RackTop Systems

RackTop SPEAR v1.0

Security Target



Document Version: 1.2

Prepared for:

RACKTOP

RackTop Systems
13454 Sunrise Valley Drive
Suite 550
Herndon, VA 20171
United States of America
Phone: +1 443 276 6973
www.racktopsystems.com

Prepared by:

Corsec

Corsec Security, Inc.
12600 Fair Lakes Circle
Suite 210
Fairfax, VA 22033
United States of America
Phone: +1 703 267 6050
www.corsec.com

Table of Contents

1. Introduction 5
1.1 Purpose 5
1.2 Security Target and TOE References 5
1.3 Product Overview 6
1.4 TOE Overview 6
1.5 TOE Environment 6
1.6 TOE Description 7
1.6.1 Physical Scope 7
1.6.2 TOE Delivery 8
1.6.3 Logical Scope 8
1.6.4 Product Physical/Logical Features and Functionality not included in the TOE 9
2. Conformance Claims 10
3. Security Problem Definition 11
3.1 Threats 11
3.2 Organizational Security Policy 14
4. Security Objectives 15
4.1 Security Objectives for the Operational Environment 15
5. Extended Components 17
6. Security Assurance Requirements 18
7. Security Functional Requirements 19
7.1 Conventions 19
7.2 Security Functional Requirements 19
7.2.1 Class FCS: Cryptographic Support 20
FCS_CKM_EXT.4(a)/AA Cryptographic Key and Key Material Destruction (Destruction Timing) 22
FCS_CKM_EXT.4(a)/EE Cryptographic Key and Key Material Destruction (Destruction Timing) 22
FCS_CKM_EXT.4(b)/AA Cryptographic Key and Key Material Destruction (Power Management) 22
FCS_CKM_EXT.4(b)/EE Cryptographic Key and Key Material Destruction (Power Management) 22
FCS_CKM_EXT.6/EE Cryptographic Key Destruction Types 22
FCS_COP.1(a)/AA&EE Cryptographic Operation (Signature Verification) 22
FCS_COP.1(b)/AA&EE Cryptographic Operation (Hash Algorithm) 23
FCS_COP.1(c)/AA Cryptographic Operation (Keyed Hash Algorithm) 23
FCS_COP.1(c)/EE Cryptographic Operation (Message Authentication) 23
FCS_COP.1(f)/EE Cryptographic Operation (AES Data Encryption/Decryption) 23
FCS_COP.1(g)/EE Cryptographic Operation (Key Encryption) 23
FCS_KYC_EXT.1/AA Key Chaining (Initiator) 23
FCS_KYC_EXT.2/EE Key Chaining (Recipient) 23
FCS_PCC_EXT.1/AA Cryptographic Password Construct and Conditioning 24
FCS_SNI_EXT.1/AA Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) 24
FCS_SNI_EXT.1/EE Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) 24
FCS_VAL_EXT.1/EE Validation 24
FCS_KDF_EXT.1/EE Cryptographic Key Derivation 25
7.2.2 Class FDP: User Data Protection 25
FDP_DSK_EXT.1/EE Protection of Data on Disk 25
7.2.3 Class FMT: Security Management 25
FMT_MOF.1/AA Management of Functions Behavior 25

 ©2026 RackTop Systems

FMT_SMF.1/AA Specification of Management Functions 25
FMT_SMF.1/EE Specification of Management Functions 25
FMT_SMR.1/AA Security Roles 26
7.2.4 Class FPT: Protection of the TSF 26
FPT_PWR_EXT.1/AA&EE Power Saving States 26
FPT_PWR_EXT.2/AA Timing of Power Saving States 26
FPT_PWR_EXT.2/EE Timing of Power Saving States 26
FPT_TST_EXT.1/EE TSF Testing 26
FPT_TUD_EXT.1/AA Trusted Update 27
FPT_TUD_EXT.1/EE Trusted Update 27
8. TOE Summary Specification 28
8.1 TOE Security Functionality 28
8.1.1 Cryptographic Support 29
8.1.2 User Data Protection 36
8.1.3 Security Management 37
8.1.4 Protection of the TSF 38
9. Rationale 42
9.1 Conformance Claims Rationale 42
9.1.1 Variance Between the PP and this ST 42
9.1.2 Security Assurance Requirements Rationale 42
9.1.3 Dependency Rationale 42
10. Acronyms and Terms 43
10.1 Acronyms 43

List of Figures

Figure 1 – TOE Boundary 7

List of Tables

Table 1 – ST and TOE References 5
Table 2 – Guidance Documentation 8
Table 3 – CC and PP Conformance 10
Table 4 – NIAP FDE AA and FDE EE Technical Decisions 10
Table 5 – Security Assurance Requirements 18
Table 6 – TOE Security Functional Requirements 19
Table 7 – Mapping of TOE Security Functionality to Security Functional Requirements 28
Table 8 - CAVP Algorithm Certificate References 29
Table 9 - Files required for validation of the software packages 32
Table 10 - Kernal Crypto Module KATs 40
Table 11 – OpenSSL Crypto Module (user space) KATs 40
Table 12 – Acronyms 43

©2026 RackTop Systems

Revision History

Version	Modification Date	Modified By	Description of Changes
0.1	2025-09-05	Dayanandini Pathmanathan	Initial draft.
0.2	2025-09-11	Dayanandini Pathmanathan	Addressed customer comments and submission to lab.
0.3	2025-09-23	Dayanandini Pathmanathan	Addressing lab comments.
0.4	2026-01-26	Dayanandini Pathmanathan	Addressing lab comments.
0.5	2026-02-06	Dayanandini Pathmanathan	Addressing lab comments.
0.6	2026-03-12	Dayanandini Pathmanathan	Addressing lab comments.
0.7	2026-03-24	Kathleen Moyer	Removing SHA-256 and adding SHA-512
0.8	2026-04-07	Dayanandini Pathmanathan	Adding CAVP certificate numbers.
0.9	2026-05-20	Dayanandini Pathmanathan	Internal reviews and submission to lab.
1.0	2026-08-05	Dayanandini Pathmanathan	Addressing lab comments.
1.1	2026-08-12	Dayanandini Pathmanathan	Addressing lab comments.
1.2	2026-10-08	Dayanandini Pathmanathan	Addressing ACA comments.

©2026 RackTop Systems

1. Introduction

This section identifies the Security Target (ST), Target of Evaluation (TOE), and the ST organization. The Target of Evaluation (TOE) is the RackTop SPEAR v1.0 full disk encryption solution and will hereafter be referred to as the TOE throughout this document.

1.1 Purpose

This ST is divided into ten sections, as follows:

- Introduction (Section 1) – Provides a brief summary of the ST contents and describes the organization of other sections within this document. It also provides an overview of the TOE security functionality and describes the physical and logical scope for the TOE as well as the ST and TOE references.
- Conformance Claims (Section 2) – Provides the identification of any Common Criteria (CC), Protection Profile (PP), and Evaluation Assurance Level (EAL) package claims. It also identifies whether the ST contains extended security requirements.
- Security Problem (Section 3) – Describes the threats, Organizational Security Policies (OSPs), and assumptions that pertain to the TOE and its environment.
- Security Objectives (Section 4) – Identifies the security objectives that are satisfied by the TOE and its environment.
- Extended Components (Section 5) – Identifies new components (extended Security Functional Requirements (SFRs) and extended Security Assurance Requirements (SARs)) that are not included in CC Part 2 or CC Part 3.
- Security Assurance Requirements (Section 6) – Presents the SARs met by the TOE.
- Security Functional Requirements (Section 7) – Presents the SFRs met by the TOE.
- TOE Summary Specification (Section 8) – Describes the security functions provided by the TOE that satisfy the SFRs and objectives.
- Rationale (Section 9) – Presents the rationale for the SFR dependencies as to their consistency, completeness, and suitability.
- Acronyms and Terms (Section 10) – Defines the acronyms and terminology used within this ST.

1.2 Security Target and TOE References

Table 1 shows the ST and TOE references.

Table 1 – ST and TOE References

ST Title	RackTop Systems, RackTop SPEAR v1.0 Security Target
ST Version	Version 1.2
ST Author	Corsec Security, Inc.
ST Publication Date	October 8, 2026
TOE Reference	RackTop SPEAR v1.0

©2026 RackTop Systems

1.3 Product Overview

The Product Overview provides a high-level description of the RackTop SPEAR v1.0. The following section, TOE Overview, will provide the introduction to the parts of the overall product offering that are specifically being evaluated.

1.4 TOE Overview

The TOE Overview summarizes the usage and major security features of the TOE. The TOE Overview provides a context for the TOE evaluation by identifying the TOE type, describing the product, and defining the specific evaluated configuration.

The TOE is RackTop SPEAR v1.0, a Full Drive Encryption (FDE) solution including both Authorization Acquisition (AA) and Encryption Engine (EE) components. The TOE is built on the Linux Unified Key Setup (LUKS) version 2 specification and leverages RHEL version v9.4.

