Cigent Software FDE version 1.2.1 Security Target

Version 1.5
02/04/2026

Prepared for:

Cigent Technology, Inc.

2211 Widman Way, Suite 150
Fort Myers, Florida 33901

Prepared By:

www.gossamersec.com

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is Cigent Software FDE provided by Cigent Technology, Inc.. The TOE is being evaluated as a full drive encryption solution.

The Security Target contains the following additional sections:

- Conformance Claims (Section 2)
- Security Objectives (Section 3)
- Extended Components Definition (Section 4)
- Security Requirements (Section 5)
- TOE Summary Specification (Section 6)

Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations. In this ST, iteration may be indicated by a parenthetical number placed at the end of the component. For example FDP_ACC.1(1) and FDP_ACC.1(2) indicate that the ST includes two iterations of the FDP_ACC.1 requirement. Alternately, a usually descriptive textual extension may be added after a slash (/) character to identify a specific iteration. For example, iterations of a requirement such as FCS_COP.1 might be identified as FCS_COP.1/HASH and FCS_COP.1/CRYPT.
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").
- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.

1.1 Security Target Reference

ST Title – Cigent Software FDE version 1.2.1 Security Target

ST Version – Version 1.5

ST Date – 02/04/2026

1.2 TOE Reference

TOE Identification – Cigent Technology, Inc. Cigent Software FDE version 1.2.1

TOE Developer – Cigent Technology, Inc.

Evaluation Sponsor – Cigent Technology, Inc.

1.3 TOE Overview

The Target of Evaluation (TOE) is the Cigent Software FDE version 1.2.1. Cigent's Software FDE supports many different operating systems and was evaluated on Windows 11. The evaluation testing utilized a laptop with an Intel Core Ultra 7 155H CPU; however, the TOE does not depend upon any specific hardware configuration.

1.4 TOE Description

The software TOE consists an OS-independent UEFI pre-boot authentication component (the "AA") and an OS-specific file system driver component (the "EE").

The TOE's AA component accepts the user's authorization factors and if correct, unlocks the OS system partition, allowing the OS to boot. During the OS boot, the TOE's EE component allows the OS to decrypt and encrypt drive reads and writes (respectively).

1.4.1 TOE Architecture

The TOE has both a UEFI pre-boot authentication component and OS file system driver (specific to each supported Operating System) that run on any computer with an x86 compatible CPU. The pre-boot authentication component provides Authorization Acquisition (AA) functionality and presents a user with a graphic interface to enter the password and/or access a smartcard. The OS file system driver provides the Encryption Engine (EE) functionality to allow the OS to access the encrypted drive.

1.4.1.1 Physical Boundaries

The TOE is purely a software TOE that one installs onto the drive within an x86-based computer.

1.4.1.2 Logical Boundaries

This section summarizes the security functions provided by the Cigent Software FDE:

- Cryptographic support
- User data protection
- Security management
- Protection of the TSF

1.4.1.2.1 Cryptographic support

The TOE includes cryptographic functionality for key management, user authentication, and block-based encryption including: symmetric key generation, encryption/decryption, cryptographic hashing, keyed-hash message authentication, and password-based key derivation. These functions are supported with suitable random bit generation, key derivation, salt generation, initialization vector generation, secure key storage, and key destruction. These primitive cryptographic functions are used to encrypt Data-At-Rest (including the generation and protection of keys and key encryption keys) used by the TOE.

1.4.1.2.2 User data protection

The TOE performs Full Drive Encryption on all partitions on the drive (so that no plaintext exists) and does so without user intervention.

1.4.1.2.3 Security management

The TOE provides each of required management services to manage the full drive encryption using a graphical user interface.

1.4.1.2.4 Protection of the TSF

The TOE implements a number of features to protect itself to ensure the reliability and integrity of its security features. It protects key and key material, and includes functions to perform self-tests and software/firmware integrity checking so that it might detect when it is failing or may be corrupt. If any of the self-tests fail, the TOE will not go into an operational mode.

