Archon Linux Unified Key Setup (LUKS) v3.0.0.2 Security Target

Document Version: 2.10

Document Date: 13 May 2025

intertek acumen security

2400 Research Blvd

Suite 395

Rockville, MD 20850

CACI

2342 Richmond Hwy

Arlington, VA 22202

Revision History

Version	Date	Changes
Version 0.1	April 15, 2024	Initial draft
Version 0.5	September 5, 2024	Addressed vendor comments.
Version 0.6	September 26, 2024	Updated information based on installation experience
Version 0.7	September 27, 2004	Updated again
Version 1.3	October 15, 2024	Updated per CCTL peer review.
Version 2.0	October 31, 2024	Modified the EUDs from Dell 5430 & 3570 i7-1255U to 5430 i7-1265U and added exact dates in TOE guidance docs.
Version 2.1	November 7, 2024	Version 2.0 modified the CAVP certs to be TBD. This version updates CAVP cert table. Modified FCS_PCC passphrase length from 512 to 64. This addresses Issue 10/16 FCS_PCC Test #2.
Version 2.2	November 8, 2024	Deleted 2 platforms and updated 2 CPUs
Version 2.3	December 19, 2024	Addressed the 3 ECRs.
Version 2.4	January 20, 2025	Addressed the lab ORs per AAR
Version 2.5	February 20, 2025	Addressed more comments.
Version 2.6	March 01, 2025	Addressed comments.
Version 2.7	March 10, 2025	Addressed comments.
Version 2.8	April 22, 2025	Addressed CO ECRs
Version 2.9	May 8, 2025	Added '~' to passphrase allowed characters per ECR.
Version 2.10	May 13, 2025	Addressed an ECR for section 2.2

Table of Contents

List of Figures

List of Tables

1 Introduction

The Security Target (ST) serves as the basis for the Common Criteria (CC) evaluation and identifies the Target of Evaluation (TOE), the scope of the evaluation, and the assumptions made throughout. This document will also describe the intended operational environment of the TOE, and the functional and assurance requirements that the TOE meets.

1.1 Security Target and TOE Reference

This section provides the information needed to identify and control the TOE and the ST.

Table 1: TOE and ST Identification

Category	Identifier
ST Title	Archon Linux Unified Key Setup (LUKS) v3.0.0.2 Security Target
ST Version	2.10
ST Date	13 May 2025
ST Author	Acumen Security
TOE Identifier	Archon Linux Unified Key Setup (LUKS) v3.0.0.2
TOE Version	v3.0.0.2
TOE Developer	CACI
Key Words	Full Disk Encryption, On-Disk Encryption, Whole Disk Encryption, Full Drive Encryption, LUKS

1.2 TOE Overview

Archon Linux Unified Key Setup (LUKS) is a software Full Drive Encryption (FDE) product that is integrated into Archon Operating System (OS) (a Red Hat Enterprise Linux (RHEL) v8.10 derivative). It provides an FDE solution for the single hard drive installed in laptop models.

Archon LUKS is used to encrypt a block device. The contents of the encrypted device are arbitrary, and therefore any filesystem can be encrypted, including swap partitions. There is an unencrypted header at the beginning of an encrypted volume, which allows up to 32 encryption keys to be stored (in an encrypted form) along with encryption parameters such as cipher type and key size.

By default, the option to encrypt the block device is selected during the Archon OS installation. The system prompts users for a LUKS passphrase every time the system is booted. This passphrase unlocks the bulk encryption key that decrypts the data.

Archon LUKS provides a set of tools that simplifies managing the encrypted devices. With Archon LUKS, you can encrypt block devices and enable multiple user keys to decrypt a master key.

1.3 Usage and Major Security Features of the TOE

The TOE conforms to the *collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition*, Version 2.0 + Errata (FDE_AA cPP) and *collaborative Protection Profile for Full Drive Encryption – Encryption Engine*, Version 2.0 + Errata (FDE_EE cPP).

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

1.4 TOE Type

The TOE is a full disk encryption software application.

1.5 TOE Description

1.5.1 Evaluated Configuration

The TOE is a software TOE and has been evaluated on the following host platforms.

Table 2: Archon Linux Unified Key Setup (LUKS) v3.0.0.2 Hardware Platforms

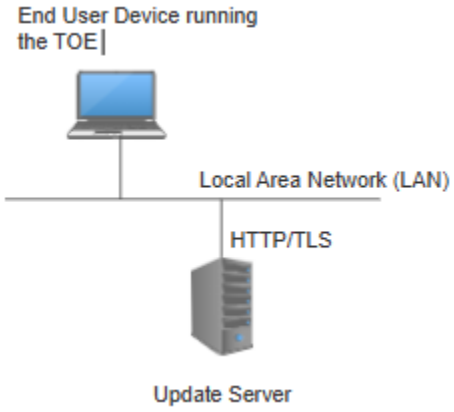
Vendor	TOE's Model	Central Processing Unit (CPU)	CPU Microarchitecture	CPU Family
Dell Inc.	Latitude 5430	Intel® Core™ i7-1265U	Golden Cove	Alder Lake
	Precision 3260	Intel® Core™ i7-12700	Golden Cove	Alder Lake
	Precision 3570	Intel® Core™ i7-1265U	Golden Cove	Alder Lake
	Precision 5570	Intel® Core™ i7-12700	Golden Cove	Alder Lake
	Latitude 5440	Intel® Core™ i5-1335U	Raptor Cove	Raptor Lake
	Latitude 5540	Intel® Core™ i5-1335U	Raptor Cove	Raptor Lake
	Precision 3580	Intel® Core™ i5-1355U	Raptor Cove	Raptor Lake
	Precision 3590	Intel® Core™ Ultra 7 155U	Ultra/Redwood Cove	Meteor Lake

1.5.2 Physical Boundary

1.5.2.1.1 TOE in the Operational Environment - Deployment

The diagram below depicts a representative TOE deployment.

Figure 1: Representative TOE Deployment



The following items are required for the operational environment.

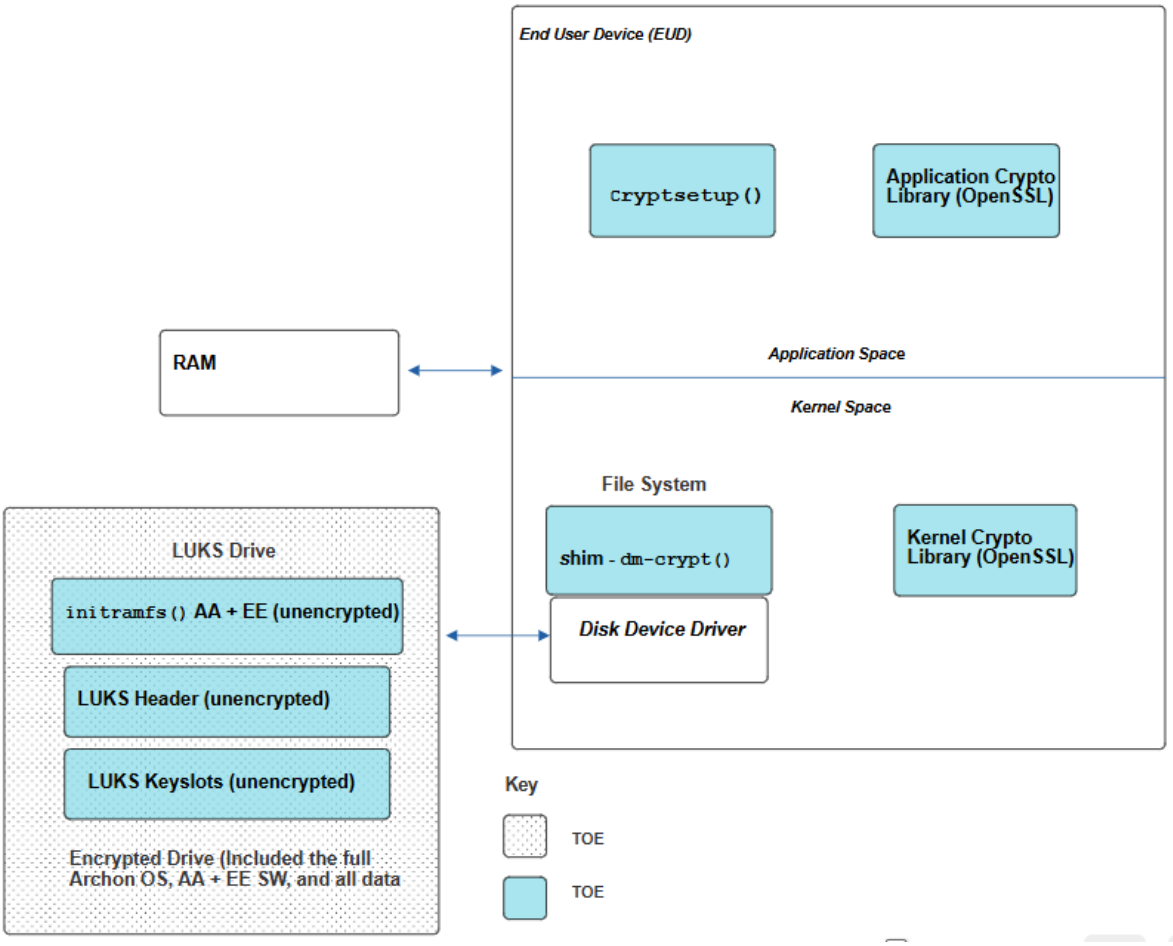
Table 3: Hardware and Software Environmental Components

Components	Mandatory/ Optional	Description
End User Device (EUD)	Mandatory	The hardware running Archon OS with LUKS enabled (the TOE). The evaluated systems are identified in Table 2 above.
Update Server	Mandatory	Provides the ability to check for TOE software updates as well as providing signed updates. The communication is performed by the Operational Environment (Archon OS).

1.5.2.2 The TOE

The following figure depicts the TOE (in blue and blue dots).

Figure 2: The TOE



The TOE is described in the following sections.

- `cryptsetup()`
  An application included in the TOE that supports a Command Line Interface (CLI) command and provides all of the AA (Authorization Acquisition) functionality, including:
  - Management interactions that include
    - Re-encrypting the drive,
    - changing the DEK,
    - erasing the DEK,
    - changing the current passphrase.