It may be deployed as a component of a multilayer encryption architecture designed to meet the requirements of the Commercial Solutions for Classified (CSfC) Data-at-Rest (DAR) Capability Package. The TOE provides a software sector-level encryption of a block device by means of the Linux kernel `dm-crypt` driver, with key hierarchy and protection mechanisms as defined by the LUKS2 specification, complementing hardware-based encryption solutions provided by self-encrypting drives (SEDs). In such a complementary dual-layer configuration, all user-accessible data within a configured locking range is encrypted independently by both the SED and the TOE. This layered approach ensures that the compromise or failure of encryption in one layer does not result in the exposure of sensitive information. The TOE implements both the Authorization Acquisition (AA) function and the Encryption Engine (EE**)** function, as defined in the Full Disk Encryption collaborative Protection Profiles. The bootloader integration with the AA component ensures that the operating system cannot start until valid credentials are successfully presented and verified. When deployed in its intended configuration, the TOE protects both the operating system drive and all data drives, requiring successful unlocking of each protected volume before the OS completes its boot process and any protected data becomes accessible.

1.5 TOE Environment

The TOE relies on the following components of the RHEL OS 9.4:

- Kernel cryptography
- Kernel device mapper
- User space cryptography

It is assumed that there will be no untrusted users or software on the TOE Server components. In addition, the TOE is intended to be deployed in a physically secured cabinet, room, or data center with the appropriate level of physical access control and physical protection (e.g. badge access, fire control, locks, alarms, etc.).

©2026 RackTop Systems

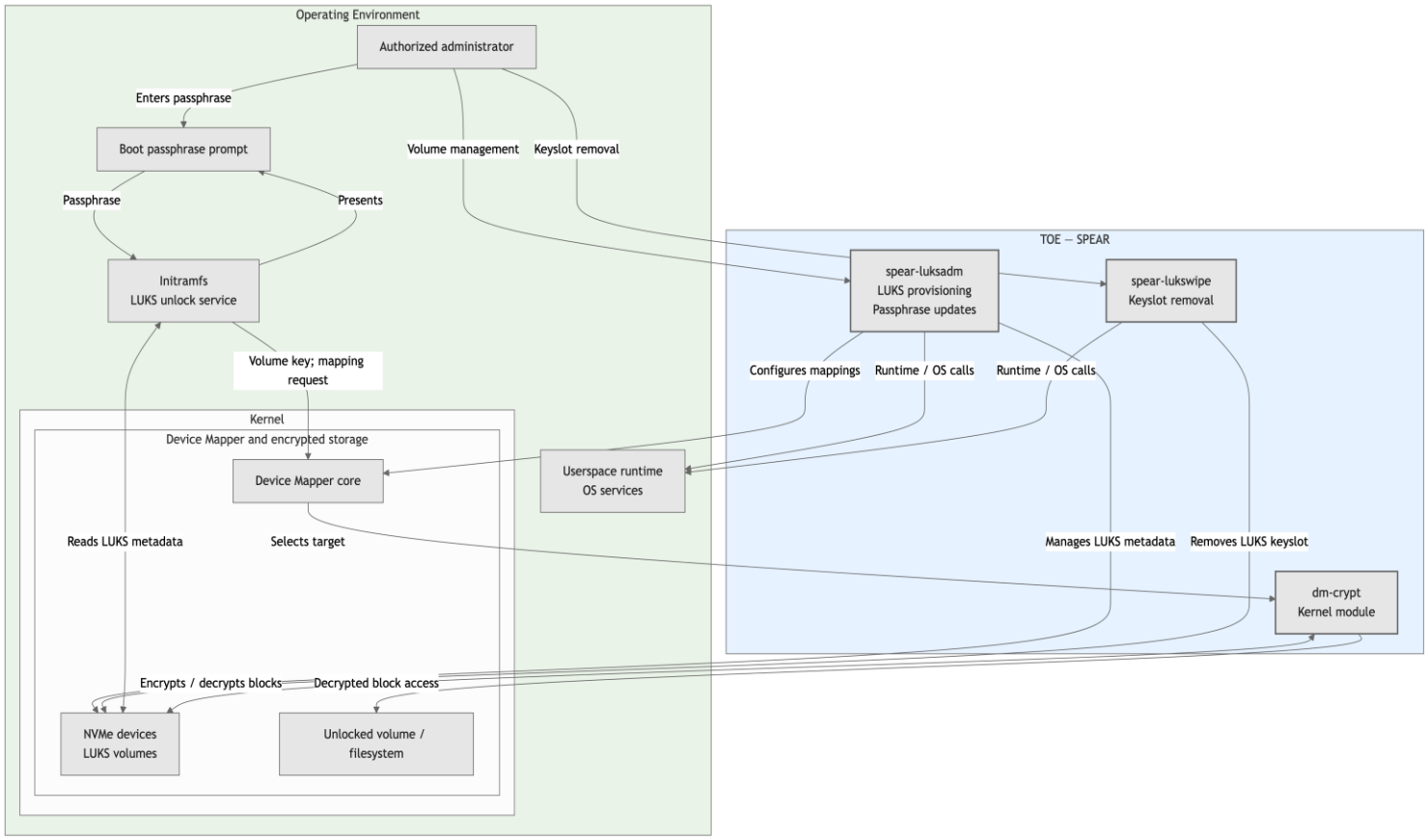
1.6 TOE Description

This section primarily addresses the physical and logical components of the TOE that are included in the evaluation.

1.6.1 Physical Scope

The TOE has been evaluated using the following hardware configuration details below:
Processor: Intel(R) Xeon(R) D-2796NT CPU @ 2.00GHz
Motherboard: Supermicro X12SDV-20C-SPT8F
Operating system: Red Hat Enterprise Linux (RHEL) v9.4
Crypto module: OpenSSL 3.x with FIPS provider 3.0.7

Figure 1 illustrates the physical scope and the physical boundary of the overall solution and ties together all of the components of the TOE. The TOE Boundary includes all the RackTop-developed parts of the TOE. The TOE is a Full Drive Encryption (FDE) solution including both Authorization Acquisition (AA) and Encryption Engine (EE) components.



Legend

TOE Boundary

Figure 1 – TOE Boundary

©2026 RackTop Systems

1.6.1.1 Guidance Documentation

Table 2 lists the TOE Guidance Documentation needed to install, configure, and maintain the TOE.

Table 2 – Guidance Documentation

Document Name	Description
RackTop SPEAR Guidance Document Supplement v0.9	Guidance Supplement Document.

1.6.2 TOE Delivery

Customers will be able to download the TOE ISO from the following website:
https://www.racktopsystems.com/spear

TOE ISO filename: rhel-9-CSFC-20260626-d01c26e7.iso
SHA-256 hash value: 5ba29fb776be50bfe9d94827afef53d5264f832d20c30d1ee9f3c98667be3af6

The TOE installation is performed remotely through a web browser via a management user interface exposed through the Baseboard Management Controller (BMC).

BMC Usage in the TOE evaluated configuration
The BMC is used solely to facilitate initial provisioning and installation of the operational environment. Following completion of operating system installation and system initialization, the BMC shall not remain network accessible

The TOE RPMs are listed below:

• spear-utils-1.0.6-33
• updatefde-1.0.2-44

Customers will be able to download the TOE documentation from the following website:
https://www.myracktop.com

Users would need to reach out to RackTop to have an account provisioned for them to access the website.

1.6.3 Logical Scope

The logical boundary of the TOE will be broken down into the following security classes, which are further described in sections 7 and 8 of this ST.

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

©2026 RackTop Systems

1.6.3.1 Cryptographic Support

The TOE relies on CAVP tested cryptographic algorithms as stated in Table 9. The TOE supports validation of Border Encryption Value, key derivation, key chaining and key destruction.

1.6.3.2 User Data Protection

The TOE implements full data drive encryption as per FCS_COP.1(f) such that the drive contains no plaintext sensitive data.

1.6.3.3 Security Management

The TOE supports two Compliant power-saving states, G2 (S5) – Soft Off and G3 – Mechanical Off and switching into the states are restricted to authorized users. The only role supported is an authorized user.

The following management functions are supported:

- forwarding requests to change the DEK to the EE,
- forwarding requests to cryptographically erase the DEK to the EE,
- allowing authorized users to change authorization factors or set of authorization factors used,
- initiate TOE firmware/software updates,
- change the DEK, as specified in FCS_CKM.1, when re-provisioning or when commanded,
- erase the DEK, as specified in FCS_CKM.4(a),
- initiate TOE firmware/software updates

1.6.3.4 Protection of the TSF

The TOE protects key material and supports compliant power states. Authorized users can perform software updates. Authenticity of software updates is ensured through the RSA digital signature mechanism.

1.6.4 Product Physical/Logical Features and Functionality not included in the TOE

There are no excluded features and functionality that are not part of the evaluated configuration of the TOE.

©2026 RackTop Systems

2. Conformance Claims

This section provides the identification for any CC, PP, and EAL package conformance claims. Rationale is provided for any extensions or augmentations to the conformance claims. Rationale for CC and PP conformance claims can be found in section 9.1.

