCRY_MAC_HMAC_SHA384
- GCRY_MAC_HMAC_SHA512
- GCRY_MAC_HMAC_SHA3_224
- GCRY_MAC_HMAC_SHA3_256
- GCRY_MAC_HMAC_SHA3_384
- GCRY_MAC_HMAC_SHA3_512
- GCRY_MAC_CMAC_3DES
- GCRY_MAC_CMAC_AES

iv) **Key Derivation Function Tests**:

- GCRY_KDF_PBKDF2

v) **DRBG Tests**:

- DRBG_NOPR_HASHSHA256
- DRBG_NOPR_HMACSHA256
- DRBG_NOPR_CTRAES128

- DRBG_NOPR_HASHSHA1
- DRBG_NOPR_HASHSHA1
- DRBG_PR_HASHSHA256
- DRBG_PR_HMACSHA256
- DRBG_PR_CTRAES128

vi) **Public Key Algorithm Tests**:

- GCRY_PK_RSA
- GCRY_PK_DSA
- GCRY_PK_ECC

118 If any of the above tests fail, the module will not initialize and no services will be rendered while in the error state.

7 Rationale

7.1 Conformance Claim Rationale

119 The following rationale is presented with regard to the PP conformance claims:

a) **TOE type.** As identified in section 2.1, the TOE is consistent with the claimed PPs.

b) **Security problem definition.** As shown in section 3, the threats, OSPs and assumptions are reproduced directly from the claimed PPs.

c) **Security objectives.** As shown in section 4, the security objectives are reproduced directly from the claimed PPs.

d) **Security requirements.** As shown in section 5, the security requirements are reproduced directly from the claimed PPs. No additional requirements have been specified.

7.2 Security Objectives Rationale

120 All security objectives are drawn directly from the claimed PPs.

7.3 Security Requirements Rationale

121 All security requirements are drawn directly from the claimed PPs. No optional SFRs are included in the ST.

122 The following selection based SFRs have been included from PP_FDE_AA:

a) FCS_CKM.1(b)

b) FCS_COP.1(a)

c) FCS_COP.1(b)

d) FCS_COP.1(c)

e) FCS_COP.1(g)

f) FCS_KDF_EXT.1

g) FCS_PCC_EXT.1

h) FCS_RBG_EXT.1

i) FCS_SMC_EXT.1

123 The following selection based SFRs have been included from PP_FDE_EE:

a) FCS_CKM.4(d)

b) FCS_COP.1(a)

c) FCS_COP.1(b)

d) FCS_COP.1(c)

e) FCS_COP.1(f)

f) FCS_COP.1(g)

g) FCS_KDF_EXT.1

h) FCS_RBG_EXT.1