  Also provides EE (Encryption Engine) functionality, including
  - invoking the TOE DRBG (FCS_RBG_EXT.1/OSSL) to generate the salts and XTS keys,
  - performing the PBKDF2 operation for the DEK digest validation,
  - encrypting (AES-CBC) the DEK candidate when creating a new DEK,
  - and decrypting (AES-CBC) the DEK candidate.
- Application Crypto Library (OpenSSL, OpenSSL version 1.1.1k, #A5342)
- Kernel Crypto Library (OpenSSL, Linux Kernel Crypto API 4.18.0, #A5343)
- `dm-crypt()`

Provides EE functionality to transparently encrypt and decrypt (AES-XTS) data on the protected drive. It invokes OpenSSL in kernel space to perform cryptographic operations. `Dm-crypt` integrates with the kernel device mapper subsystem to provide a low-level mapping to provide the cryptographic protection of the data on the protected drive.

- `initramfs()`
  `initramfs` is a scaled-down Archon OS image that is invoked during the boot of the TOE platform. It is extracted into the root file system when the kernel boots. `initramfs` is part of the TOE, and AA functionality within `initramfs`
  - collects the passphrase from the user,
  - derives the KEK and
  - supplies it to EE functionality.

  The EE functionality
  - decrypts the DEK candidate and
  - validates it.

  If successful, the DEK is then used to boot the complete Archon OS system from the protected drive. If more than three (not configurable) number of consecutive failures occur, the TOE automatically reboots to limit the rate of attempts.
- LUKS Header (unencrypted)
  - Header
    Contains metadata about the drive encryption including information about the encryption algorithm used, and includes the cipher name (AES), cipher mode (XTS), salt, and hash (SHA-512).
  - Keyslots (unencrypted)
    Holds the AES-CBC encrypted DEK.
- Hard Drive
  The complete contents stored on the hard drive is in the TOE boundary. It includes the full Archon OS, `initramfs` , and all data. Everything is encrypted except for `initramfs` and the LUKS Header.

A full description of the TOE's key management is described in a proprietary ancillary *Key Management Description,* made available at the discretion of the developer.

1.5.3 TOE Documentation (Operational Guidance)

The following documents are essential to understanding and controlling the TOE in the evaluated configuration and included in the TOE physical boundary.

- [AGD_OS] *CACI Archon OS v3.0.0.2 Common Criteria User Guidance*, Version 1.2, July 2024 (Can be obtained from the NIAP PCL (www.niap-ccevs.org) posting VID11429).
- [AGD_LUKS] *CACI Archon Linux Unified Key Setup (LUKS) v3.0.0.2 Common Criteria User Guidance,* Version 2.7, May 2025.

1.6 Security Functions Provided by the TOE

The TOE provides the security functions required by the FDE_AA cPP and FDE_EE cPP.

1.6.1 Cryptographic Support

The TOE includes NIST CAVP-validated cryptographic algorithms supporting cryptographic functions. The TOE provides Key Derivation, BEV (Border Encrypt Value) Validation, and data encryption.

1.6.2 User Data Protection

The TOE performs Full Drive Encryption such that the drive contains no plaintext user data. The TOE performs user data encryption by default in the out-of-the-box configuration using XTS-AES-256 mode.

1.6.3 Security Management

The TOE supports management functions for changing and erasing the DEK (Data Encryption Key), modifying the passphrase, and initiating the TOE updates using a command line interface.

1.6.4 Protection of the TSF

The TOE provides trusted firmware updates, protects Key and Key Material; and supports power saving states. The TOE runs a suite of self-tests during initial start-up (on power on).

1.7 Product Functionality not Included in the Scope of the Evaluation

The following product functionality is not included in the CC evaluation:

- The TOE only supports a CLI interface; no GUI (Graphical User Interface) or program, or script.

2 Conformance Claims

This section identifies the TOE conformance claims, conformance rationale, and relevant Technical Decisions (TDs).

2.1 CC Conformance Claims

The TOE is conformant to the following:

- *Common Criteria for Information Technology Security Evaluation Part 2*, Version 3.1, Revision 5, April 2017 extended.
- *Common Criteria for Information Technology Security Evaluation Part 3*, Version 3.1, Revision 5, April 2017 conformant.

2.2 Protection Profile Conformance

This ST and the TOE it describes are exact conformance to the following CC specifications:

- *PP-Configuration for Full Drive Encryption - Authorization Acquisition and Full Drive Encryption – Encryption Engine*, Version 1.0, 31 May 2024.

  This PP-Configuration includes the following:

  - *collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition*, Version 2.0 + Errata 20190201, February 1, 2019. (FDE_AA).
  - *collaborative Protection Profile for Full Drive Encryption – Encryption Engine*, Version 2.0 + Errata 20190201, February 1, 2019 (FDE_EE).

2.3 Conformance Rationale

This ST provides exact conformance to FDE_AA cPP and FDE_EE cPP. The security problem definition, security objectives, and security requirements in this ST are all taken from the collaborative Protection Profiles (cPPs), performing only the operations defined there.

2.3.1 Technical Decisions (TDs)

All NIAP TDs issued to date and applicable to FDE AA v2.0 and FDE EE v2.0 have been considered. The following table identifies all applicable TDs. Note, the SFRs included in the Notes column follow the conventions identified in Section 5.1.2.

Note, if a TD pertains to an SFR claimed by the ST, it is considered applicable to the evaluation (marked as Yes in third column of the table below) even if the modification of the SFR does not directly affect the ST's SFR claim. Specifically, e.g. if the modification effects a selection the TOE does not select, the TD still is applicable in this ST. For each SFR claimed, there is an application note following the SFR listing if a TD is applicable.

Table 4: Relevant Technical Decisions

Relevant Technical Decisions	cPP		Applicable to the Evaluation (Yes/No)	Notes and/or Exclusion Rationale (if applicable)
	FDE_AA	FDE_EE		
TD0901: FIT Technical Decision: Clarification to FCS_PCC_EXT.1.1	✓		Yes	Applies to FCS_PCC_EXT.1.1
TD0769: FIT Technical Decision for FPT_KYP_EXT.1.1	✓	✓	Yes	Applies to FPT_KYP_EXT.1/AA and FPT_KYP_EXT.1/EE SFRs.
TD0767: FIT Technical Decision for FMT_SMF.1.1	✓		Yes	Applies to SFR and Test for FMT_SMF.1.1/AA.
TD0766: FIT Technical Decision for FCS_CKM.4(d) Test Notes	✓	✓	Yes	Applies to Test only for FCS_CKM.4(d) (FDE_AA and FDE_EE).
TD0765: FIT Technical Decision for FMT_MOF.1	✓		Yes	Applies to Test only for FMT_MOF.1/AA.
TD0764: FIT Technical Decision for FCS_PCC_EXT.1	✓		Yes	Applies to FCS_PCC_EXT.1/AA SFR.
TD0760: FIT Technical Decision for FCS_SNI_EXT.1.3, FCS_COP.1(f).	✓		Yes	Applies to the SFR and TSS for FCS_SNI_EXT.1.3/AA and SFR for FCS_COP.1(f)/AA (not claimed).
TD0759: FIT Technical Decision for FCS_AFA_EXT.1.1	✓		Yes	Applies to FCS_AFA_EXT.1/AA SFR.
TD0606: FIT Technical Recommendation for Evaluating a NAS against the FDE AA and FDEE	✓	✓	Yes	Does not affect any TOE evidence.
TD0464: FIT Technical Decision for FPT PWR EXT.1 compliant power savings states		✓	Yes	Applies to FPT_PWR_EXT.1/EE SFR.
TD0460: FIT Technical Decision for FPT PWR EXT.1 non-compliant power saving states		✓	Yes	Applies to AGD for FPT_PWR_EXT.1/EE.
TD0458: FIT Technical Decision for FPT_KYP_EXT.1 evaluation activities	✓	✓	Yes	Applies to TSS, AGD, and KMD for FPT_KYP_EXT.1/AA and FPT_KYP_EXT.1/EE.

3 Security Problem Definition

The security problem definition has been taken directly from the claimed cPPs specified in Section 2.2 and is reproduced here for the convenience of the reader. The security problem is described in terms of the threats that the TOE is expected to address, assumptions about the operational environment, and any Organizational Security Policies (OSPs) that the TOE is expected to enforce.

3.1 Threats

The threats included in Table 5 are drawn directly from the cPPs. The threats listed below that apply only to the FDE_AA or FDE_EE cPPs are appended with /AA or /EE respectively.

Table 5: Threats

ID	Threat
T.AUTHORIZATION_GUESSING/AA	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.AUTHORIZATION_GUESSING/EE	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.CHOSEN_PLAINTEXT/EE	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.KEYING_MATERIAL_COMPROMISE/AA	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.KEYING_MATERIAL_COMPROMISE/EE	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

ID	Threat
	peripherals in the operating environment (OE), e.g. BIOS configuration, SPI flash, or TPMs.
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/EE	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.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.UNAUTHORIZED_FIRMWARE_MODIFY/EE	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.
T.UNAUTHORIZED_FIRMWARE_UPDATE/EE	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_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.

3.2 Assumptions

The assumptions included in the following table are drawn directly from the cPPs. The assumptions listed below that apply only to the FDE_AA or FDE_EE cPPs are appended with /AA or /EE respectively.

Table 6: Assumptions

ID	Assumption
A.INITIAL_DRIVE_STATE/AA	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.INITIAL_DRIVE_STATE/EE	Users enable Full Drive Encryption on a newly provisioned or initialized storage device free of protected data in areas not targeted for encryption. It is also assumed that data intended for protection should not be on the targeted storage media until after provisioning. The cPP does not intend to include requirements to find all the areas on storage devices that potentially contain protected data. In some cases, it may not be possible - for example, data contained in “bad” sectors. While inadvertent exposure to data contained in bad sectors or un-partitioned space is unlikely, one may use forensics tools to recover data from such areas of the storage device. Consequently, the cPP assumes bad sectors, un-partitioned space, and areas that must contain unencrypted code (e.g., MBR and AA/EE pre-authentication software) contain no protected data.
A.PASSWORD_STRENGTH/AA	Authorized administrators ensure password/passphrase authorization factors have sufficient strength and entropy to reflect the sensitivity of the data being protected.
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.
A.PLATFORM_I&A/AA	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.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.