Table 3 – CC and PP Conformance

CC Identification and Conformance	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, April 2017; CC Part 2 extended; CC Part 3 conformant and no interpretations apply to the claims made in this ST.
PP Identification	Exact Conformance[1] to the *collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition, Version 2.0 + Errata 20190201, February 1, 2019 and collaborative Protection Profile for Full Drive Encryption – Encryption Engine, Version 2.0 + Errata 20190201, February 1, 2019 (FDE_EE).*

Table 4 – NIAP FDE AA and FDE EE Technical Decisions

Technical Decisions	cPP	Applicable (Y/N)	Exclusion Rationale (if applicable)
TD0458– FIT Technical Decision for FPT_KYP_EXT.1 evaluation activities	FDE_AA and FDE_EE	Yes	N/A
TD0460: FIT Technical Decision for FPT PWR EXT.1 non-compliant power saving states	FDE_EE	Yes	N/A
TD0464: FIT Technical Decision for FPT PWR EXT.1 compliant power savings states	FDE_EE	Yes	N/A
TD0606 – FIT Technical Recommendation for Evaluating a NAS against the FDE AA and FDEE	FDE_AA and FDE_EE	Yes	N/A
TD0759– FIT Technical Decision for FCS_AFA_EXT.1.1	FDE_AA	Yes	N/A
TD0760– FIT Technical Decision for FCS_SNI_EXT.1.3, FCS_COP.1(f)	FDE_AA	Yes	N/A
TD0765- FIT Technical Decision for FMT_MOF.1	FDE_AA	Yes	N/A
TD0766 - FIT Technical Decision for FCS_CKM.4(d) Test Notes	FDE_AA and FDE_EE	Yes	N/A
TD0767- FIT Technical Decision for FMT_SMF.1.1	FDE_AA	Yes	N/A
TD0769 - FIT Technical Decision for FPT_KYP_EXT.1.1	FDE_AA and FDE_EE	Yes	N/A
TD0929 - FIT Technical Decision: Clarification to FCS_PCC_EXT.1.1	FDE_AA	Yes	N/A
TD1042 - FIT Technical Decision: Replacing FIPS 186-4 with 186-5	FDE_AA and FDE_EE	Yes	N/A
TD1046 - FIT Technical Decision: Addition of AES-XTS in FCS_COP.1(g)	FDE_AA and FDE_EE	No	AES XTS is not used for key encryption and decryption.

[1] Exact Conformance is defined as the ST contains all of the requirements in Section 5, Appendix A and B of the cPPs.

©2026 RackTop Systems

3. Security Problem Definition

This section describes Security Problem Definition derived from the collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition, Version 2.0 + Errata 20190201, February 1, 2019 (FDE AA) and collaborative Protection Profile for Full Drive Encryption – Encryption Engine, Version 2.0 + Errata 20190201, February 1, 2019 (FDE_EE). The security problem consists of threats that the TOE must mitigate, Assumptions on the operational environment and security objectives for the Operational Environment.

3.1 Threats

This section details all threats as highlighted in the cPP. A threat consists of a threat agent, an asset and an adverse action of that threat agent on that asset. The threat agents are the entities that put the assets at risk if an adversary obtains a lost or stolen storage device. Threats drive the functional requirements for the target of evaluation (TOE).

T.UNAUTHORIZED _DATA_ACCESS
The cPP addresses the primary threat of unauthorized disclosure of protected data stored on a storage device. If an adversary obtains a lost or stolen storage device (e.g., a storage device contained in a laptop or a portable external storage device), they may attempt to connect a targeted storage device to a host of which they have complete control and have raw access to the storage device (e.g., to specified disk sectors, to specified blocks).

T.KEYING_MATERIAL_COMPROMISE/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 peripherals in the operating environment (OE), e.g. BIOS configuration, SPI flash, or TPMs.

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.

©2026 RackTop Systems

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

T.UNAUTHORIZED_UPDATE/AA
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.

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

T.UNAUTHORIZED_FIRMWARE_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.

Assumptions
This section details all assumptions that must remain true in order to mitigate the threats.

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.

©2026 RackTop Systems

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

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.TRUSTED_CHANNEL
Communication among and between product components (e.g., AA and EE) is sufficiently protected to prevent information disclosure. In cases in which a single product fulfils both cPPs, then the communication between the components does not extend beyond the boundary of the TOE (e.g., communication path is within the TOE boundary). In cases in which independent products satisfy the requirements of the AA and EE, the physically close proximity of the two products during their operation means that the threat agent has very little opportunity to interpose itself in the channel between the two without the user noticing and taking appropriate actions.

A.TRAINED_USER/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.PLATFORM_STATE
The platform in which the storage device resides (or an external storage device is connected) is free of malware that could interfere with the correct operation of the product.

A.SINGLE_USE_ET/AA
External tokens that contain authorization factors are used for no other purpose than to store the external token authorization factors.

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 the device is in a Compliant power saving state or has fully powered off. This properly clears memories and locks down the device. Authorized users

©2026 RackTop Systems

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.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.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.STRONG_CRYPTO
All cryptography implemented in the Operational Environment and used by the product meets the requirements listed in the cPP. This includes generation of external token authorization factors by a RBG.

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

3.2 Organizational Security Policy

There are no organizational security policies addressed by FDE AA cPP and FDE EE cPP.

©2026 RackTop Systems

4. Security Objectives

4.1 Security Objectives for the Operational Environment

The following subsections describe objectives for the Operational Environment.

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

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

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

OE.PASSPHRASE_STRENGTH
An authorized administrator will be responsible for ensuring that the passphrase authorization factor conforms to guidance from the Enterprise using the TOE.

OE.POWER_DOWN/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.

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

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.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.

©2026 RackTop Systems

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

©2026 RackTop Systems

5. Extended Components

Extended components defined in this ST have been drawn directly from FDE_AA cPP and FDE_EE cPP. Refer to FDE_AA cPP and FDE_EE cPP for further details.

©2026 RackTop Systems

6. Security Assurance Requirements

The cPP identifies the SARs to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of SARs from CC3.1 Revision 5 that are required in evaluations against the FDE AA and FDE EE cPP. Individual Evaluation Activities to be performed are specified in the Supporting Document (Mandatory Technical Document) Full Drive Encryption: Authorization Acquisition, February 2019 (FDE_AA) and Encryption Engine, February 2019 (FDE_EE).

The general model for evaluation of TOEs against STs written to conform to the FDE AA and FDE EE cPP is as follows: after the ST has been approved for evaluation, the ITSEF[2] will obtain the TOE, supporting environmental IT (if required), and the guidance documentation for the TOE. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The ITSEF also performs the Evaluation Activities contained within the SD, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The Evaluation Activities that are captured in the SD also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the FDE AA and FDE EE cPP.

The TOE security assurance requirements are identified in Table 5.

Table 5 – Security Assurance Requirements

Assurance Requirements	
Security Target (ASE)	Conformance claims (ASE_CCL.1)
	Extended components definition (ASE_ECD.1)
	ST introduction (ASE_INT.1)
	Security objectives for the operational environment (ASE_OBJ.1)
	Stated security requirements (ASE_REQ.1)
	Security Problem Definition (ASE_SPD.1)
	TOE summary specification (ASE_TSS.1)
Development (ADV)	Basic functional specification (ADV_FSP.1)
Guidance documents (AGD)	Operational user guidance (AGD_OPE.1)
	Preparative procedures (AGD_PRE.1)
Life Cycle Support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM[3] coverage (ALC_CMS.1)
Tests (ATE)	Independent testing – sample (ATE_IND.1)
Vulnerability assessment (AVA)	Vulnerability survey (AVA_VAN.1)

[2] ITSEF – Information Technology Security Evaluation Facility
[3] CM – Configuration Management

©2026 RackTop Systems

7. Security Functional Requirements

The individual security functional requirements are specified in the following sections. SFRs in this section are mandatory SFRs that any conformant TOE must meet. Based on selections made in these SFRs, it will also be necessary to include some of the selection-based SFRs in Appendix B. Additional optional SFRs may also be adopted from those listed in Appendix A.

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

7.1 Conventions

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

- Assignment: Indicated with italicized text surrounded by brackets (e.g., [*assignment*]).
- Refinement made in the PP: Indicated with bold text and strikethroughs (e.g., "**refinement**" or "~~refinement~~").
- Selection: Indicated with underlined text surrounded by brackets (e.g., [selection]).
- Assignment within a selection: Indicated with italicized and underlined text surrounded by brackets (e.g., [*assignment within a selection*]);

7.2 Security Functional Requirements

This section specifies the SFRs for the TOE and organizes the SFRs by CC class. Table 6 identifies all SFRs implemented by the TOE and indicates the ST operations made by the ST author performed on each requirement. Refinements made in the PP are also indicated. The SFRs are indicated with AA/EE depending on applicability to the cPP. Each SFR has either AA/EE or AA&EE which states which cPP it is derived from.