1.4.2 TOE Documentation

Cigent SW FDE Installation Guide and User Manual Dec 2025, Version 1.2.1 [Admin Guide]

2. Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017.
  - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017.
  - Part 3 Conformant
- Package Claims:
  - collaborative Protection Profile for Full Drive Encryption - Authorization Acquisition, Version 2.0 + Errata 20190201, 01 February 2019(FDEAAcPP20E)
  - collaborative Protection Profile for Full Drive Encryption - Encryption Engine, Version 2.0 + Errata 20190201, 01 February 2019 (FDEEEcPP20E)

Package	Technical Decision	Applied	Notes
CPP_FDE_AA_V2.0E	TD0458 - FIT Technical Decision for FPT_KYP_EXT.1 evaluation activities	Yes	
CPP_FDE_AA_V2.0E	TD0606 - FIT Technical Recommendation for Evaluating a NAS against the FDE AA and FDEE	No	Product not a NAS
CPP_FDE_AA_V2.0E	TD0766 - FIT Technical Decision for FCS_CKM.4(d) Test Notes	Yes	
CPP_FDE_AA_V2.0E	TD0765 - FIT Technical Decision for FMT_MOF.1	Yes	
CPP_FDE_AA_V2.0E	TD0760 - FIT Technical Decision for FCS_SNI_EXT.1.3, FCS_COP.1(f)	Yes	
CPP_FDE_AA_V2.0E	TD0759 - FIT Technical Decision for FCS_AFA_EXT.1.1	Yes	
CPP_FDE_AA_V2.0E	TD0769 - FIT Technical Decision for FPT_KYP_EXT.1.1	Yes	
CPP_FDE_AA_V2.0E	TD0767 - FIT Technical Decision for FMT_SMF.1.1	Yes	
CPP_FDE_AA_V2.0E	TD0769 - FIT Technical Decision for FPT_KYP_EXT.1.1	Yes	
CPP_FDE_AA_V2.0E	TD0929 - FIT Technical Decision: Clarification to FCS_PCC_EXT.1.1	Yes	
CPP_FDE_EE_V2.0E	TD0464 - FIT Technical Decision for FPT_PWR_EXT.1 compliant power saving states	Yes	
CPP_FDE_EE_V2.0E	TD0460 - FIT Technical Decision for FPT_PWR_EXT.1 non-compliant power saving states	Yes	
CPP_FDE_EE_V2.0E	TD0458 - FIT Technical Decision for FPT_KYP_EXT.1 evaluation activities	Yes	
CPP_FDE_EE_V2.0E	TD0606 - FIT Technical Recommendation for Evaluating a NAS against the FDE AA and FDEE	No	Product not a NAS
CPP_FDE_EE_V2.0E	TD0766 - FIT Technical Decision for FCS_CKM.4(d) Test Notes	Yes	
CPP_FDE_EE_V2.0E	TD0769 - FIT Technical Decision for FPT_KYP_EXT.1.1	Yes	

Table 1 Technical Decisions

2.1 Conformance Rationale

The ST conforms to the FDEAAcPP20E/FDEEEcPP20E. The security problem definition, security objectives, and security requirements are referenced and available in the PP(s) listed above.

3. Security Objectives

The Security Problem Definition may be found in the FDEAAcPP20E/FDEEEcPP20E and this section reproduces only the corresponding Security Objectives for operational environment for reader convenience. The FDEAAcPP20E/FDEEEcPP20E offers additional information about the identified security objectives, but that has not been reproduced here and the FDEAAcPP20E/FDEEEcPP20E should be consulted if there is interest in that material.

In general, the FDEAAcPP20E/FDEEEcPP20E has defined Security Objectives appropriate for full drive encryption solution and as such are applicable to the Cigent Software FDE TOE.

3.1 Security Objectives for the Operational Environment

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 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 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 Volatile memory is cleared after power-off so memory remnant attacks are infeasible.

OE.SINGLE_USE_ET External tokens that contain authorization factors will be used for no other purpose than to store the external token authorization factor.

OE.STRONG_ENVIRONMENT_CRYPTO The Operating Environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE and Appendix A.

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

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

4. Extended Components Definition

All of the extended requirements in this ST have been drawn from the FDEAAcPP20E/FDEEEcPP20E. The FDEAAcPP20E/FDEEEcPP20E defines the following extended requirements and since they are not redefined in this ST the FDEAAcPP20E/FDEEEcPP20E should be consulted for more information in regard to those CC extensions.