ID	Assumption
A.POWER_DOWN/AA	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”.
A.POWER_DOWN/EE	The user does not leave the platform and/or storage device unattended until all volatile memory is cleared after a power-off. This properly clears memories and locks down the device. 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.SECURE_STATE/AA	Upon the completion of proper provisioning, the drive is only assumed secure when in a powered off state up until it is powered on and receives initial authorization.
A.SINGLE_USE_ET/AA	External tokens that contain authorization factors are used for no other purpose than to store the external token authorization factors.
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.TRAINED_USER/AA	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.TRAINED_USER/EE	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.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.

3.3 Organizational Security Policies

There are no OSPs associated with the TOE.

4 Security Objectives

The security objectives have been taken directly from the FDE_AA and FDE_EE cPPs and are reproduced here for the convenience of the reader.

4.1 Security Objectives for the TOE

The cPPs do not define security objectives for the TOE.

4.2 Security Objectives for the Operational Environment

Security objectives for the operational environment assist the TOE in correctly providing its security functionality. These objectives, which are found in the table below, track with the assumptions about the TOE operational environment. The objectives for the operational environment listed below that apply only to the FDE_AA or FDE_EE cPPs are appended with /AA or /EE respectively.

Table 7: Security Objectives for the Operational Environment

ID	Security Objectives
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.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.
OE.PLATFORM_I&A/AA	The Operational Environment will provide individual user identification and authentication mechanisms that operate independently of the authorization factors used by the TOE.
OE.PLATFORM_STATE/AA	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.POWER_DOWN/AA	Volatile memory is cleared after power-off so memory remnant attacks are infeasible. .
OE.POWER_DOWN/EE	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.

ID	Security Objectives
OE.TRAINED_USERS	Authorized users will be properly trained and follow all guidance for securing the TOE and authorization factors.
OE.TRUSTED_CHANNEL	Communication among and between product components (i.e., AA and EE) is sufficiently protected to prevent information disclosure.

5 Security Requirements

This section identifies the Security Functional Requirements (SFRs) and the Security Assurance Requirements (SARs) for the TOE. The Security Requirements included in this section are derived from Part 2 of the *Common Criteria for Information Technology Security Evaluation*, Version 3.1, Revision 5; the *collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition*, Version 2.0 + Errata 20190201, and the *collaborative Protection Profile for Full Drive Encryption – Encryption Engine*, Version 2.0 + Errata 20190201.

5.1 Functional Requirements

5.1.1 Extended Functional Requirements

All the extended requirements in this ST have been drawn from FDE_AA or FDE_EE cPPs and are itemized below. The cPPs define the extended SFRs. Since they have not been redefined in this ST, the FDE_AA or FDE_EE cPP should be consulted for more information regarding these extensions to CC Part 2.

From FDE_AA cPP:

- FCS_AFA_EXT.1/AA
- FCS_AFA_EXT.2/AA
- FCS_CKM_EXT.4(a)
- FCS_CKM_EXT.4(b)/AA
- FCS_KDF_EXT.1/AA
- FCS_KYC_EXT.1/AA
- FCS_PCC_EXT.1/AA
- FCS_RBG_EXT.1/OSSL
- FCS_SNI_EXT.1/AA
- FPT_KYP_EXT.1/AA
- FPT_PWR_EXT.1/AA
- FPT_PWR_EXT.2/AA
- FPT_TST_EXT.1
- FPT_TUD_EXT.1

From FDE_EE cPP:

- FCS_CKM_EXT.4(a)
- FCS_CKM_EXT.4(b)/EE
- FCS_CKM_EXT.6/EE
- FCS_KYC_EXT.2/EE
- FCS_RBG_EXT.1/OSSL
- FCS_RBG_EXT.1/KERN
- FCS_SNI_EXT.1/EE
- FCS_VAL_EXT.1/EE
- FDP_DSK_EXT.1/EE
- FPT_KYP_EXT.1/EE
- FPT_PWR_EXT.1/EE
- FPT_PWR_EXT.2/EE
- FPT_TST_EXT.1

- FPT_TUD_EXT.1

5.1.2 Conventions

The CC allows the following types of operations to be performed on the functional requirements: assignments, selections, refinements, and iterations. The following font conventions are used within this document to identify operations defined by CC:

- Assignment: Indicated with *italicized* text;
- Refinement: Indicated with **bold** text;
- Selection: Indicated with underlined text;
- Iteration: The cPP authors indicate iterations within the cPPs by placing parenthesis placed around a lowercase letter following the SFR, e.g., FCS_COP.1(f). The ST author indicates iterations required for the identification of source FDE_AA and FDE_EE cPP SFRs by placing a forward slash followed by either AA or EE after the SFR, e.g., FCS_CKM.4(a)/AA indicating that this SFR is from FDE_AA. The /AA or /EE iteration is applied:
  - if the SFR is included in both cPPs but the SFR text or operations are not identical;
  - if the SFR is included in both cPPs but the TSS, Operational Guidance, KMD, or Test assurance activities are not identical;
  - if the SFR is included in both cPPs and a TD applies to only one cPP;
  - if the SFR includes a reference or selection to another SFR that has been iterated by the ST author;
  - the SFR is only included in one cPP.

  Otherwise, the SFR is listed without a following /AA or /EE indicating the SFR applies to both cPPs and are identical.
- An exception to the above iteration rule is FCS_RBG_EXT.1. This SFR is iterated to identify the cryptographic library (/OSSL and /KERN) used. FCS_RBG_EXT.1/OSSL applies to both FDE_AA and FDE_EE cPPs. FCS_RBG_EXT.1/KERN applies to FDE_EE cPP.
- The font formatting used in the cPPs or TD is not retained. Italicized text, bold text, and strikeout text are replaced with plain text.

5.1.3 Security Functional Requirements (SFRs)

The TOE functional requirements for this ST are taken directly from the cPPs which are derived from *Common Criteria for Information Technology Security Evaluation Part 2*, Version 3.1, Revision 5.

Table 8: SFRs

Requirement Class	Requirement	Description
FCS: Cryptographic support	FCS_AFA_EXT.1/AA	Authorization Factor Acquisition *(FDE_AA)*
	FCS_AFA_EXT.2/AA	Timing of Authorization Factor Acquisition *(FDE_AA)*
	FCS_CKM.1(c)/EE	Cryptographic Key Generation (Data Encryption Key) *(FDE_EE)*

Requirement Class	Requirement	Description
	FCS_CKM.4(a)/AA	Cryptographic Key Destruction (Power Management) *(FDE_AA)*
	FCS_CKM.4(a)/EE	Cryptographic Key Destruction (Power Management) *(FDE_EE)*
	FCS_CKM.4(d)	Cryptographic Key Destruction (Software TOE, 3rd Party Storage) *(FDE_AA) (FDE_EE)*
	FCS_CKM_EXT.4(a)	Cryptographic Key and Key Material Destruction (Destruction Timing) *(FDE_AA) (FDE_EE)*
	FCS_CKM_EXT.4(b)/AA	Cryptographic Key and Key Material Destruction (Power Management) *(FDE_AA)*
	FCS_CKM_EXT.4(b)/EE	Cryptographic Key and Key Material Destruction (Power Management) *(FDE_EE)*
	FCS_CKM_EXT.6/EE	Cryptographic Key Destruction Types *(FDE_EE)*
	FCS_COP.1(a)	Cryptographic Operation (Signature Verification) *(FDE_AA) (FDE_EE)*
	FCS_COP.1(b)	Cryptographic Operation (Hash Algorithm) *(FDE_AA) (FDE_EE)*
	FCS_COP.1(c)/AA	Cryptographic Operation (Keyed Hash Algorithm) *(FDE_AA)*
	FCS_COP.1(c)/EE	Cryptographic Operation (Message Authentication) *(FDE_EE)*
	FCS_COP.1(f)/EE	Cryptographic Operation (AES Data Encryption/Decryption) *(FDE_EE)*
	FCS_COP.1(g)/EE	Cryptographic Operation (Key Encryption) *(FDE_EE)*
	FCS_KDF_EXT.1/AA	Cryptographic Key Derivation *(FDE_AA)*
	FCS_KYC_EXT.1/AA	Key Chaining (Initiator) *(FDE_AA)*
	FCS_KYC_EXT.2/EE	Key Chaining (Recipient) *(FDE_EE)*
	FCS_PCC_EXT.1/AA	Cryptographic Password Construct and Conditioning *(FDE_AA)*
	FCS_RBG_EXT.1/OSSL	Cryptographic Operation (Random Bit Generation) *(FDE_AA) (FDE_EE)*

Requirement Class	Requirement	Description
	FCS_RBG_EXT.1/KERN	Cryptographic Operation (Random Bit Generation) *(FDE_EE)*
	FCS_SNI_EXT.1/AA	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_AA)*
	FCS_SNI_EXT.1/EE	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_EE)*
	FCS_VAL_EXT.1/EE	Validation *(FDE_EE)*
FDP: User data protection	FDP_DSK_EXT.1/EE	Protection of Data on Disk *(FDE_EE)*
FMT: Security management	FMT_MOF.1/AA	Management of Functions Behavior *(FDE_AA)*
	FMT_SMF.1/AA	Specification of Management Functions *(FDE_AA)*
	FMT_SMF.1/EE	Specification Management Functions *(FDE_EE)*
	FMT_SMR.1/AA	Security Roles *(FDE_AA)*
FPT: Protection of the TSF	FPT_KYP_EXT.1/AA	Protection of Key and Key Material *(FDE_AA)*
	FPT_KYP_EXT.1/EE	Protection of Key and Key Material *(FDE_EE)*
	FPT_PWR_EXT.1/AA	Power Savings States *(FDE_AA)*
	FPT_PWR_EXT.1/EE	Power Savings States *(FDE_EE)*
	FPT_PWR_EXT.2/AA	Timing of Power Savings States *(FDE_AA)*
	FPT_PWR_EXT.2/EE	Timing of Power Savings States *(FDE_EE)*
	FPT_TST_EXT.1	TSF Testing *(FDE_AA) (FDE_EE)*
	FPT_TUD_EXT.1	Trusted Update *(FDE_EE)*

5.1.3.1 Cryptographic Support (FCS)

5.1.3.1.1 FCS_AFA_EXT.1/AA Authorization Factor Acquisition *(FDE_AA)*

FCS_AFA_EXT.1.1/AA The TSF shall accept the following authorization factors: [

- a submask derived from a password authorization factor conditioned as defined in FCS_PCC_EXT.1**/AA**,

].

Application Note: Addressed TD0759 which did not result in a modification to the SFR.