Table 6 – TOE Security Functional Requirements

Name	Description	S	A	R
FCS_AFA_EXT.1/AA	Authorization Factor Acquisition	✓		
FCS_AFA_EXT.2/AA	Timing of Authorization Factor Acquisition			
FCS_CKM.1(c)/EE	Cryptographic Key Generation (Data Encryption Key)	✓		✓
FCS_CKM.4(a)/AA	Cryptographic Key Destruction (Power Management)	✓	✓	✓
FCS_CKM.4(a)/EE	Cryptographic Key Destruction (Power Management)	✓	✓	✓
FCS_CKM.4(d)/AA	Cryptographic Key Destruction (Software TOE, 3rd Party Storage)	✓	✓	✓
FCS_CKM.4(d)/EE	Cryptographic Key Destruction (Software TOE, 3rd Party Storage)	✓	✓	✓
FCS_CKM_EXT.4(a)/AA	Cryptographic Key and Key Material Destruction (Destruction Timing)			
FCS_CKM_EXT.4(a)/EE	Cryptographic Key and Key Material Destruction (Destruction Timing)			
FCS_CKM_EXT.4(b)/AA	Cryptographic Key and Key Material Destruction (Power Management)	✓		✓
FCS_CKM_EXT.4(b)/EE	Cryptographic Key and Key Material Destruction (Power Management)			
FCS_CKM_EXT.6/EE	Cryptographic Key Destruction Types	✓		
FCS_COP.1(a)/AA&EE	Cryptographic Operation (Signature Verification)	✓		✓

©2026 RackTop Systems

Name	Description	S	A	R
FCS_COP.1(b)/AA&EE	Cryptographic Operation (Hash Algorithm)	✓		✓
FCS_COP.1(c)/AA	Cryptographic Operation (Keyed Hash Algorithm)	✓		✓
FCS_COP.1(c)/EE	Cryptographic Operation (Message Authentication)	✓	✓	✓
FCS_COP.1(f)/EE	Cryptographic Operation (AES Data Encryption/Decryption)	✓		✓
FCS_COP.1(g)/EE	Cryptographic Operation (Key Encryption)	✓		✓
FCS_KYC_EXT.1/AA	Key Chaining (Initiator)	✓		
FCS_KYC_EXT.2/EE	Key Chaining (Recipient)	✓		
FCS_PCC_EXT.1/AA	Cryptographic Password Construct and Conditioning	✓	✓	
FCS_SNI_EXT.1/AA	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)	✓		
FCS_SNI_EXT.1/EE	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)	✓		
FCS_VAL_EXT.1/EE	Validation	✓	✓	
FCS_KDF_EXT.1/EE	Cryptographic Key Derivation	✓	✓	
FDP_DSK_EXT.1/EE	Protection of Data on Disk			
FMT_MOF.1/AA	Management of Functions Behavior	✓	✓	
FMT_SMF.1/AA	Specification of Management Functions	✓		✓
FMT_SMF.1/EE	Specification of Management Functions	✓		✓
FMT_SMR.1/AA	Security Roles		✓	
FPT_KYP_EXT.1/AA	Protection of Key and Key Material	✓		
FPT_KYP_EXT.1/EE	Protection of Key and Key Material	✓		
FPT_PWR_EXT.1/AA&EE	Power Saving States	✓		
FPT_PWR_EXT.2/AA	Timing of Power Saving States	✓		
FPT_PWR_EXT.2/EE	Timing of Power Saving States	✓		
FPT_TST_EXT.1/EE	TSF Testing	✓	✓	
FPT_TUD_EXT.1/AA	Trusted Update	✓		✓
FPT_TUD_EXT.1/EE	Trusted Update	✓		✓

Note: S=Selection; A=Assignment; R=Refinement

7.2.1 Class FCS: Cryptographic Support

FCS_AFA_EXT.1/AA Authorization Factor Acquisition

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].

TD0759 applied but does not modify the SFR.

FCS_AFA_EXT.2/AA Timing of Authorization Factor Acquisition

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.

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

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

- **accept a DEK that is generated by the RBG provided by the host platform,]**

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

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

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

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

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

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

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

- ***For volatile memory, the destruction shall be executed by a [***
    - ***single overwrite consisting of [***
        - ***zeroes,]***
    - ***removal of power to the memory,];***

]

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

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

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

- ***For volatile memory, the destruction shall be executed by a [***
    - ***single overwrite consisting of [***
        - ***zeroes,]***
    - ***removal of power to the memory,];***

- ***For non-volatile storage that consists of the invocation of an interface provided by the underlying platform that [***
    - ***logically addresses the storage location of the key and performs a [***

single,] overwrite consisting of

- ***[a pseudo-random pattern utilizing the underlying platform's RBG];]***

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

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

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

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

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

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

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**.

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

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.

FCS_CKM_EXT.6/EE Cryptographic Key Destruction Types

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

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

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

- ***RSA Digital Signature Algorithm with a key size (modulus) of [3072-bit, 4096-bit];***

]
that meet the following: [

- ***FIPS PUB 186-5, "Digital Signature Standard (DSS)", Section 5, using PKCS #1 v2.2 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***

].

TD1042 has been applied.

©2026 RackTop Systems

FCS_COP.1(b)/AA&EE Cryptographic Operation (Hash Algorithm)

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

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

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-512***] and cryptographic key sizes *[**512 bits**]* that meet the following: *[ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"*].

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

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

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

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]***].

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

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]***].

FCS_KYC_EXT.1/AA Key Chaining (Initiator)

FCS_KYC_EXT.1.1/AA The TSF shall maintain a key chain of: [
- one, using a submask as the BEV;

] .

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 [*the EE*] [
- without validation taking place].

FCS_KYC_EXT.2/EE Key Chaining (Recipient)

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

©2026 RackTop Systems

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

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

FCS_PCC_EXT.1/AA Cryptographic Password Construct and Conditioning

FCS_PCC_EXT.1.1 A password used by the TSF to generate a password authorization factor shall enable at least [*64 or more*] characters in the set of {upper case characters, lower case characters, numbers, and [*the following characters!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~]}* and shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC-[SHA-512], with [*1000000*] iterations and output cryptographic key sizes [256 bits] that meet the following: [NIST SP 800-132].

TD0929 applied.

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

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

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

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

TD 0760 applied.

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

FCS_SNI_EXT.1.1/EE The TSF shall use [use no salts].

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;]

FCS_VAL_EXT.1/EE Validation

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(b), FCS_COP.1(c)] and compare it to a stored hashed [**value**]; **]**

©2026 RackTop Systems

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 [*1 or 3*] of consecutive failed validation attempts].

FCS_KDF_EXT.1/EE Cryptographic Key Derivation

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

•NIST SP 800-132], using the keyed-hash functions specified in FCS_COP.1(c), such that the output is at least of equivalent security strength (in number of bits) to the BEV.

7.2.2 Class FDP: User Data Protection

FDP_DSK_EXT.1/EE Protection of Data on Disk

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

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

7.2.3 Class FMT: Security Management

FMT_MOF.1/AA Management of Functions Behavior

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*].

FMT_SMF.1/AA Specification of Management Functions

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]***].

TD0767 is applied.

FMT_SMF.1/EE Specification of Management Functions

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),
c) initiate TOE firmware/software updates,
*d) **[no other functions]**].*

FMT_SMR.1/AA Security Roles

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.

7.2.4 Class FPT: Protection of the TSF

FPT_KYP_EXT.1/AA Protection of Key and Key Material

FPT_KYP_EXT.1.1/AA The TSF shall [

- not store keys in non-volatile memory].

TD0769 applied but does not modify the SFR.

FPT_KYP_EXT.1/EE Protection of Key and Key Material

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) or FCS_COP.1(e)].

TD0769 applied but does not modify the SFR.

FPT_PWR_EXT.1/AA&EE Power Saving States

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

TD0464 applied.

FPT_PWR_EXT.2/AA Timing of Power Saving States

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

FPT_PWR_EXT.2/EE Timing of Power Saving States

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

FPT_TST_EXT.1/EE TSF Testing

FPT_TST_EXT.1.1/EE 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*].

©2026 RackTop Systems

FPT_TUD_EXT.1/AA Trusted Update

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

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

FPT_TUD_EXT.1.3/AA 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.

FPT_TUD_EXT.1/EE Trusted Update

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

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

FPT_TUD_EXT.1.3/EE 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.

©2026 RackTop Systems

8. TOE Summary Specification

This section presents information to detail how the TOE meets the functional requirements described in previous sections of this ST.

8.1 TOE Security Functionality

Each of the security requirements and the associated descriptions correspond to the security functions. Hence, each function is described by how it specifically satisfies each of its related requirements. This serves to both describe the security functions and rationalize that the security functions satisfy the necessary requirements.