Extended SFRs:

- FDEAAcPP20E:FCS_AFA_EXT.1: Authorization Factor Acquisition - per TD0759
- FDEAAcPP20E:FCS_AFA_EXT.2: Timing of Authorization Factor Acquisition
- FDEAAcPP20E:FCS_CKM_EXT.4(a): Cryptographic Key and Key Material Destruction (Destruction Timing)
- FDEEEcPP20E:FCS_CKM_EXT.4(a): Cryptographic Key and Key Material Destruction (Destruction Timing)
- FDEAAcPP20E:FCS_CKM_EXT.4(b): Cryptographic Key and Key Material Destruction (Power Management)
- FDEEEcPP20E:FCS_CKM_EXT.4(b): Cryptographic Key and Key Material Destruction (Power Management)
- FDEEEcPP20E:FCS_CKM_EXT.6: Cryptographic Key Destruction Types
- FDEAAcPP20E:FCS_KDF_EXT.1: Cryptographic Key Derivation
- FDEAAcPP20E:FCS_KYC_EXT.1: Key Chaining (Initiator)
- FDEEEcPP20E:FCS_KYC_EXT.2: Key Chaining (Recipient)
- FDEAAcPP20E:FCS_PCC_EXT.1: Cryptographic Password Construct and Conditioning - per TD0929
- FDEAAcPP20E:FCS_RBG_EXT.1: Extended: Cryptographic Operation (Random Bit Generation)
- FDEEEcPP20E:FCS_RBG_EXT.1: Random Bit Generation
- FDEAAcPP20E:FCS_SMC_EXT.1: Submask Combining
- FDEAAcPP20E:FCS_SNI_EXT.1: Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) - per TD0760
- FDEEEcPP20E:FCS_SNI_EXT.1: Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
- FDEEEcPP20E:FCS_VAL_EXT.1: Validation
- FDEEEcPP20E:FDP_DSK_EXT.1: Protection of Data on Disk
- FDEEEcPP20E:FPT_FUA_EXT.1: Firmware Update Authentication
- FDEAAcPP20E:FPT_KYP_EXT.1: Protection of Key and Key Material - per TD0769
- FDEEEcPP20E:FPT_KYP_EXT.1: Protection of Key and Key Material - per TD0769
- FDEAAcPP20E:FPT_PWR_EXT.1: Power Saving States
- FDEEEcPP20E:FPT_PWR_EXT.1: Power Saving States
- FDEAAcPP20E:FPT_PWR_EXT.2: Timing of Power Saving States
- FDEEEcPP20E:FPT_PWR_EXT.2: Timing of Power Saving States
- FDEEEcPP20E:FPT_RBP_EXT.1: Rollback Protection
- FDEAAcPP20E:FPT_TST_EXT.1: TSF Testing
- FDEEEcPP20E:FPT_TST_EXT.1: TSF Testing
- FDEAAcPP20E:FPT_TUD_EXT.1: Trusted Update
- FDEEEcPP20E:FPT_TUD_EXT.1: Trusted Update

5. Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

The SFRs have all been drawn from the FDEAAcPP20E/FDEEEcPP20E. The refinements and operations already performed in the FDEAAcPP20E/FDEEEcPP20E are not identified (e.g., highlighted) here, rather the requirements have been copied from the FDEAAcPP20E/FDEEEcPP20E and any residual operations have been completed herein. Of particular note, the FDEAAcPP20E/FDEEEcPP20E made a number of refinements and completed some of the SFR operations defined in the Common Criteria (CC) and that PP should be consulted to identify those changes if necessary.

The SARs are also drawn from the FDEAAcPP20E/FDEEEcPP20E. The FDEAAcPP20E/FDEEEcPP20E should be consulted for the assurance activity definitions.

5.1 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by Cigent Software FDE TOE.