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.2 FCS_AFA_EXT.2/AA Timing of Authorization Factor Acquisition *(FDE_AA)*

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.3 FCS_CKM.1(c)/EE Cryptographic Key Generation (Data Encryption Key) *(FDE_EE)*

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

- generate a DEK using the RBG as specified in FCS_RBG_EXT.1**/OSSL**,

] and specified cryptographic key sizes [256 bits].

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.4 FCS_CKM.4(a)/AA Cryptographic Key Destruction (Power Management) *(FDE_AA)*

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.5 FCS_CKM.4(a)/EE Cryptographic Key Destruction (Power Management) *(FDE_EE)*

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.6 FCS_CKM.4(d) Cryptographic Key Destruction (Software TOE, 3rd Party Storage) *(FDE_AA) (FDE_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 [
  - removal of power to the memory,

];

] that meets the following: [no standard].

Application Note: This SFR is mandatory for FDE_AA (and also a selection (FCS_CKM.4.1(a)/AA)) and a selection based SFR (FCS_CKM_EXT.6.1) for FDE_EE.

5.1.3.1.7 FCS_CKM_EXT.4(a) Cryptographic Key and Key Material Destruction (Destruction Timing) *(FDE_AA) (FDE_EE)*

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

5.1.3.1.1 FCS_CKM_EXT.4(b)/AA Cryptographic Key and Key Material Destruction (Power Management) *(FDE_AA)*

FCS_CKM_EXT.4.1(b)/AA 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 FPT_PWR_EXT.1**/AA**.

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.1 FCS_CKM_EXT.4(b)/EE Cryptographic Key and Key Material Destruction (Power Management) *(FDE_EE)*

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.2 FCS_CKM_EXT.6/EE Cryptographic Key Destruction Types *(FDE_EE)*

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

5.1.3.1.3 FCS_COP.1(a) Cryptographic Operation (Signature Verification) *(FDE_AA) (FDE_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 [4096-bit];
- Elliptic Curve Digital Signature Algorithm with a key size of 256 bits or greater

] 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 9796-2, Digital signature scheme 2 or Digital Signature scheme 3, for RSA schemes
- FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" [P-384]; ISO/IEC 14888-3, Section 6.4, for ECDSA schemes

].

5.1.3.1.4 FCS_COP.1(b) Cryptographic Operation (Hash Algorithm) *(FDE_AA) (FDE_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, SHA-512] that meet the following: [ISO/IEC 10118-3:2004].

5.1.3.1.5 FCS_COP.1(c)/AA Cryptographic Operation (Hash Algorithm) *(FDE_AA)*

FCS_COP.1.1(c)/AA Refinement: The TSF shall perform cryptographic [keyed-hash message

authentication] in accordance with a specified cryptographic algorithm [HMAC-SHA-256, HMAC_SHA-512] and cryptographic key sizes [*256, 512 bits*] that meet the following: [ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"].

5.1.3.1.6 FCS_COP.1(c)/EE Cryptographic Operation (Message Authentication) *(FDE_EE)*

FCS_COP.1.1(c)/EE Refinement: The TSF shall perform cryptographic [message authentication] in accordance with a specified cryptographic algorithm [HMAC-SHA-256, HMAC-SHA-512] and cryptographic key sizes [*256 bits, 512 bits*] that meet the following: [ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"].

5.1.3.1.7 FCS_COP.1(f)/EE Cryptographic Operation (AES Data Encryption/Decryption) *(FDE_EE)*

FCS_COP.1.1(f)/EE 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 [256 bits] that meet the following: [

- AES as specified in ISO/IEC 18033-3,
- [XTS as specified in IEEE 1619]

].

5.1.3.1.8 FCS_COP.1(g)/EE Cryptographic Operation (Key Encryption) *(FDE_EE)*

FCS_COP.1.1(g)/EE 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 [256 bits] that meet the following: [

- AES as specified in ISO/IEC 18033-3,
- [CBC as specified in ISO/IEC 10116]

].

5.1.3.1.9 FCS_ KDF_EXT.1/AA Cryptographic Key Derivation *(FDE_AA)*

FCS_KDF_EXT.1.1/AA 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 FCS_COP.1(c)**/AA**, such that the output is at least of equivalent security strength (in number of bits) to the BEV.

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2

5.1.3.1.10 FCS_KYC_EXT.1/AA Key Chaining (Initiator) *(FDE_AA)*

FCS_KYC_EXT.1.1/AA 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 FCS_KDF_EXT.1**/AA**,

]

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

FCS_KYC_EXT.1.2/AA The TSF shall provide at least a [256 bit] BEV to [*EE*] [

- without validation taking place

].

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.11 FCS_KYC_EXT.2/EE Key Chaining (Recipient) *(FDE_EE)*

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

FCS_KYC_EXT.2.2/EE The TSF shall maintain a chain of intermediary keys originating from the BEV to the DEK using the following method(s): [

- key encryption as specified in FCS_COP.1(g)**/EE**

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

FCS_PCC_EXT.1/AA Cryptographic Password Construct and Conditioning *(FDE_AA)*

FCS_PCC_EXT.1.1/AA A password used by the TSF to generate a password authorization factor shall enable at least [*64*] characters in the set of {upper case characters, lower case characters, numbers, and [*the following special characters: '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '<', '>', ',', '.', '/', '?', '|', '\', ':', ';', '"', '', '-', '+', '~'*]} and shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC-[SHA-512], with [*10000*] iterations, and output cryptographic key sizes [256 bits] that meet the following: [*NIST SP 800-132*].

Application Note: Addressed TD0764.

Application Note: Applied TD0901.

Application Note: This password authorization factor refers to the passphrase given by the LUKS User and not the password given by the Archon User (OE).

5.1.3.1.12 FCS_RBG_EXT.1/OSSL Random Bit Generation *(FDE_AA) (FDE_EE)*

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

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

- [*1*] software-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.

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

Application Note: This OpenSSL DRBG is invoked by both AA and EE. AA is entirely instantiated within application space (primarily cryptsetup). EE is split between application space (primarily cryptsetup) and the kernel (dm-crypt).

5.1.3.1.13 FCS_RBG_EXT.1/KERN Random Bit Generation *(FDE_EE)*

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

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

- [*1*] software-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.

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.14 FCS_SNI_EXT.1/AA Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_AA)*

FCS_SNI_EXT.1.1/AA The TSF shall [

- use salts that are generated by [
  - DRBG as specified in FCS_RBG_EXT.1**/OSSL**

  ]

].

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

FCS_SNI_EXT.1.3/AA The TSF shall [use no IVs].

Application Note: Updated FCS_SNI_EXT.1.3 as per TD0760.

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.1.15 FCS_SNI_EXT.1/EE Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_EE)*

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

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

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

- CBC: IVs shall be non-repeating and unpredictable;
- XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer;

].

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

Application Note: Per FCS_SNI_EXT.1 Application Note, "This SFR does not prescribe when salts, nonces, and IVs must be used, only that when they are used, they must be generated in a certain manner.".

*Application Note: CBC IVs are generated using FCS_RBG_EXT.1/**OSSL** and XTS initial Tweak value is generated using FCS_RBG_EXT.1**/KERN**.*

5.1.3.1.16 FCS_VAL_EXT.1/EE Validation *(FDE_EE)*

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

- hash the [BEV] as specified in [FCS_COP.1(c)**/EE**] and compare it to a stored hashed [value];

].

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

FCS_VAL_EXT.1.3/EE The TSF shall [

- require power cycle/reset the TOE after [*3*] of consecutive failed validation attempts

].

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.2 User Data Protection (FDP)

5.1.3.2.1 FDP_ DSK_EXT.1/EE Protection of Data on Disk *(FDE_EE)*

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

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.3 Security Management (FMT)

5.1.3.3.1 FMT_MOF.1/AA Management of Functions Behavior *(FDE_AA)*

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

Application Note: The Archon OS system includes two administrators: LUKS User and Archon User. This SFR for this evaluation refers to the TOE User.

5.1.3.3.2 FMT_SMF.1/AA Specification of Management Functions *(FDE_AA)*

FMT_SMF.1.1/AA 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) [no other functions]

].

Application Note: Addressed TD0767.

5.1.3.3.3 FMT_SMF.1/EE Specification of Management Functions *(FDE_EE)*

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

a) change the DEK, as specified in FCS_CKM.1, when re-provisioning or when commanded,
b) erase the DEK, as specified in FCS_CKM.4(a)**/EE**,
c) initiate TOE firmware/software updates,
d) [*[change the default authorization factor]]*

].

Application Note: The FCS_CKM.1 reference is interpreted to refer to all FCS_CKM.1 iterations: FCS_CKM.1(a) Cryptographic Operation (Asymmetric Keys), FCS_CKM.1(b) Cryptographic Operation (Symmetric Keys), and FCS_CKM.1(c). This ST interprets FCS_CKM.1 as FCS_CKM.1(c)/EE Cryptographic Key Generation.

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.3.4 FMT_SMR.1/AA Security Roles *(FDE_AA)*

FMT_SMR.1.1/AA The TSF shall maintain the roles [authorized user].

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

5.1.3.4 Protection of the TSF (FPT)

5.1.3.4.1 FPT_KYP_EXT.1/AA Protection of Key and Key Material *(FDE_AA)*

FPT_KYP_EXT.1.1/AA The TSF shall [

- not store keys in non-volatile memory

].

Application Note: Addressed TD0769 which did not result in a modification to the SFR.

5.1.3.4.2 FPT_KYP_EXT.1/EE Protection of Key and Key Material *(FDE_EE)*

FPT_KYP_EXT.1.1/EE The TSF shall [

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

].

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

Application Note: Addressed TD0769 which did not result in a modification to the SFR.

Application Note: The ST does not claim FCS_COP.1(d) Cryptographic Operation (Key Wrapping) or FCS_COP.1(e) Cryptographic Operation (Key Transport). This ST claims stored keys are protected by encryption as specified in FCS_COP.1(g)/EE Cryptographic Operation (Key Encryption).

5.1.3.4.3 FPT_PWR_EXT.1/AA Power Saving States *(FDE_AA)*

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

5.1.3.4.1 FPT_PWR_EXT.1/EE Power Saving States *(FDE_EE)*

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

Application Note: Addressed TD0464.

5.1.3.4.2 FPT_PWR_EXT.2/AA Timing of Power Saving States *(FDE_AA)*

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.4.3 FPT_PWR_EXT.2/EE Timing of Power Saving States *(FDE_EE)*

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

Refinement Rationale: The SFR was modified to reflect the SFR conventions followed by the ST as described in section 5.1.2.