Table 7 – Mapping of TOE Security Functionality to Security Functional Requirements

TOE Security Function	Name	Description
Cryptographic Support	FCS_AFA_EXT.1/AA	Authorization Factor Acquisition
	FCS_AFA_EXT.2/AA	Timing of Authorization Factor Acquisition
	FCS_CKM.1(c)/EE	Cryptographic Key Generation (Data Encryption Key)
	FCS_CKM.4(a)/AA	Cryptographic Key Destruction (Power Management)
	FCS_CKM.4(a)/EE	Cryptographic Key Destruction (Power Management)
	FCS_CKM.4(d)/AA	Cryptographic Key Destruction (Software TOE, 3rd Party Storage)
	FCS_CKM.4(d)/EE	Cryptographic Key Destruction (Software TOE, 3rd Party Storage)
	FCS_CKM_EXT.4(a)/AA	Cryptographic Key and Key Material Destruction (Destruction Timing)
	FCS_CKM_EXT.4(a)EE	Cryptographic Key and Key Material Destruction (Destruction Timing)
	FCS_CKM_EXT.4(b)/AA	Cryptographic Key and Key Material Destruction (Power Management)
	FCS_CKM_EXT.4(b)/EE	Cryptographic Key and Key Material Destruction (Power Management)
	FCS_CKM_EXT.6/EE	Cryptographic Key Destruction Types
	FCS_COP.1(a)/AA&EE	Cryptographic Operation (Signature Verification)
	FCS_COP.1(b)AA&EE	Cryptographic Operation (Hash Algorithm)
	FCS_COP.1(c)/AA	Cryptographic Operation (Keyed Hash Algorithm)
	FCS_COP.1(c)/EE	Cryptographic Operation (Message Authentication)
	FCS_COP.1(f)/EE	Cryptographic Operation (AES Data Encryption/Decryption)
	FCS_COP.1(g)/EE	Cryptographic Operation (Key Encryption)
	FCS_KYC_EXT.1/AA	Key Chaining (Initiator)
	FCS_KYC_EXT.2/EE	Key Chaining (Recipient)
	FCS_PCC_EXT.1/AA	Cryptographic Password Construct and Conditioning
	FCS_SNI_EXT.1/AA	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)

©2026 RackTop Systems

TOE Security Function	Name	Description
	FCS_SNI_EXT.1/EE	Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
	FCS_VAL_EXT.1/EE	Validation
	FCS_KDF_EXT.1/EE	Cryptographic Key Derivation
User Data Protection	FDP_DSK_EXT.1/EE	Protection of Data on Disk
Security Management	FMT_MOF.1/AA	Management of Functions Behavior
	FMT_SMF.1/AA	Specification of Management Functions
	FMT_SMF.1/EE	Specification of Management Functions
	FMT_SMR.1/AA	Security Roles
Protection of the TSF	FPT_KYP_EXT.1/AA	Protection of Key and Key Material
	FPT_KYP_EXT.1/EE	Protection of Key and Key Material
	FPT_PWR_EXT.1/AA&EE	Power Saving States
	FPT_PWR_EXT.2/AA	Timing of Power Saving States
	FPT_PWR_EXT.2/EE	Timing of Power Saving States
	FPT_TST_EXT.1/EE	TSF Testing
	FPT_TUD_EXT.1/AA	Trusted Update
	FPT_TUD_EXT.1/EE	Trusted Update

8.1.1 Cryptographic Support

Table 8 - CAVP Algorithm Certificate References

Algorithm	Description/Operation	Supported Mode/Standard	Key Sizes	CAVP Cert. #	SFR
RSA	Signature Verification	FIPS PUB 186-5	3072 bits, 4096 bits	A8190	FCS_COP.1(a)/AA&EE
AES	Data Encryption and Decryption	XTS IEEE 1619	256 bits	A8191	FCS_COP.1(f)/EE
AES	Key Encryption Volume Master Key Encryption	CBC ISO/IEC 10116	256 bits	A8190	FCS_COP.1(g)/EE
SHA	Hashing Services	ISO/IEC 10118-3:2004	256, 512 bits	A8190 A8191	FCS_COP.1(b)/AA&EE
HMAC	Keyed hashing services	ISO/IEC 9797-2:2011	512 bits	A8190 A8191	FCS_COP.1(c)/AA FCS_COP.1(c)/EE

NOTE: Please refer to the KMD (Key Management Description) for SPEAR v1.0 (August 5, 2026) for the description of the Key Management details.

©2026 RackTop Systems

FCS_AFA_EXT.1
The TOE enforces a single authorization factor: passphrases. A passphrase consists of alphanumeric and special characters and is required to provide sufficient entropy such that the cryptographic keys derived from it are resistant to brute-force and dictionary attacks. The TOE is capable of randomly generating strong passphrases during initial drive encryption enablement and during subsequent passphrase change operations. An authorized user may alternatively configure a passphrase of their choosing. Absolute minimum acceptable passphrase length is 8 characters. The TOE supports passphrases of at least 15 characters in length and longer. As a best practice, passphrases should be no less than 15 characters and much longer passphrases are supported.

In order to initiate the Operating System boot process, an authorized user is required to successfully authenticate by unlocking the system volume and all associated data volumes. Unlocking is performed at the system console through interactive entry of the passphrase via the keyboard. This authentication step occurs upon each system reboot, following entry into a compliant power-saving state, and whenever the contents of volatile memory are cleared. The derived cryptographic keys necessary for volume access are retained only in random-access memory (RAM) after successful authentication and are not persistent across power cycles or memory-clearing events.

TOE Security Functional Requirements Satisfied: FCS_AFA_EXT.1

FCS_AFA_EXT.2
The TOE supports two power states: powered on and powered off. Following initial configuration, during which the LUKS volumes on the relevant NVMe storage devices are provisioned in a compliant manner, all encrypted volumes transition to the locked state whenever the system is powered off. Upon the next power-on, the encrypted volumes remain locked and inaccessible until successfully unlocked using the configured passphrase authorization factor. In this state, the system will not proceed with the Operating System boot process until an authorized user correctly enters the passphrase and all protected volumes are unlocked. When authentication is successful, the TOE derives the necessary cryptographic keys from the passphrase and retains them exclusively in volatile memory (RAM) for use during operation. These keys are never stored in persistent storage and are cleared upon system power-off, system reboot, or entry into a compliant power-saving state, thereby ensuring that protected data cannot be accessed without re-authentication.

TOE Security Functional Requirements Satisfied: FCS_AFA_EXT.2

FCS_CKM.1(c)/EE
The TOE relies on the operating environment to generate the Data Encryption Key (DEK). The DEK is the Volume Master Key. The LUKS subsystem defines an interface, `crypt_backend_rng` , which abstracts the underlying random number generator and provides discrete implementations for supported backends. For the evaluated configuration, the TOE uses the OpenSSL backend, which invokes the `RAND_bytes` API from OpenSSL to obtain cryptographically strong random data. The quantity of random data requested depends on the key size. For `luksFormat` , the TOE requests 512 bytes of random data, which is sufficient to produce two independent 256-bit keys required for operation in AES-XTS mode. The highest quality level of random number generation is reserved exclusively for volume master keys, ensuring that the DEK is generated with appropriate entropy and strength.

TOE Security Functional Requirements Satisfied: FCS_CKM.1(c)/EE

FCS_CKM.4(a)/AA and FCS_CKM.4(a)/EE
The TOE ensures that all memory locations containing cryptographic key material are securely cleared as soon as the key is no longer required.

 ©2026 RackTop Systems

Volume Master Key Destruction:
  - When the volume is closed
  - When the system is shut down, the VMK is explicitly zeroized in kernel memory using `memzero_explicit()` or equivalent secure erasure function.
  - When the destruction of VMK is explicitly requested by the authorized user. This effectively destroys the data and securely wipes blocks where keys are stored on the block device. The stored key is encrypted.
- No residual VMK data remains in user space.

Key Encryption Key Destruction:
- KEK values are overwritten in memory immediately after decrypting the VMK.
- Implemented via explicit buffer zeroization functions.

TOE Security Functional Requirements Satisfied: FCS_CKM.4(a)/AA and FCS_CKM.4(a)/EE

FCS_CKM.4(d)/AA
The TOE's working memory consists of all volatile Random Access Memory (RAM) provided by the host platform. Cryptographic keys are never left abandoned in memory; instead, any key material that is no longer required is immediately and explicitly zeroized. Keys in volatile memory are destroyed by the removal of power and explicit memory zeroization. Keys are not stored in non-volatile memory.

Please refer to Section 3 of the KMD for details on key destruction.

TOE Security Functional Requirements Satisfied: FCS_CKM.4(d)/AA

FCS_CKM.4(d)/EE
Keys in volatile memory are destroyed by the removal of power and explicit memory zeroization. Keys in non-volatile memory are overwritten with random digits leveraging the underlying platforms' RBG.

Please refer to Section 3 of the KMD for details on key destruction.

TOE Security Functional Requirements Satisfied: FCS_CKM.4(d)/EE

FCS_CKM_EXT.4(a)/AA and FCS_CKM_EXT.4(a)/EE

The TOE destroys all keys and key material when no longer needed.

This lifecycle ensures that no key material remains in memory beyond the specific, minimal periods in which it is operationally required.