Requirement Class	Requirement Component
FCS: Cryptographic support	FDEAAcPP20E:FCS_AFA_EXT.1: Authorization Factor Acquisition - per TD0759
	FDEAAcPP20E:FCS_AFA_EXT.2: Timing of Authorization Factor Acquisition
	FDEAAcPP20E:FCS_CKM.1(b): Cryptographic key generation (Symmetric Keys)
	FDEEEcPP20E:FCS_CKM.1(c): Cryptographic Key Generation (Data Encryption Key)
	FDEAAcPP20E:FCS_CKM.4(a): Cryptographic Key Destruction (Power Management)
	FDEEEcPP20E:FCS_CKM.4(a): Cryptographic Key Destruction (Power Management)
	FDEAAcPP20E:FCS_CKM.4(d): Cryptographic Key Destruction (Software TOE, 3rd Party Storage) - per TD0766
	FDEEEcPP20E:FCS_CKM.4(d): Cryptographic Key Destruction (Software TOE, 3rd Party Storage) - per TD0766
	FDEAAcPP20E:FCS_CKM_EXT.4(a): Cryptographic Key and Key Material Destruction (Destruction Timing)
	FDEEEcPP20E:FCS_CKM_EXT.4(a): Cryptographic Key and Key Material Destruction (Destruction Timing)
	FDEAAcPP20E:FCS_CKM_EXT.4(b): Cryptographic Key and Key Material Destruction (Power Management)
	FDEEEcPP20E:FCS_CKM_EXT.4(b): Cryptographic Key and Key Material Destruction (Power Management)
	FDEEEcPP20E:FCS_CKM_EXT.6: Cryptographic Key Destruction Types
	FDEAAcPP20E:FCS_COP.1(a): Cryptographic Operation (Signature Verification)
	FDEEEcPP20E:FCS_COP.1(a): Cryptographic Operation (Signature Verification)
	FDEAAcPP20E:FCS_COP.1(b): Cryptographic operation (Hash Algorithm)
	FDEEEcPP20E:FCS_COP.1(b): Cryptographic Operation (Hash Algorithm)
	FDEAAcPP20E:FCS_COP.1(c): Cryptographic operation (Keyed Hash Algorithm)
	FDEEEcPP20E:FCS_COP.1(c): Cryptographic Operation (Message Authentication)
	FDEAAcPP20E:FCS_COP.1(f): Cryptographic Operation (AES Data Encryption/Decryption) – per TD0760
	FDEEEcPP20E:FCS_COP.1(f): Cryptographic Operation (AES Data Encryption/Decryption)
	FDEAAcPP20E:FCS_COP.1(g): Cryptographic Operation (Key Encryption)
	FDEEEcPP20E:FCS_COP.1(g): Cryptographic Operation (Key Encryption)

	FDEAAcPP20E:FCS_KDF_EXT.1: Cryptographic Key Derivation
	FDEEEcPP20E:FCS_KDF_EXT.1: Cryptographic Key Derivation
	FDEAAcPP20E:FCS_KYC_EXT.1: Key Chaining (Initiator)
	FDEEEcPP20E:FCS_KYC_EXT.2: Key Chaining (Recipient)
	FDEAAcPP20E:FCS_PCC_EXT.1: Cryptographic Password Construct and Conditioning - per TD0929
	FDEAAcPP20E:FCS_RBG_EXT.1: Extended: Cryptographic Operation (Random Bit Generation)
	FDEAAcPP20E:FCS_SMC_EXT.1: Submask Combining
	FDEAAcPP20E:FCS_SNI_EXT.1: Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) - per TD0760
	FDEEEcPP20E:FCS_SNI_EXT.1: Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation)
	FDEAAcPP20E:FCS_VAL_EXT.1: Validation
	FDEEEcPP20E:FCS_VAL_EXT.1: Validation
FDP: User data protection	FDEEEcPP20E:FDP_DSK_EXT.1: Protection of Data on Disk
FMT: Security management	FDEAAcPP20E:FMT_MOF.1: Management of Functions Behavior - per TD0765
	FDEAAcPP20E:FMT_SMF.1: Specification of Management Functions - per TD0767
	FDEEEcPP20E:FMT_SMF.1: Specification of Management Functions
	FDEAAcPP20E:FMT_SMR.1: Security Roles
FPT: Protection of the TSF	FDEAAcPP20E:FPT_KYP_EXT.1: Protection of Key and Key Material - per TD0769
	FDEEEcPP20E:FPT_KYP_EXT.1: Protection of Key and Key Material - per TD0769
	FDEAAcPP20E:FPT_PWR_EXT.1: Power Saving States
	FDEEEcPP20E:FPT_PWR_EXT.1: Power Saving States
	FDEAAcPP20E:FPT_PWR_EXT.2: Timing of Power Saving States
	FDEEEcPP20E:FPT_PWR_EXT.2: Timing of Power Saving States
	FDEAAcPP20E:FPT_TST_EXT.1: TSF Testing
	FDEEEcPP20E:FPT_TST_EXT.1: TSF Testing
	FDEAAcPP20E:FPT_TUD_EXT.1: Trusted Update
	FDEEEcPP20E:FPT_TUD_EXT.1: Trusted Update

Table 2 TOE Security Functional Components

5.1.1 Cryptographic support (FCS)

5.1.1.1 Authorization Factor Acquisition - per TD0759 (FDEAAcPP20E:FCS_AFA_EXT.1)

FDEAAcPP20E:FCS_AFA_EXT.1.1

The TSF shall accept the following authorization factors: [
- a submask derived from a password authorization factor conditioned as defined in FCS_PCC_EXT.1,
- an external Smartcard factor that is protecting a submask that is [generated by the TOE (using the RBG as specified in FCS_RBG_EXT.1)] protected using [RSA with Key size [2048 bits]] with user presence proved by presentation of the smartcard and [an OE defined PIN]].
(TD0759 applied)

5.1.1.2 Timing of Authorization Factor Acquisition (FDEAAcPP20E:FCS_AFA_EXT.2)

FDEAAcPP20E:FCS_AFA_EXT.2.1

The TSF shall reacquire the authorization factor(s) specified in FCS_AFA_EXT.1 upon transition

from any Compliant power saving state specified in FPT_PWR_EXT.1 prior to permitting access to plaintext data.