5.1.3.4.4 FPT_TST_EXT.1/EE TSF Testing *(FDE_AA) (FDE_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: [*FIPS Power-On Self Tests*].

Application Note: This SFR is mandatory for FDE_EE and optional for FDE_AA. The ST only claims the mandatory SFR required by FDE_EE.

5.1.3.4.5 FPT_TUD_EXT.1 Trusted Update *(FDE_AA) (FDE_EE)*

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

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

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

5.1.4 TOE SFR Dependencies Rationale for SFRs

The cPPs contain all the requirements claimed in this ST. As such, dependencies are not applicable since the cPPs have been approved.

5.2 Assurance Requirements

5.2.1 Extended Assurance Requirements

There are no extended assurance requirements defined in the FDE_AA or the FDE_EE cPPs and therefore, none are included in this ST.

5.2.2 Security Assurance Requirements (SARs)

The TOE assurance requirements for this ST are taken directly from the cPPs which are derived from *Common Criteria for Information Technology Security Evaluation Part 3*, Version 3.1, Revision 5.

Table 9: Security Assurance Requirements

Functional Class	Functional Components	Component Description
Security Target	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 Functionality 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

5.2.3 Assurance Measures

The TOE satisfied the identified assurance requirements. This section identifies the Assurance Measures applied by CACI to satisfy the assurance requirements. The following table lists the details.

Table 10: TOE Security Assurance Measures

SAR Component	How the SAR will be met
ASE_TSS.1	Refinement: The TOE summary describes how the TOE meets each SFR. It also includes a reference to the proprietary Key Management Description (Appendix E) and [Entropy Essay].

SAR Component	How the SAR will be met
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 may interfaces to the Operational Environment that are not directly invoked by TOE users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional "functional specification" documentation is necessary to satisfy the Evaluation Activities specified in the SD. The Evaluation Activities in the SD are associated with the applicable SFRs; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.
AGD_OPE.1	The operational user guidance does not have to be contained in a single document. Guidance to users, administrators, and integrators can be spread among documents or web pages. The developer should review the Evaluation Activities contained in the SD to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.
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.
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. The evaluator performs the CEM work units associated with ALC_CMC.1
ALC_CMS.1	Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.
ATE_IND.1	CACI will provide the TOE for testing.
AVA_VAN.1	CACI will provide the TOE for testing. CACI will provide a document identifying the list of software and hardware components.

5.2.4 Rationale for Security Assurance Requirements

The functional specification describes the external interfaces of the TOE, such as the means for a user to invoke a service and the corresponding response of those services. The description includes the interface(s) that enforces a security functional requirement, the interface(s) that supports the enforcement of a security functional requirement, and the interface(s) that does not enforce any security functional requirements. The interfaces are described in terms of their purpose (general goal of the interface), method of use (how the interface is to be used), parameters (explicit inputs to and outputs from an interface that control the behavior of that interface), parameter descriptions (tells what the parameter is in some meaningful way), and error messages (identifies the condition that generated it,

what the message is, and the meaning of any error codes). The development evidence also contains a tracing of the interfaces to the SFRs described in this ST.

6 TOE Summary Specification

6.1 Cryptographic Support (FCS)

Note: a full description of the TOE's key management is described in a proprietary ancillary *Key Management Description,* made available at the discretion of the developer.

6.1.1 FCS_AFA_EXT.1/AA Authorization Factor Acquisition *(FDE_AA)*

The TOE supports passphrase authorization factor. Passphrases are created when a drive is encrypted. Users are prompted for a passphrase that is entered via the keyboard at system boot. Additionally, Users may change a passphrase at any time via the "cryptsetup luksChangeKey" command.

Administrators are instructed in the user guidance to configure the minimum passphrase to 12 when they receive their system. Archon OS does not support a CLI command or configuration parameter that enables an Administrator to set a maximum passphrase length. However, the CC evaluated configuration requires the system support at least 64 characters of passphrase length.

The passphrase can be composed of ASCII upper case and lower case letters, numbers, and the following special characters: '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '<', '>', ',', '.', '/', '?', '|', '\', ':', ';', '"', '', '-', '+', '~']. FCS_PCC_EXT.1/AA describes the specific requirements of a LUKS passphrase.

6.1.2 FCS_AFA_EXT.2/AA Timing of Authorization Factor Acquisition *(FDE_AA)*

The TOE does not have any power-saving states beyond power-on and power-off. After transitioning from the power-off to the power-on state, the LUKS User must authenticate to the TOE, using the passphrase authorization factor. Once the boot process completes, the LUKS User must authenticate to Archon OS (Operational Environment) before the TOE will allow data to be read from or written to the drive.

6.1.3 FCS_CKM.1(c)/EE Cryptographic Key Generation (Data Encryption Key) *(FDE_EE)*

The CACI Archon OS installer enables Archon LUKS by issuing the `cryptsetup encrypt` CLI command. The TOE generates 256-bit DEKs (Data Encryption Key) using AES-256 CTR_DRBG and stores the DEK, encrypted by the KEK (Key Encryption Key), in the dedicated LUKS header which is plaintext at the beginning of the LUKS-encrypted drive.

Additionally, users may change the DEK at anytime by issuing the `cryptsetup reencrypt` CLI command. Once issued, the TOE generates a new DEK and stores the DEK, encrypted by the KEK, in the LUKS header.

The KEK is derived from the passphrase. Each time a DEK is created, the user is prompted for the passphrase and the KEK is derived.

Note that AES-XTS uses two independent 256-bit symmetric keys. References to the DEK in this document refer to both of the keys. The two keys are stored as a single encrypted 512-bit entity in the LUKS header.

6.1.4 FCS_CKM.4(a)/AA Cryptographic Key Destruction (Power Management) *(FDE_AA)* and FCS_CKM.4(a)/EE Cryptographic Key Destruction (Power Management) *(FDE_EE)*

When the TOE powers off (Operational Environment), all values in memory drain to a zero state. Refer to Table 13 for more information.

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

The TOE's RAM (Random Access Memory) serves as the working memory in which the TOE temporarily stores working copies of key material. The TOE clears keys from memory by removal of power to the system.

There are no circumstances that do not conform to the key destruction requirement.

Refer to Table 13: Key Zeroization for more detail.

6.1.6 FCS_CKM_EXT.4(a)/ Cryptographic Key and Key Material Destruction (Destruction Timing) *(FDE_AA) (FDE_EE)*

The TOE clears the KEK (BEV (Border Encrypt Value)) and passphrase from application space memory after access is granted to the LUKS partition by successfully authenticating with a passphrase. EE continues to execute after access is granted and control is passed to the OS, and the DEK stays in kernel memory until the system is shut down.

There are no circumstances that do not conform to the key destruction requirement.

6.1.7 FCS_CKM_EXT.4(b)/AA Cryptographic Key and Key Material Destruction (Power Management) *(FDE_AA)* and FCS_CKM_EXT.4(b)/EE Cryptographic Key and Key Material Destruction (Power Management) *(FDE_EE)*

The TOE has no Compliant power saving states other than shutdown (G2(S5) and G3).

All plaintext key material, BEV, and authentication factors stored in volatile memory are cleared per FCS_CKM.4(d) when transitioning to a shutdown state.

6.1.8 FCS_CKM_EXT.6/EE Cryptographic Key Destruction Types *(FDE_EE)*

The TOE clears its keys in accordance with FCS_CKM.4(d).

6.1.9 FCS_COP.1(a) Cryptographic Operation (Signature Verification) *(FDE_AA) (FDE_EE)*

The TOE's only use of signature verification is verifying TOE updates. Refer to section 6.4.6 (FPT_TUD_EXT.1) for a description of TOE Updates.

6.1.10 FCS_COP.1(b) Cryptographic Operation (Hash Algorithm) *(FDE_AA) (FDE_EE)*

The TOE's OpenSSL libraries provides SHA-256, SHA-384 and SHA-512 algorithms that meet ISO/IEC 10118-3:2004. The application space OpenSSL library (OpenSSL version 1.1.1k) provides SHA-256, SHA-384 and SHA-512 algorithms. The kernel space OpenSSL library (Linux Kernel Crypto API 4.18.0) provides SHA-512.

The SHA-256 algorithm is used by EE (in application space) when calculating the DEK digest, which is used to validate the BEV. SHA-384 is used by TOE (AA and EE) when verifying ECDSA digital signatures of the Archon OS updates signed by CACI. The SHA-512 algorithm is used by AA as part of PBKDF2 password-based key derivation, by the EE as part of PBKDF2 on the candidate DEK, and by the TOE (both AA and EE) when a trusted update signature verification of the software signed by RHEL.

6.1.11 FCS_COP.1(c)/AA Cryptographic Operation (Keyed Hash Algorithm) *(FDE_AA)* and FCS_COP.1(c)/EE Cryptographic Operation (Message Authentication) *(FDE_EE)*

The TOE implements:

- HMAC-SHA-256 using 256-bit keys, SHA-256 hash algorithm, a 512-bit block size, and an output MAC length of 256 bits
- HMAC-SHA-512 using 512-bit keys, SHA-512 hash algorithm, a 512-bit block size, and an output MAC length of 256 bits

6.1.12 FCS_COP.1(f)/EE Cryptographic Operation (AES Data Encryption/Decryption) *(FDE_EE)*

The TOE supports Full Drive Encryption and Decryption. Encryption and Decryption is implemented by the kernel and uses AES with XTS mode and cryptographic key size of 256 bits. AES as specified in ISO/IEC 18033-3; XTS is specified in IEEE 1619.

6.1.13 FCS_COP.1(g)/EE Cryptographic Operation (Key Encryption) *(FDE_EE)*

The TOE has an AES CBC implementation used for key management operations (encryption and decryption of the DEK). This implementation uses AES-CBC with 256-bit keys. The TOE encrypts the DEK with AES-CBC before it stores the DEK in the LUKS Header keyslot.

6.1.14 FCS_KDF_EXT.1/AA Cryptographic Key Derivation *(FDE_AA)*

The TOE uses NIST SP 800-132 PBKDF2 with HMAC-SHA-512, a number of iterations, and a 256 bit salt to transform the passphrase into a derived key (KEK). The KEK is transferred to the EE LUKS (the BEV) and used to encrypt and decrypt the DEK in the key slot associated with the passphrase.

The number of iterations is configured as 10,000 in the “cryptsetup reencrypt” CLI command.