TOE Security Functional Requirements Satisfied: FCS_CKM_EXT.4(a)/AA and FCS_CKM_EXT.4(a)/EE

FCS_CKM_EXT.4(b)/AA, FCS_CKM_EXT.4(b)/EE
When the system transitions to a shutdown state, power is removed from system memory and all key material in volatile memory is cleared out.

TOE Security Functional Requirements Satisfied: FCS_CKM_EXT.4(b)/AA and FCS_CKM_EXT.4(b)/EE

FCS_CKM_EXT.6/EE

The TOE implements key destruction as defined in FCS_CKM.4(d).

TOE Security Functional Requirements Satisfied: FCS_CKM_EXT.6/EE

FCS_COP.1(a)/AA, FCS_COP.1/EE
RSA algorithm is used for signature verification and supports key sizes of 3072 bits and 4096 bits conforming to FIPS PUB 186-5, "Digital Signature Standard (DSS)", Section 5, using PKCS #1 v2.2 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. The RSA algorithm is used for signing trusted updates.

The TOE software update mechanism uses RSA cryptographic signing to ensure the authenticity and integrity of all packages in an update bundle before any installation occurs. Packages are installed only after all validation checks have successfully passed, confirming their authenticity and trusted source. Software updates are delivered as a bundle of platform-specific RPM packages (which are themselves `cpio` archives), combined into a single TAR archive. A `cpio` archive is a file format used to store a collection of files and directories. The public key used for signature verification is provided separately, and may be either a 3072-bit or 4096-bit RSA key, potentially unique per customer. The TAR archive may reside anywhere on the filesystem, though the recommended location is `/var/tmp/updates`. The authorized user is responsible for securely transferring the update to the system, for example via SSH.

Table 9 - Files required for validation of the software packages

Files	Descriptions
Metadata	Acts as a manifest and includes the name of each package in the bundle as well as the package's SHA-512 checksum.
Signature	Signature file contains the signature of the metadata file included in the bundle. The signature file contains RSASSA-PKCS1-v1_5 formatted binary payload.
RPM	Update bundle package content

The validation process is as follows:

1. Unpack the TAR archive and read the metadata and signature files into memory.
2. Load the separately provided public key into memory. If the key is missing or invalid, the update process is terminated with an error.
3. Verify the signature of the metadata file using the RSA public key. If verification fails, the update process is terminated with a signature validation error.
4. Parse the metadata file and create a list of RPM packages and their recorded SHA-512 checksums.
5. Compare the manifest contents against the provided RPM files. If any package is missing or extra, the update process is terminated with a validation error.
6. For each RPM file, calculate its SHA-512 digest and compare it with the corresponding checksum in the metadata file. Any mismatch results in termination with a validation error.
7. After all RPMs are successfully validated, they are installed using the `dnf` system command.

TOE Security Functional Requirements Satisfied: FCS_COP.1(a)/AA and & FCS_COP.1(a)/EE

FCS_COP.1(b)/AA&EE

The TOE supports SHA-256 and SHA-512 hash algorithms are used to perform hashing operation conforming to ISO/IEC 10118-3:2004.

The association of SHA-512 with cryptographic functions is as follows:

- Key Derivation (PBKDF2):
  - The TOE derives cryptographic keys using the PBKDF2 construction with HMAC-SHA-512. This process is applied both for generating Key Encryption Keys (KEKs) from authorized user passphrases and for deriving candidate Data Encryption Keys (DEKs).

- Border Encryption Value (BEV) Validation:
  - For each keyslot, the TOE computes a PBKDF2 of the DEK. This digest is used to validate the Border Encryption Value, ensuring the integrity and correctness of key material.

The association of SHA-512 with other TSF cryptographic functions is as follows:

- Trusted Update (Digital Signature Verification):
  - The trusted update mechanism verifies the authenticity of update bundles using RSA signatures. RSA signature generation and verification depend on SHA-512 to compute message digests of the metadata file prior to signature validation.

These associations demonstrate how SHA-512 are integrated into the TOE's cryptographic operations to provide key derivation, signature verification, and key integrity validation. IV is derived using SHA2-256 hash algorithm.

TOE Security Functional Requirements Satisfied: FCS_COP.1(b)/AA and FCS_COP.1(b)/EE

FCS_COP.1(c)/AA and FCS_COP.1(c)/EE
The TOE uses the HMAC function as part of the PBKDF2 process that transforms an authorized user-supplied passphrase into a Key Encryption Key (KEK). The same HMAC configuration is also applied when generating a Data Encryption Key (DEK) at the time a keyslot is created or updated. Within PBKDF2, the HMAC function always uses the password or derived key as the HMAC key and the salt (plus iteration-specific data) as the message to be authenticated.

The HMAC parameters are as follows:

- HMAC algorithm: HMAC-SHA-512
- Key length: 512 bits
- Block size: 64 bytes
- Output MAC length: 512 bits

This configuration ensures that both KEKs and DEKs are derived with cryptographic strength suitable for securing the Master Volume Key and protecting data at rest.

TOE Security Functional Requirements Satisfied: FCS_COP.1(c)/AA and FCS_COP.1(c)/EE

FCS_COP.1(f)/EE and FCS_COP.1(g)/EE
The TOE provides data encryption and decryption using AES XTS mode with 256 bits. Key encryption and decryption use AES in CBC mode with a 256-bit key size for the protection of the Volume Master key.

TOE Security Functional Requirements Satisfied: FCS_COP.1(f)/EE and FCS_COP.1(g)/EE

 ©2026 RackTop Systems

FCS_KYC_EXT.1/AA and FCS_KYC_EXT.2/EE
The TOE conditions the passphrase authorization factor using the PBKDF2 key derivation function, producing a 256-bit Key Encryption Key (KEK). This KEK serves as the Border Encryption Value (BEV) input to the LUKS encryption engine for master key decryption.

TOE Security Functional Requirements Satisfied: FCS_KYC_EXT.1/AA and FCS_KYC_EXT.2/EE

FCS_PCC_EXT.1/AA
The TOE enforces passphrase quality through a built-in quality checking function in the `spear-luksadm --ask-for-passphrase` command. This utility is responsible for enabling LUKS-based encryption on NVMe drives within the system. The enforcement mechanism ensures that all passphrases used to derive cryptographic keys meet defined strength and complexity requirements, thereby resisting offline attacks.

The following rules are applied to all passphrases:
Minimum Length: Absolute minimum acceptable passphrase length is 8 characters. Anything below the minimum length will disable retries effectively changing the number of times a passphrase may be tried before the system gives-up on a given drive to one. As a best practice, passphrases should be no less than 15 characters. Longer passphrases are also supported.
Character Set Allowed:
Lowercase letters (a–z)
Uppercase letters (A–Z)
Digits (0–9)
Whitespace symbol (0x20)
Printable punctuation symbols (as defined by Python's `string.punctuation`)

Entropy Requirement: The passphrase must provide at least 80 bits of estimated entropy, in line with recommendations from RFC 4086 for protection against brute-force and offline dictionary attacks. [subject to change]

Pattern Restrictions: Passphrases are rejected if they contain:
Repeating sequences (e.g., ababab, 123123)
Long sequential substrings (e.g., abcdef12345, lkjhgfd)
Horizontal keyboard walks (e.g., qwerty, poiuyt)
Vertical keyboard walks (e.g., qazwsx, tgbnhy)
Excessive character repetition (e.g., more than 3–4 identical characters in a row, such as aaaaaaa)

Character Class Diversity: Passphrases must include at least two of the following character classes: lowercase, uppercase, digits, punctuation.

Whitespace Restriction: Passphrases consisting solely of whitespace are not permitted.

Passphrase Conditioning:
Before being used in cryptographic operations, the user supplied passphrase undergoes the following conditioning processed by a password-based key derivation function (PBKDF2-HMAC-SHA-512) using keyslot-specific parameters (salt and iteration count) stored in the LUKS2 metadata. The passphrase itself is not stored persistently and is not used directly for any cryptographic operation. The output of the PBKDF2 operation is an intermediate key value used to construct the Border Encryption Value (BEV), which serves as the Key Encryption Key (KEK) for decrypting the LUKS keyslot and recovering the volume encryption key. Intermediate values

 ©2026 RackTop Systems

produced during derivation are treated as submasks. This conditioning ensures that even if a passphrase is entered in a predictable form, it is transformed into a uniformly distributed cryptographic key with the required security properties.

TOE Security Functional Requirements Satisfied: FCS_PCC_EXT.1/AA

FCS_SNI_EXT.1/AA
The TOE leverages salts that are generated by the DRBG provided by the host platform. Nonces and IVs are not supported.

Salt Generation:
Salts are required inputs to the PBKDF2 key derivation function, which derives the Key Encryption Keys (KEKs) from authorized user-provided passphrases. The use of random salts prevents pre-computation and dictionary-based attacks. Salts are also used when creating the master key digest, both during the addition of a new keyslot and whenever the master key for a given volume is updated.

Salt size: All salts used for keyslot derivation and digest creation are fixed at 32 bytes.
Source of randomness: The LUKS subsystem defines the `crypt_backend_rng` interface, which provides access to supported random number generator backends. In the evaluated configuration, the TOE uses the OpenSSL backend, which invokes the `RAND_bytes` API to obtain cryptographically strong random values.