5.1.1.3 Cryptographic key generation (Symmetric Keys) (FDEAAcPP20E:FCS_CKM.1(b))

FDEAAcPP20E:FCS_CKM.1.1(b)

Refinement: The TSF shall generate symmetric cryptographic keys using a Random Bit Generator as specified in FCS_RBG_EXT.1 and specified cryptographic key sizes [***256 bit***] that meet the following: No Standard.

5.1.1.4 Cryptographic Key Generation (Data Encryption Key) (FDEEEcPP20E:FCS_CKM.1(c))

FDEEEcPP20E:FCS_CKM.1.1(c)

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***] and specified cryptographic key sizes [***256 bits***].

5.1.1.5 Cryptographic Key Destruction (Power Management) (FDEAAcPP20E:FCS_CKM.4(a))

FDEAAcPP20E:FCS_CKM.4.1(a)

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

5.1.1.6 Cryptographic Key Destruction (Power Management) (FDEEEcPP20E:FCS_CKM.4(a))

FDEEEcPP20E:FCS_CKM.4.1(a)

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 that meets the following: a key destruction method specified in FCS_CKM_EXT.6.

5.1.1.7 Cryptographic Key Destruction (Software TOE, 3rd Party Storage) - per TD0766 (FDEAAcPP20E:FCS_CKM.4(d))

FDEAAcPP20E:FCS_CKM.4.1(d)

Refinement: The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [
- For volatile memory, the destruction shall be executed by a [single overwrite consisting of [
- zeroes,
- ones]],
- For non-volatile storage that consists of the invocation of an interface provided by the underlying platform that [instructs the underlying platform to destroy the abstraction that represents the key]]
that meets the following: no standard.

5.1.1.8 Cryptographic Key Destruction (Software TOE, 3rd Party Storage) - per TD0766 (FDEEEcPP20E:FCS_CKM.4(d))

FDEEEcPP20E: FCS_CKM.4.1(d)

Refinement: The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [
- For volatile memory, the destruction shall be executed by a [single overwrite consisting of [
- zeroes,
- ones]],
- For non-volatile storage that consists of the invocation of an interface provided by the underlying platform that [instructs the underlying platform to destroy the abstraction that represents the key]]

that meets the following: no standard.

5.1.1.9 Cryptographic Key and Key Material Destruction (Destruction Timing) (FDEAAcPP20E:FCS_CKM_EXT.4(a))

FDEAAcPP20E:FCS_CKM_EXT.4.1(a)

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

5.1.1.10 Cryptographic Key and Key Material Destruction (Destruction Timing) (FDEEEcPP20E:FCS_CKM_EXT.4(a))

FDEEEcPP20E:FCS_CKM_EXT.4.1(a)

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

5.1.1.11 Cryptographic Key and Key Material Destruction (Power Management) (FDEAAcPP20E:FCS_CKM_EXT.4(b))

FDEAAcPP20E:FCS_CKM_EXT.4.1(b)

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

5.1.1.12 Cryptographic Key and Key Material Destruction (Power Management) (FDEEEcPP20E:FCS_CKM_EXT.4(b))

FDEEEcPP20E:FCS_CKM_EXT.4.1(b)

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

5.1.1.13 Cryptographic Key Destruction Types (FDEEEcPP20E:FCS_CKM_EXT.6)

FDEEEcPP20E:FCS_CKM_EXT.6.1

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

5.1.1.14 Cryptographic Operation (Signature Verification) (FDEAAcPP20E:FCS_COP.1(a))

FDEAAcPP20E: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]***]
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***].

5.1.1.15 Cryptographic Operation (Signature Verification) (FDEEEcPP20E:FCS_COP.1(a))

FDEEEcPP20E:FCS_COP.1.1(a)

The TSF shall perform cryptographic signature services (verification) in accordance with a [***RSA Digital Signature Algorithm with a key size (modulus) of [4096-bit]***] that meets 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***].