6.1.15 FCS_KYC_EXT.1/AA Key Chaining (Initiator) *(FDE_AA)*

The TOE uses PBKDF2 to transform the user’s passphrase into a 256-bit KEK and passes this value to LUKS EE (as the BEV (Border Encrypt Value)).

6.1.16 FCS_KYC_EXT.2/EE Key Chaining (Recipient) *(FDE_EE)*

LUKS EE uses the BEV, created by LUKS AA, to AES-CBC decrypt the key-slot DEK stored in the LUKS header.

6.1.17 FCS_PCC_EXT.1/AA Cryptographic Password Construct and Conditioning *(FDE_AA)*

User guidance, [AGD_LUKS], instructs that passphrases must contain at least one character from each of the following four character groups: upper case, lower case, numbers, and special characters ‘!’, ‘@’, ‘#’, ‘$’, ‘%’, ‘^’, ‘&’, ‘*’, ‘(‘, ‘)’, ‘<’, ‘>’, ‘,’, ‘.’, ‘/’, ‘?’, ‘|’, ‘\’, ‘:’, ‘;’, ‘"’, ‘'’, ‘-‘, ‘+’ and ‘~’. A passphrase may not contain more than four consecutive characters from any of the four categories.

Administrators are instructed in the user guidance to configure the minimum passphrase to 12 when they receive their system. Archon OS does not support a CLI command or configuration parameter that enables an Administrator to set a maximum passphrase length. However, the CC evaluated configuration requires the system support of at least 64 characters of passphrase length.

The KEK/BEV is derived from the passphrase using PBKDF2. The salt, hash (SHA512), and PBDKF interactions (10000) are obtained from the LUKS header to create the KEK. The KEK(DEK) is passed to the EE functionality and used to unencrypt the DEK.

6.1.18 FCS_RBG_EXT.1/OSSL Random Bit Generation *(FDE_AA) (FDE_EE)*

The OpenSSL CTR_DRBG is used by AA (cryptsetup):

- To generate salts

The OpenSSL CTR_DRBG is used by EE (cryptsetup):

- To generate salts
- To generate IVs
- To generate the DEK

Archon OS (Operational Environment) offers two Deterministic Random Bit Generators (DRBGs), one in the kernel and the other in application space. The OpenSSL CTR_DRBG in application space receives a seed of at least 256 bits of entropy. This DRBG is utilized for generating session and ephemeral keys during TLS protocol negotiation, as well as for all other instances requiring entropy, such as nonce generation.

The second DRBG is a kernel cryptographic API featuring an HMAC_DRBG mechanism that receives a seed of at least 384 bits of entropy from the TOE's noise source. This particular DRBG is employed to produce random output for key generation and to supply seed material to the OpenSSL DRBG or TOE applications when invoked using /dev/random, /dev/urandom, or the 'getrandom' system call.

A full description of the entropy source is described in a proprietary ancillary Entropy Assessment Report made available at the discretion of the developer.

6.1.19 FCS_RBG_EXT.1/KERN Random Bit Generation *(FDE_EE)*

The Kernel HMAC_DRBG (SHA-512) is used by EE (`dm-crypt`) to generate AES-XTS tweaks.

A full description of the entropy source is described in a proprietary ancillary Entropy Assessment Report made available at the discretion of the developer.

6.1.20 FCS_SNI_EXT.1/AA Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_AA)*

The TOE generates its salts using the CTR_DRBG (OSSL RBG). The TOE generates no nonces and uses no IVs.

6.1.21 FCS_SNI_EXT.1/EE Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_EE)*

The TOE generates its salts and DEKs using the CTR_DRBG (OSSL RBG). The TOE generates its AES-CBC IVs using ESSIV:SHA256 (OSSL RBG). The TOE generates no nonces but generates 256-bit AES XTS tweaks (used for data partition encryption) using its HMAC_DRBG (KERN RBG).

6.1.22 FCS_VAL_EXT.1/EE Validation *(FDE_EE)*

The TOE validates the BEV (and therefore Archon User's passphrase) by first subjecting the passphrase and salt to PBKDF2 to form the derived key (KEK/BEV). The TOE uses the KEK to decrypt the DEK, stored in the LUKS header. However, before using the DEK, the TOE first validates the DEK by performing PBKDF2

(10,000 iterations of HMAC-SHA-256) on the candidate DEK and salt stored in the header, and then compares the resulting value to the DEK digest stored in the header to ensure the two match.

In this TOE, the BEV is the submask. The TOE does not use multiple submasks in its cryptographic process.

6.2 User Data Protection (FDP)

6.2.1 FDP_DSK_EXT.1/EE Protection of Data on Disk *(FDE_EE)*

The TOE provides FDE that encrypts the entire drive using AES-256 XTS block based encryption. LUKS operates on a block level. It encrypts whole partitions or disks rather than individual files. LUKS protects data by encrypting it inside physical and logical disk partitions so that only authorized users can access the content of that encrypted partition. It uses the partition header, created by LUKS at setup, to store the necessary encryption/decryption information such as the encryption algorithm and key size.

At LUKS setup, LUKS creates an encrypted container called LUKS volume on a disk partition. It uses AES-XTS to encrypt the volume which can only be accessed using a passphrase. When Administrators unlock the LUKS volume at system boot with a correct passphrase, it becomes accessible like a regular block device. So, the user can read from it and write to it.

LUKS stores its encryption metadata at the beginning of the encrypted partition called the LUKS header. This header contains the cipher and mode, hash function, and key slots. The actual encrypting of the partition is done using a master key. This master key is randomly generated when the LUKS setup is initialized. This master key is what directly encrypts and decrypts the data on the partition. The passphrase doesn't encrypt the data. Instead, it encrypts the master key stored in the key slots.

To access the encrypted data, LUKS requires users to enter a passphrase. This passphrase is then used for decrypting the master key stored in one of the key slots. After the master key is decrypted, that key in turn is used to decrypt the data on the partition.

LUKS encrypts the entire partition except the LUKS header and the initramfs first root filesystem.

The system boot chain consists of the following steps:

- Hardware responsibility
  - Firmware initialization
  - First stage boot loader (shim.efi)
- Archon OS (OE) responsibility
  - Second Stage Boot Loader (GRUB 2)
- Archon LUKS (TOE) responsibility
  - First root filesystem (`initramfs`)
  - Prompt for LUKS passphrase
- Archon OS (OE) responsibility
  - Linux kernel (drivers and modules)

Secure Boot is a UEFI firmware security feature developed by the UEFI Consortium that ensures only immutable and signed software is loaded during the boot time. The first application loaded by the platform’s firmware is the signed and trusted first-stage boot loader (shim.efi). This shim package itself holds the signing certificate and its own databases of trusted keys and hashes that are allowed to be loaded. The shim package’s signature is verified by the signing certificate’s RSA 2048 public key included

in the shim package. The shim then uses this public key to verify the signature on the code signing public key held in the database. This code signing key is used to verify the signature of the second stage boot loader, GRUB 2 (grubx64.efi). Next, GRUB 2 uses the code signing key to verify the signature on the first root filesystem (initramfs). Initramfs then uses RSA 4096 code (SHA 512) signing keys, from the database, to verify the signatures of the OS kernel.

Refer to the User Guidance for more information about the TOE's initialization process and setup.

6.3 Security Management (FMT)

The management of users, groups, passwords, and access is provided by the operational environment (Archon OS). Entry to the TOE is via the CLI commands.

6.3.1 FMT_MOF.1/AA Management of Functions Behavior *(FDE_AA)*

The TOE, in the CC evaluated configuration, only supports two compliant-power saving states. These cannot be modified while the TOE is configured in the evaluated configuration. Only authorized administrators can issue the shutdown command. The only role supported by the TOE's is authorized administrators. The authorization factor used is the BEV, which corresponds to the Archon User's passphrase allowing the user to exit the compliant power-saving state.

6.3.2 FMT_SMF.1/AA Specification of Management Functions *(FDE_AA)* and FMT_SMF.1/EE Specification of Management Functions *(FDE_EE)*

The TOE provides each of the required management services with no additional ones. Because the TOE fulfills the AA and EE requirements together, the TOE does not need to "forward" requests to change the DEK or cryptographically erase the DEK.

The TOE allows the Users to perform the following management functions:

a) Change the DEK (FMT_SMF.1.1a/AA and FMT_SMF.1.1a/EE) (Section 6.3.2.1),
b) Erase the DEK (FMT_SMF.1.1b/AA and FMT_SMF.1.1b/EE) (Section 6.3.2.2),
c) Change the passphrase (FMT_SMF.1.1c/AA) (Section 6.3.2.3),
d) Initiate a TOE update (FMT_SMF.1.1d/AA and FMT_SMF.1.1c/EE) (Section 6.3.2.4),
e) Change the default authorization factor. (FMT_SMF.1.1d/EE) (Section 6.3.2.5),

The management of users, groups, passwords, and access is provided by the operational environment (Archon OS). Entry to the TOE is via the `cryptsetup` CLI commands. Initiating a TOE update is done by the OE.

User Guidance ([AGD_LUKS]) provides specific information about all of the management commands.

6.3.2.1 Change the DEK (AA and EE)

- `cryptsetup reencrypt --cipher aes-xts-plain64 --key-size 512 --pbkdf PBKDF2 --hash sha512 --pbkdf-force-iterations 10000 /dev/<LUKS partition name>`

`cryptsetup reencrypt` CLI command s used to re-encrypt the data in a LUKS partition with a new key (DEK). The user must give the cipher (AES-XTS), key size (512), PBKDF (Password-Based Key Derivation Function) algorithm (PBKDF2), hash (SHA512), iterations (10000), and the name of the physical volume to be protected. The user is prompted for the current passphrase. The passphrase does not change once the command is completed.

6.3.2.2 Erase the DEK (AA and EE)

The DEK can be erased by the user by issuing either of the following commands:

- `cryptsetup luksErase`
- `cryptsetup reencrypt --decrypt`

`cryptsetup luksErase` is used to perform a crypto erase (wipe) of the key (DEK). Data in the LUKS partition is still encrypted and is no longer accessible since the DEK is erased.

`cryptsetup reencrypt --decrypt` is used to decrypt all data in the LUKS partition and disable LUKS functionality. It also erases the key (DEK). Note that execution of this command takes the TOE out of its CC evaluated configuration.