Compliance: This random number generation relies on the operating environment's implementation of a deterministic random bit generator (DRBG) consistent with AES-XTS mode, sector numbers also serve as the tweak input, ensuring that encryption of identical plaintext across different sectors produces distinct ciphertext values. This approach guarantees that salts are generated using a compliant RBG, and that IVs and tweaks are unique, deterministic, and aligned with the AES-XTS mode of operation.

TOE Security Functional Requirements Satisfied: FCS_SNI_EXT.1/AA

FCS_SNI_EXT.1/EE
The TOE does not utilize salts and nonces. The TOE creates IVs which are non-repeating and unpredictable.

IV and Tweak Handling:
The TOE also generates initialization vectors (IVs) for encryption operations. IVs are computed deterministically based on the logical sector number into which data is being written. This is implemented using the `cpu_to_le64` function, which produces a 64-bit unsigned value representing the sector number.

Uniqueness: The sector numbers begin at zero and increment sequentially; each IV value is unique for every encrypted block of data.

Tweak derivation: For AES-XTS mode, sector numbers also serve as the tweak input, ensuring that encryption of identical plaintext across different sectors produces distinct ciphertext values.
This approach guarantees that salts are generated using a compliant RBG, and that IVs and tweaks are unique, deterministic, and aligned with the AES-XTS mode of operation.

TOE Security Functional Requirements Satisfied: FCS_SNI_EXT.1/EE

FCS_VAL_EXT.1

©2026 RackTop Systems

The TOE supports a single authorization factor: a passphrase (referred to in the TOE context as the password authorization factor). When an authorized user supplies a passphrase, the TOE derives a Key Encryption Key (KEK) using the PBKDF2 key derivation function. This KEK is then used to decrypt the encrypted blocks that hold the volume master key.

Validation occurs at the point where the candidate master key is obtained. Specifically, the TOE computes a digest of the candidate volume master key and compares it against the previously stored digest associated with the volume. Only if the digests match is the master key accepted as valid. If the comparison fails, the passphrase is rejected, and no further operations are possible.

The TOE requires a power cycle after 1 failed validation attempt if the provided passphrase is equal to or less than 7 characters but allows 3 consecutive failed validation attempts before a required power cycle if the provided passphrase is greater than 7 characters.

Since the TOE supports only a single authorization factor, no submasking or multi-factor validation mechanisms are present. Likewise, the TOE does not support exiting from a Compliant power saving state using any factor other than the authorized user-supplied passphrase.

TOE Security Functional Requirements Satisfied: FCS_VAL_EXT.1/EE

FCS_KDF_EXT.1/EE
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.

TOE Security Functional Requirements Satisfied: FCS_KDF_EXT.1/EE

8.1.2 User Data Protection

FDP_DSK_EXT.1/EE
The TOE implements full-disk encryption by integrating with the Linux kernel's device-mapper crypt (dm-crypt) target. When provisioned (via the `luksFormat` operation), the TOE initializes the on-disk LUKS header and encrypts the designated storage volume. The process ensures that all data written to the protected volume thereafter is encrypted transparently before being committed to the underlying block device. Whenever a LUKS encrypted device is activated, it creates a device mapper target that handles all subsequent I/O operations. The volume key is passed to the kernel through the device mapper interface.

The **dm-crypt** target in the kernel is created through the device mapper backend. The backend constructs target parameters including:

- Cipher specification (e.g., `aes-xts-plain64`)
- Volume key or keyring reference
- IV offset and sector size
- Data device path and offset

During initialization (luksFormat), the TOE generates a random Volume Master Key and stores only its encrypted form within the on-disk LUKS header. No user data area is left unencrypted after provisioning. The only disk regions that remain in plaintext are:

- The LUKS header, which contains encrypted key material, metadata, and keyslot information, and

©2026 RackTop Systems

- For the boot drive only, partitions outside the scope of the encrypted volume (e.g., partition tables, bootloader partitions, and the EFI System Partition)

All data access is mediated through the standard Linux block I/O stack. Applications and filesystems issue read and write requests via normal operating system interfaces (e.g., read(2), write(2), mmap(2) system calls). These requests traverse the virtual file system (VFS) layer and are ultimately processed by the dm-crypt device mapper target, which enforces encryption and decryption. As a result, the TOE ensures that all standard disk access methods traverse the encryption boundary and that no bypass path exists to the underlying storage.
All access paths to protected storage pass through dm-crypt because:

- The encrypted volume is exposed as a standard block device to applications and filesystems.
- The VFS layer routes all I/O requests through the device mapper infrastructure.
- There is no direct access path to the underlying encrypted block device.
- Even raw block access tools (e.g., dd) operate on the encrypted device abstraction rather than the physical storage.

Encryption is applied at the block layer, immediately prior to data being passed to the physical device driver. For each write operation, dm-crypt retrieves the relevant data encryption material derived from the Volume Master Key, computes the initialization vector based on the logical sector number, and applies the AES-XTS encryption function. This processing occurs entirely within kernel space, ensuring that plaintext data is never written to persistent storage. Correspondingly, read operations are decrypted by dm-crypt before data is returned to the calling application.

The TOE relies on the operating environment for the underlying AES cryptographic implementation. In the evaluated configuration, cryptographic operations are performed via the Linux kernel cryptographic API as invoked by the dm-crypt target. Typical kernel interfaces include crypto_alloc_skcipher, crypto_skcipher_encrypt, and crypto_skcipher_decrypt. The kernel may utilize hardware acceleration (e.g., AES-NI) or software implementations; however, from the TOE perspective, these mechanisms are accessed through a consistent and standardized interface.

TOE Security Functional Requirements Satisfied: FDP_DSK_EXT.1/EE

8.1.3 Security Management

FMT_MOF.1/AA
The TOE supports two Complaint power savings states and only authorized users of the TOE can manage the states.

TOE Security Functional Requirements Satisfied: FMT_MOF.1/AA

FMT_SMF.1/AA and FMT_SMF.1/EE
LUKS Volume Master Key Change
The TOE handles volume master key (VMK) changes through the reencryption subsystem of `cryptsetup`, using the `cryptsetup reencrypt` command. This operation generates a new VMK and re-encrypts all existing keyslots with the new VMK, without requiring a change to the passphrase.

In the TOE configuration, only keyslot 0 is active. To initiate a VMK change, replace <`device name`> with the block device which has LUKS volume configured and execute the following command:

```
for i in 0 1; do sudo cryptsetup reencrypt --key-slot=$i --keyslot-
cipher=aes-cbc-essiv:sha256 --keyslot-key-size=256 <device name>; done
```
A valid passphrase for the active keyslot is required. If the passphrase is incorrect, the operation will fail. This procedure must be repeated for each NVMe device in the system.

LUKS Volume Master Key Destruction
Volume master key (VMK) destruction in the TOE is achieved by erasing the only active keyslot (0). This keyslot contains the metadata describing the volume and required parameters to decrypt the encrypted Volume master key (VMK).

The TOE provides a dedicated tool for this purpose: `spear-lukswipe`

As with VMK change, knowledge of the correct passphrase is required. If the passphrase is entered incorrectly, the operation will fail.

TOE Software Updates
Software updates are applied using the `updatefde` utility from within the running operating system. Refer to the User Guidance documentation for detailed usage instructions.

TOE Security Functional Requirements Satisfied: FMT_SMF.1/AA and FMT_SMF.1/EE

8.1.4 Protection of the TSF

FPT_KYP_EXT.1/AA and FPT_KYP_EXT.1/EE
The authorization acquisition functions in the TOE do not store any key material in the nonvolatile memory. The encryption engine component is responsible for deriving the intermediate Key Encryption Key, which is not stored, persistently and is used solely for encrypting (protecting) and decrypting (releasing) the Volume Master Key. This key, frequently referred to as the Volume Master Key (VMK) is generated at the time of volume creation by the encryption engine and stored on non-volatile storage.

TOE Security Functional Requirements Satisfied: FPT_KYP_EXT.1/AA and FPT_KYP_EXT.1/EE

FPT_PWR_EXT.1/AA and FPT_PWR_EXT.1/EE
The TOE supports two Compliant power-saving states, G2 (S5) – Soft Off and G3 – Mechanical Off.

G2 (S5) – Soft Off: in this state, the system is fully shut down, consuming minimal power. No user mode or system mode code can run in this state. This state occurs when a shutdown is explicitly requested by an authorized user. Recovery requires a full reboot.

G3 – Mechanical Off: In this state, the system is completely powered off and consumes no energy. Recovery requires a full reboot.

The drive is considered secure only when in a powered-off state, up until it is powered on and receives the initial authorization factor during boot. In both compliant states, all drives are powered off.

RAM receives no power in either G2 (S5) or G3 thus no keys remain resident in system memory. Time spent in S5 during shutdown and restart is sufficient to ensure that RAM is fully cleared.

©2026 RackTop Systems

TOE Security Functional Requirements Satisfied: FPT_PWR_EXT.1/AA and FPT_PWR_EXT.1/EE

FPT_PWR_EXT.2/AA and FPT_PWR_EXT.2/EE
G2(S5) power state is reached after an authorized user issues a “shutdown” command, and G3 power state is reached when the system is turned off or physically loses power.

TOE Security Functional Requirements Satisfied: FPT_PWR_EXT.2/AA and FPT_PWR_EXT.2/EE

FPT_TUD_EXT.1/AA and FPT_TUD_EXT.1/EE
The TOE updates are signed using RSA algorithm. The TOE installs updates only after verifying their authenticity by checking the signature of the update bundle. Details of this process are covered in FCS_COP.1(a). Authorized updates are packaged, signed and distributed by RackTop Systems. Only authorized users are allowed to perform updates, due to the elevated permissions required by the system. The RPM versions installed on the TOE can be verified by executing the following commands:

```
dnf info spear-utils | grep spear-utils
dnf info updatefde | grep updatefde
```

The RPM output should be the following:
spear-utils-1.0.6-33
updatefde-1.0.2-44

TOE Security Functional Requirements Satisfied: FPT_TUD_EXT.1/AA and FPT_TUD_EXT.1/EE

FPT_TST_EXT.1/EE
Upon each boot a series of tests validating proper function of the cryptographic module are executed. Any algorithm or suite that does not successfully pass the test is excluded from the available algorithms within the module. The TOE spans two contexts, Operating System kernel and userspace. The Operating System provides a cryptographic module as a service to other components in the kernel. The `dm-crypt` module, which handles all the read and write operations between the filesystems and physical storage is using the kernel crypto module, specifically AES in XTS mode to transparently encrypt all data going to the drive(s) and transparently decrypt all data being read from the drive(s). While the module is performing its integrity tests which includes the Known Answer Tests listed here, no consumers in the kernel can access and use any algorithms present in the Kernel Crypto module. Only relevant algorithms are mentioned in Table 11 and Table 12 with vast majority of tests which aren't relevant being omitted.

TOE Security Functional Requirements Satisfied: FPT_TST_EXT.1/EE

©2026 RackTop Systems

Table 10 - Kernal Crypto Module KATs

Algorithm	Parameters	Condition	Type	Test
AES XTS	128 bits, 256 bits	Initialization	Cryptographic Algorithm Self-Test	KAT encryption and decryption
Counter DRBG	128 bits, 192 bits, 256 bits keys With/without PR	Initialization	Cryptographic Algorithm Self-Test	KAT seed, generate
Hash DRBG	SHA-256 With/without PR	Initialization	Cryptographic Algorithm Self-Test	KAT seed, generate
HMAC DRBG	SHA-256, SHA-512 With/without PR	Initialization	Cryptographic Algorithm Self-Test	KAT seed, generate
Entropy source	1024 samples	Source Initialization	Cryptographic Algorithm Self-Test	RCT, entropy source startup test
Entropy source	1024 samples	Source Initialization	Cryptographic Algorithm Self-Test	APT, entropy source startup test
Entropy source	Cutoff C = 61	Continuous	Cryptographic Algorithm Self-Test	RCT, entropy source continuous test
Entropy source	Cutoff C = 355	Continuous	Cryptographic Algorithm Self-Test	APT, entropy source continuous test

In userspace the OpenSSL cryptographic module provides the TOE with all the necessary cryptographic services. The OpenSSL module at the time of its initial installation executes a suite which includes several Known Answer Tests, some of which cover algorithms which the TOE depends upon. This happens once only, and if there are any failures, the installation of the OpenSSL package fails. Each time the FIPS provider module is loaded, before its algorithms are available to the calling application, the module performs pre-operational self-tests and conditional self-tests. No services provided by the module are available while the tests are running. The module does not return control to the calling application until after the tests are successful, and module is allowed to remain operational.

Table 11 – OpenSSL Crypto Module (user space) KATs

Algorithm	Parameters	Condition	Type	Test
HMAC	SHA-512	Initialization	Pre-operational Integrity Test	MAC tag verification on fips.so file
SHA	SHA-256, SHA-512	Initialization	Cryptographic Algorithm Self-Test	KAT hashing

©2026 RackTop Systems

Algorithm	Parameters	Condition	Type	Test
AES CBC	256 bits	Initialization	Cryptographic Algorithm Self-Test	KAT Key Encryption
PBKDF2	SHA-512 with 4096 iterations and 256-bit salt	Initialization	Cryptographic Algorithm Self-Test	KAT password based key derivation
Counter DRBG	AES-128 with derivation function	Initialization	Cryptographic Algorithm Self-Test	KAT DRBG generation and reseed
Hash DRBG	SHA-256 with prediction resistance	Initialization	Cryptographic Algorithm Self-Test	KAT DRBG generation and reseed
HMAC DRBG	SHA-1 with prediction resistance	Initialization	Cryptographic Algorithm Self-Test	KAT DRBG generation and reseed
RSA	PKCS#1 v1.5 with SHA-512 with 3072 bit key and 4096 bit key	Initialization	Cryptographic Algorithm Self-Test	KAT DRBG generation and reseed

Validation During `luksFormat`

When creating a new encrypted volume with `luksFormat`, the TOE performs the following validation checks:

Storage layout validation – Verifies that the data offset is aligned to 4096-byte sectors, provides enough space for the full LUKS header and all keyslots, and that device size and sector size are consistent. Arbitrary offsets (e.g., 1, 16385, 32) are rejected.

Sector size validation – Ensures that the sector size is a power of 2 between 512 and 4096 bytes. The TOE only supports 512-byte sectors, as larger sizes require unsupported `dm-crypt` extensions.

Region boundary validation – Verifies that keyslot areas and data regions do not overlap. Cryptographic support checks – Ensures that the requested cipher is supported by the kernel.

Entropy source validation – Verifies that a functional random number generator is available.

Validation During `luksOpen`

When opening an existing volume with `luksOpen`, the TOE validates the LUKS header to ensure integrity and compatibility:

Header identification – Verifies the presence of valid magic bytes and confirms the header version is LUKS2.

Header structure validation – Ensures the header size is within bounds (≥16 KB and below the maximum), and that the header offset is correct.

Algorithm validation – Verifies that the specified checksum algorithm is supported. Integrity and consistency checks – Confirms header integrity by verifying magic values, recomputing and comparing checksums, and validating all JSON metadata sections.

©2026 RackTop Systems

9. Rationale

9.1 Conformance Claims Rationale

This Security Target conforms to Common Criteria Standard for Information Technology Security Evaluations, Version 3.1, Revision 5, April 2017; CC Part 2 extended; CC Part 3 conformant. This ST conforms to the collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition, Version 2.0 + Errata 20190201, February 1, 2019 (FDE_AA) and collaborative Protection Profile for Full Drive Encryption – Encryption Engine, Version 2.0 + Errata 20190201, February 1, 2019 (FDE_EE).

9.1.1 Variance Between the PP and this ST

There is no variance between the FDE AA cPP and FDE EE cPP and this ST.

9.1.2 Security Assurance Requirements Rationale

This ST claims exact conformance to the collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition, Version 2.0 + Errata 20190201, February 1, 2019 (FDE_AA) and collaborative Protection Profile for Full Drive Encryption – Encryption Engine, Version 2.0 + Errata 20190201, February 1, 2019 (FDE_EE).

9.1.3 Dependency Rationale

The SFRs in this Security Target represent the SFRs identified in the FDE AA cPP and FDE EE cPP. As such, the FDE AA SFR and FDE EE SFR dependency rationale is deemed acceptable since the PP itself has been validated.

©2026 RackTop Systems

10. Acronyms and Terms

10.1 Acronyms

Table 12 – Acronyms

Acronym	Definition
AA	Authorization Acquisition
AES	Advanced Encryption Standard
BEV	Border Encrypt Value
CAVP	Cryptographic Algorithm Validation Program
CC	Common Criteria
CEM	Common Evaluation Methodology
DRBG	Deterministic Random Bit Generator
EE	Encryption Engine
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
ISO/IEC	International Organization for Standardization/ International Electrotechnical Commission
IT	Information Technology
IV	Initialization Vector
KAT	Known Answer Test
KEK	Key Encryption Key
LUKS	Linux Unified Key Setup
MAC	Message Authentication Code
OS	Operating System
OSP	Organizational Security Policy
PBKDF2	Password-Based Key Derivation Function 2
PP	Protection Profile
PUB	Publication
RBG	Random Bit Generator
RSA	Ron Rivest, Adi Shamir and Leonard Adleman
SAR	Security Assurance Requirement
SED	Self-Encrypting Drive
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SP	Special Publication
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
UEFI	Unified Extensible Firmware Interface

©2026 RackTop Systems

Prepared by:
Corsec Security, Inc.



12600 Fair Lakes Circle, Suite 210
Fairfax, VA 22033
United States of America

Phone: +1 703 267 6050
Email: info@corsec.com
http://www.corsec.com

©2026 RackTop Systems