6.3.2.3 Change the Passphrase (AA only)

Users may change their passphrase by invoking the `cryptsetup luksChangeKey` command. Refer to the administrator guidance for the command details.

6.3.2.4 Initiate a TOE Update (AA and EE)

Users may update the TOE. Refer to Section 6.4.6 for a full description. User Guidance ([AGD_LUKS]) provides specific information about updating the TOE.

6.3.2.5 Change the Default Authorization Factor

When the TOE is initially installed and configured by CACI personnel, a default LUKS passphrase Is created by the installer. This passphrase is sent to the customer via an out-of-band mechanism. The customer is instructed in the user guidance ([AGD_LUKS]) to change the default passphrase upon receipt of their system and how to change the password (Archon User issues a CLI command from the system keyboard). Refer to section 6.3.2.3 to change the passphrase.

6.3.3 FMT_SMR.1/AA Security Roles *(FDE_AA)*

The TOE support allows only authorized users to login to as a LUKS User and Archon User. A LUKS User is a user who knows the LUKS passphrase which is entered at system boot. Correctly entering the passphrase enables that user access to data on the drive. Additionally, the Operational Environment (OE), Archon OS, imposes roles for its users: Archon Users. Privileged Archon Users can perform additional actions and modify more TSF data than non-privileged Archon Users. The OE roles supersede the LUKS User role. Archon LUKS supports only Archon Users with Administrator privileges.

6.4 Protection of the TSF (FPT)

6.4.1 FPT_KYP_EXT.1/AA Protection of Key and Key Material *(FDE_AA)*

The AA functionality of the TOE does not store keys in non-volatile memory.

NOTE: Updated as per TD0458.

6.4.2 FPT_KYP_EXT.1/EE Protection of Key and Key Material *(FDE_EE)*

The DEK is created when the disk is encrypted. The TOE stores the encrypted DEKs in the LUKS header of the disk (keyslots). The encryption key is the KEK derived from a user-supplied passphrase.

NOTE: Updated as per TD0458.

6.4.3 FPT_PWR_EXT.1/AA Power Saving States *(FDE_AA)* and FPT_PWR_EXT.1/EE Power Saving States *(FDE_EE)*

The TOE supports two Compliant-power saving states:

- G2(S5) – Soft off state. The soft off state is when the system fully shuts down without a hibernation file. Soft off is also known as a full shutdown. During a full shutdown and boot, the entire user session is torn down and restarted on the next boot. Consequently, a boot/startup from this state takes significantly longer than S1-S4. A full shutdown (S5) occurs when a system shut down is requested or when an application calls a shutdown API.
- G3 – Mechanical off state. In this state, the system is completely off and consumes no power. The system returns to the working state only after a full reboot.

The drive is only assumed secure when in a powered-off state up until it is powered on and receives the initial LUKS authorization factor.

From a key destruction perspective, power is not supplied to RAM in G2(S5) nor G3. During a restart, the time spent in G2(S5) is sufficient to clear RAM.

6.4.4 FPT_PWR_EXT.2/AA Timing of Power Saving States *(FDE_AA)* and FPT_PWR_EXT.2/EE Timing of Power Saving States *(FDE_EE)*

G2(S5) is entered following an administrator issuing a `shutdown` command.

G3 is entered when the administrator turns off the system or the system loses physical power.

6.4.5 FPT_TST_EXT.1 TSF Testing *(FDE_AA) (FDE_EE)*

The TOE runs a group of self-tests at power-up referred to as FIPS Power-On Self Tests. This test suite is provided by Archon OS (Operational Environment) and includes two sets of tests: pre-operational software integrity tests and Cryptographic Algorithm Self-Tests (CASTs) on all approved cryptographic algorithms.

The pre-operational software integrity tests are performed automatically when the system is powered on, before the system transitions into the operational state. The algorithms used for the integrity tests run their CASTs (Cryptographic Algorithm Self-Tests) before the integrity test is performed. While the module is executing the pre-operation self-tests, services are not available, and data output (via the data output interface) is inhibited until the pre-operational software integrity self-tests are successfully completed. The system transitions to the operational state only after the pre-operational self-tests are passed successfully. If either of these tests fail, the system transitions to the Error State. The following table identifies the details of the pre-operational integrity tests.

Table 11: Pre-Operational Self-Tests

Algorithm	Test Properties	Test Method	Test Type	Details
HMAC-SHA2-512	128-bit key	Message Authentication	SW/FW Integrity	Integrity tests for: • initramfs – the compressed, bootable kernel image.

Algorithm	Test Properties	Test Method	Test Type	Details
				• sha512hmac – verifies HMAC and unkeyed checksum values for the contents files.
RSA SigVer FIPS186-4)	3072-bit key with SHA-256	Signature Verification	SW/FW Integrity	Integrity test for kernel object files.

Once the integrity tests pass and the system has entered into the operational state, the system performs self-tests on all approved cryptographic algorithms that are part of the approved services supported in the approved mode of operation. The tests type run is CASTs using the KAT (Known Answer Tests) method. The tests run, include but are not limited to, the algorithms identified Section 6.5 TCAVP Algorithm Certificate Details. During these tests, services are not available, and data output (via the data output interface) is inhibited during the conditional self-tests. If any of these tests fail, the system transitions to the Error State.

6.4.6 FPT_TUD_EXT.1 Trusted Update *(FDE_AA) (FDE_EE)*

The Operational Environment provides Archon Users the ability to verify the current version on the TOE. Operational Guidance instructs users to view the file `/etc/os-release` file in order to display the TOE's version. Additionally, the Operational Guidance includes a definition of the TOE's version fields (v3.0.0.2).

The Operational Environment (Archon OS) provides the ability for Archon Users to initiate updates to the TOE. An Update Server (a mandatory system required in the OE), is leveraged for downloading the TOE update.

To initiate an update, the administrator invokes the `rpm-ostree rebase` command. This command downloads the latest Archon OS from the Update Server.

Once the TOE download is complete, Archon OS verifies (authenticates) the digital signature on the bundle using the RTU (Root of Trust for Update). The TOE distribution includes two levels of signature: the complete Archon OS distribution is signed by CACI and within the Archon OS build, there are unmodified signed software packages signed by RHEL. CACI signs Archon OS distribution using ECDSA P-384. RHEL signs its software packages using RSA 4096 with SHA-384.

If any signature verification fails, the system will display an error message, halt, and the upgrade will not proceed. There is no indication that the signature verification was successful.

Updating the TOE is supported by the TOE's Operational Environment (Archon OS). Refer to the user guidance for instructions about updating the TOE.

6.5 TCAVP Algorithm Certificate Details

Each of these cryptographic algorithms have been validated as identified in the table below .

Table 12: CAVP Algorithms

Algorithm	Standard	Modes Supported	Library	CAVP Certificate
Cryptographic Operation (Signature Verification) *(FDE_AA) (FDE_EE)* (FCS_COP.1(a))				
ECDSA SigGen (FIPS186-4)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" [P-384]; ISO/IEC 14888-3, Section 6.4.	P-384 with SHA-384	OpenSSL version 1.1.1k	A5342
ECDSA SigVer (FIPS186-4)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" [P-384]; ISO/IEC 14888-3, Section 6.4.	P-384 with SHA-384	OpenSSL version 1.1.1k	A5342
RSA SigGen (FIPS 186-4)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1-v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3, for RSA schemes	PKCS 1.5 with modulo 4096	OpenSSL version 1.1.1k	A5342
RSA SigVer (FIPS 186-4)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1-v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3, for RSA schemes	PKCS 1.5 with modulo 4096	OpenSSL version 1.1.1k	A5342

Algorithm	Standard	Modes Supported	Library	CAVP Certificate
Cryptographic Operation (Hash Algorithm) *(FDE_AA) (FDE_EE)* (FCS_COP.1(b))				
SHA2-256	ISO/IEC 10118-3:2004		OpenSSL version 1.1.1k	A5342
SHA2-384				
SHA2-512				
SHA2-512	ISO/IEC 10118-3:2004		Linux Kernel Crypto API 4.18.0	A5343
Cryptographic Operation (Keyed Hash Algorithm) *(FDE_AA)* (FCS_COP.1(c)/AA)				
HMAC-SHA-256	ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"	256 bits key length	OpenSSL version 1.1.1k	A5342
HMAC-SHA-512		512 bits key length	OpenSSL version 1.1.1k	A5342
Cryptographic Operation (Message Authentication) *(FDE_EE)* (FCS_COP.1(c)/EE)				
HMAC-SHA-256	ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"	256 bits key length	OpenSSL version 1.1.1k	A5342
HMAC-SHA-512		512 bits key length	OpenSSL version 1.1.1k	A5342
HMAC-SHA-512		512 bits key length	Linux Kernel Crypto API 4.18.0	A5343
Cryptographic Operation (AES Data Encryption/Decryption) *(FDE_EE)* (FCS_COP.1(f)/EE)				
AES-XTS Testing Revision 2.0	AES: ISO/IEC 18033-3 XTS: IEEE 1619	256 bits key length (x2) Payload Length: 4096	Linux Kernel Crypto API 4.18.0	A5343
Cryptographic Operation (Key Encryption) *(FDE_EE)* (FCS_COP.1(g)/EE)				
AES-CBC	AES: ISO/IEC 18033-3 CBC as specified in ISO/IEC 10116	Key Length: 256 bits	OpenSSL version 1.1.1k	A5342
Random Bit Generation *(FDE_AA) (FDE_EE)* (FCS_RBG_EXT.1/OSSL)				
CTR_DRBG	Random bit generation (DRBG) services in accordance with NIST Special	AES-256	OpenSSL version 1.1.1k	A5342

Algorithm	Standard	Modes Supported	Library	CAVP Certificate
	Publication 800-90A using CTR_DRBG (AES)			
Random Bit Generation *(FDE_EE)* (FCS_RBG_EXT.1/KERN)				
HMAC_DRBG	Random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using HMAC_DRBG (any)	HMAC-SHA-512	Linux Kernel Crypto API 4.18.0	A5343

6.6 Cryptographic Key Destruction

The table below describes the key zeroization provided by the TOE and as referenced in FCS_CKM.4.

Table 13: Key Zeroization

Keys/CSPs	Storage Location	How Key is Protected	How Key is Derived	Strength of Key	When Key is Destroyed	Method of Zeroization
User Passphrase	RAM	N/A	N/A	N/A	Shutdown.	Removal of power.
Derived Key (KEK) (BEV)	RAM	N/A	The TOE uses 800-132 in counter mode using HMAC-SHA-256 and 10000 iterations and a 256 bit salt to transform the operator's password into a Derived Key	256 bits	Shutdown & removal of power.	Overwrite with zeros.
DEK	RAM	N/A	Generated from approved DRBG at disk encryption time	256 bits	Shutdown and removal of power.	Overwrite with zeros.
	On Disk in the LUKS header	AES CBC Encrypted			User invoked CLI command.	Overwrite with zeros.

7 Terms and Acronyms

Table 14: Terms and Definitions

Terms	Definition
Archon OS User	Administrator of the TOE.
BEV	The key(s) that is based from the AA to EE. This is a name specific to the cPPs.
LUKS User	Transparent user of the TOE (after supplying the passphrase).
VK Digest	A digest takes a plain text and generates a hashcode which can be used to verify if the plain text is unmodified but *cannot* be used to decrypt the original text from the hash value.

The following acronyms and abbreviations appear in the document and are defined in the table below.

Table 15: Acronyms and Abbreviations

Acronym/Abbreviations	Definition
AA	Authorization Acquisition
AES	Advanced Encryption Standard
BEV	Border Encrypt Value
CAST	Cryptographic Algorithm Self-Test
CBC	Cipher Block Chaining
CC	Common Criteria
CLI	Command Line Interface
cPP	collaborative Protection Profile
CPU	Central Processing Unit
CSP	Cryptographic Service Provider
DEK	Data Encryption Key
DRBG	Deterministic Random Bit Generator
EE	Encryption Engine
EUD	End Unit Device
FDE	Full Disk Encryption or Full Drive Encryption
FDE_AA cPP	*collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition,* Version 2.0 + Errata 20190201, February 1, 2019
FDE_EE cPP	*collaborative Protection Profile for Full Drive Encryption – Encryption Engineer,* Version 2.0 + Errata 20190201, February 1, 2019

Acronym/Abbreviations	Definition
FIPS	Federal Information Processing Standards
GUI	Graphical User Interface
HMAC	Keyed-Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTP/TLS	HTTP over TLS or HTTPS
IP	Internet Protocol
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
IV	Initialization Vector
KEK	Key Encryption Key
LAN	Local Area Network
LUKS	Linux Unified Key Setup
MAC	Message Authentication Code
MEK	Master Key or Master Encryption Key
NIAP	Nation Information Assurance Partnership
OS	Operating System
PBKDF2	Password-Based Key Derivation Function 2
PKCS	Public-Key Cryptography Standards
POST	Power-On Self-Test
PP	Protection Profile
PRF	Pseudorandom Function
RBG	Random Bit Generator
RSA	Rivest, Shamir, & Adleman
RTU	Root of Trust for Update
SFR	Security Functional Requirement
ST	Security Target
SWFDE	SoftWare Full Drive Encryption
TLS	Transport Layer Security
TOE	Target of Evaluation
TSS	TOE Summary Specification

Acronym/Abbreviations	Definition
UUID	Universally Unique Identifier
VK	Volume Key
V*n*	Version
XTS	XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing

Appendix A FDE_AA and FDE_EE SFR Tables

The following tables are to ensure that each cPP is complete.

Key:
M = Mandatory SFR
S = Selection Based SFR
O = Optional SFR

Table 16: FDE_AA cPP SFRs

Requirement	M/S	SFR That Prompts the Inclusion of the Selection-Based SFR
FCS_AFA_EXT.1/AA Authorization Factor Acquisition *(FDE_AA)*	M	N/A
FCS_AFA_EXT.2/AA Timing of Authorization Factor Acquisition *(FDE_AA)*	M	N/A
FCS_CKM.4(a)/AA Cryptographic Key Destruction (Power Management) *(FDE_AA)*	M	N/A
FCS_CKM.4(d) Cryptographic Key Destruction (Software TOE, 3rd Party Storage) *(FDE_AA) (FDE_EE)*	M	N/A
FCS_CKM_EXT.4(a) Cryptographic Key and Key Material Destruction (Destruction Timing) *(FDE_AA) (FDE_EE)*	M	N/A
FCS_CKM_EXT.4(b)/AA Cryptographic Key and Key Material Destruction (Power Management) *(FDE_AA)*	M	N/A
FCS_COP.1(a) Cryptographic Operation (Signature Verification) *(FDE_AA) (FDE_EE)*	S	FPT_TUD_EXT.1 Trusted Update *(FDE_AA) (FDE_EE)* (M)
FCS_COP.1(b) Cryptographic Operation (Hash Algorithm) *(FDE_AA) (FDE_EE)*	S	FCS_COP.1(c)/AA Cryptographic Operation (Keyed Hash Algorithm) *(FDE_AA)* (S)
FCS_COP.1(c)/AA Cryptographic Operation (Keyed Hash Algorithm) *(FDE_AA)*	S	FCS_KDF_EXT.1/AA Cryptographic Key Derivation *(FDE_AA)* (S)
FCS_KDF_EXT.1/AA Cryptographic Key Derivation *(FDE_AA)*	S	FCS_KYC_EXT.1/AA Key Chaining (Initiator) *(FDE_AA)* (M)
FCS_KYC_EXT.1/AA Key Chaining (Initiator) *(FDE_AA)*	M	N/A
FCS_PCC_EXT.1/AA Cryptographic Password Construct and Conditioning *(FDE_AA)*	S	FCS_AFA_EXT.1/AA Authorization Factor Acquisition *(FDE_AA)* (M)
FCS_RBG_EXT.1/OSSL Cryptographic Operation (Random Bit Generation) *(FDE_AA) (FDE_EE)*	S	FCS_SNI_EXT.1/AA Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_AA)* (M)
FCS_SNI_EXT.1/AA Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_AA)*	M	N/A
FMT_MOF.1/AA Management of Functions Behavior *(FDE_AA)*	M	N/A

Requirement	M/S	SFR That Prompts the Inclusion of the Selection-Based SFR
FMT_SMF.1/AA Specification of Management Functions *(FDE_AA)*	M	N/A
FMT_SMR.1/AA Security Roles *(FDE_AA)*	M	N/A
FPT_KYP_EXT.1/AA Protection of Key and Key Material *(FDE_AA)*	M	N/A
FPT_PWR_EXT.1/AA Power Savings States *(FDE_AA)*	M	N/A
FPT_PWR_EXT.2/AA Timing of Power Savings States *(FDE_AA)*	M	N/A
FPT_TST_EXT.1 TSF Testing *(FDE_AA) (FDE_EE)*	O	N/A
FPT_TUD_EXT.1 Trusted Update *(FDE_AA)*	M	N/A

Table 17: FDE_EE cPP SFRs

Requirement	**M/S**	**SFR That Prompts the Inclusion of the Selection-Based SFR**
FCS_CKM.1(c)/EE Cryptographic Key Generation (Data Encryption Key) *(FDE_EE)*	M	N/A
FCS_CKM.4(a)/EE Cryptographic Key Destruction (Power Management) *(FDE_EE)*	M	N/A
FCS_CKM.4(d) Cryptographic Key Destruction (Software TOE, 3rd Party Storage) *(FDE_AA) (FDE_EE)*	S	FCS_CKM_EXT.6/EE Cryptographic Key Destruction Types *(FDE_EE)* (M)
FCS_CKM_EXT.4(a) Cryptographic Key and Key Material Destruction (Destruction Timing) *(FDE_AA) (FDE_EE)*	M	N/A
FCS_CKM_EXT.4(b)/EE Cryptographic Key and Key Material Destruction (Power Management) *(FDE_EE)*	M	N/A
FCS_CKM_EXT.6/EE Cryptographic Key Destruction Types *(FDE_EE)*	M	N/A
FCS_COP.1(a) Cryptographic Operation (Signature Verification) *(FDE_AA) (FDE_EE)*	S	FPT_TUD_EXT.1 Trusted Update *(FDE_AA) (FDE_EE)* (M)
FCS_COP.1(b) Cryptographic Operation (Hash Algorithm) *(FDE_AA) (FDE_EE)*	S	FCS_COP.1(a) Cryptographic Operation (Signature Verification) *(FDE_AA) (FDE_EE)* (S)
FCS_COP.1(c)/EE Cryptographic Operation (Message Authentication) *(FDE_EE)*	S	FCS_VAL_EXT.1/EE Validation *(FDE_EE)* (M)
FCS_COP.1(f)/EE Cryptographic Operation (AES Data Encryption/Decryption) *(FDE_EE)*	S	FDP_DSK_EXT.1/EE Protection of Data on Disk *(FDE_EE)* (M)
FCS_COP.1(g)/EE Cryptographic Operation (Key Encryption) *(FDE_EE)*	S	FCS_KYC_EXT.2/EE Key Chaining (Recipient) *(FDE_EE)* (M)
FCS_KYC_EXT.2/EE Key Chaining (Recipient) *(FDE_EE)*	M	N/A

Requirement	M/S	SFR That Prompts the Inclusion of the Selection-Based SFR
FCS_RBG_EXT.1/OSSL Cryptographic Operation (Random Bit Generation) *(FDE_AA) (FDE_EE)*	S	FCS_SNI_EXT.1/EE Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_EE)* (M) For generation of salts and IVs. FCS_CKM.1(c)/EE Cryptographic Key Generation (Data Encryption Key) *(FDE_EE)* For generation of DEK.
FCS_RBG_EXT.1/KERN Cryptographic Operation (Random Bit Generation) *(FDE_EE)*	S	FCS_SNI_EXT.1/EE Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_EE)* (M) For generation of Tweaks.
FCS_SNI_EXT.1/EE Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) *(FDE_EE)*	M	N/A
FCS_VAL_EXT.1/EE Validation *(FDE_EE)*	M	N/A
FDP_DSK_EXT.1/EE Protection of Data on Disk *(FDE_EE)*	M	N/A
FMT_SMF.1/EE Specification Management Functions *(FDE_EE)*	M	N/A
FPT_KYP_EXT.1/EE Protection of Key and Key Material *(FDE_EE)*	M	N/A
FPT_PWR_EXT.1/EE Power Savings States *(FDE_EE)*	M	N/A
FPT_PWR_EXT.2/EE Timing of Power Savings States *(FDE_EE)*	M	N/A
FPT_TST_EXT.1 TSF Testing *(FDE_AA) (FDE_EE)*	M	N/A
FPT_TUD_EXT.1 Trusted Update *(FDE_AA) (FDE_EE)*	M	N/A