5.1.1.16 Cryptographic operation (Hash Algorithm) (FDEAAcPP20E:FCS_COP.1(b))

FDEAAcPP20E:FCS_COP.1.1(b)

Refinement: The TSF shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-512***] that meet the following: ISO/IEC 10118-3:2004.

5.1.1.17 Cryptographic Operation (Hash Algorithm) (FDEEEcPP20E:FCS_COP.1(b))

FDEEEcPP20E:FCS_COP.1.1(b)

The TSF shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-256, SHA-512***] that meet the following: ISO/IEC 10118-3:2004.

5.1.1.18 Cryptographic operation (Keyed Hash Algorithm) (FDEAAcPP20E:FCS_COP.1(c))

FDEAAcPP20E:FCS_COP.1.1(c)

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'.

5.1.1.19 Cryptographic Operation (Message Authentication) (FDEEEcPP20E:FCS_COP.1(c))

FDEEEcPP20E:FCS_COP.1.1(c)

Refinement: The TSF shall perform 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 800-16 38B***].

5.1.1.20 Cryptographic Operation (AES Data Encryption/Decryption) (FDEAAcPP20E:FCS_COP.1(f)) - per TD0760

FDEAAcPP20E:FCS_COP.1.1(f)

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***].

5.1.1.21 Cryptographic Operation (AES Data Encryption/Decryption) (FDEEEcPP20E:FCS_COP.1(f))

FDEEEcPP20E:FCS_COP.1.1(f)

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

5.1.1.22 Cryptographic Operation (Key Encryption) (FDEAAcPP20E:FCS_COP.1(g))

FDEAAcPP20E:FCS_COP.1.1(g)

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

5.1.1.23 Cryptographic Operation (Key Encryption) (FDEEEcPP20E:FCS_COP.1(g))

FDEEEcPP20E:FCS_COP.1.1(g)

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

5.1.1.24 Cryptographic Key Derivation (FDEAAcPP20E:FCS_KDF_EXT.1)

FDEAAcPP20E:FCS_KDF_EXT.1.1

The TSF shall accept [***a conditioned password submask, imported submask***] to derive an intermediate key, as defined in [***NIST SP 800-108 [KDF in Counter Mode], NIST SP 800-132***],

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

5.1.1.25 Cryptographic Key Derivation (FDEEEcPP20E:FCS_KDF_EXT.1)

FDEEEcPP20E:FCS_KDF_EXT.1.1

The TSF shall accept [***imported submask***] to derive an intermediate key, as defined in [***NIST SP 800-108 [KDF in Counter Mode], NIST SP 800-132***], using the keyed-hash functions specified in FCS_COP.1(c), such that the output is at least of equivalent security strength (in number of bits) to the BEV.

5.1.1.26 Key Chaining (Initiator) (FDEAAcPP20E:FCS_KYC_EXT.1)

FDEAAcPP20E:FCS_KYC_EXT.1.1

The TSF shall maintain a key chain of: [***- intermediate keys originating from one or more submask(s) to the BEV using the following method(s): [***
- key derivation as specified in FCS_KDF_EXT.1,
- 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.

FDEAAcPP20E:FCS_KYC_EXT.1.2

The TSF shall provide at least a [***256 bit***] BEV to [**the EE**] [***after the TSF has successfully performed the validation process as specified in FCS_VAL_EXT.1***].

5.1.1.27 Key Chaining (Recipient) (FDEEEcPP20E:FCS_KYC_EXT.2)

FDEEEcPP20E:FCS_KYC_EXT.2.1

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

FDEEEcPP20E:FCS_KYC_EXT.2.2

The TSF shall maintain a chain of intermediary keys originating from the BEV to the DEK using the following method(s): [
- key derivation as specified in FCS_KDF_EXT.1,
- 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.

5.1.1.28 Cryptographic Password Construct and Conditioning - per TD0929 (FDEAAcPP20E:FCS_PCC_EXT.1)

FDEAAcPP20E:FCS_PCC_EXT.1.1

A password used by the TSF to generate a password authorization factor shall enable at least [**128**] characters in the set of upper case characters, lower case characters, numbers, and ['~', '!', '@', '#', '$', '^', '&', '*', '(', ')', '_', '-', '+', '=', '[', ']', ':', '<', '>', '.'] and shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC-[***SHA-512***], with [***[120,842] iterations***], and output cryptographic key sizes [**256 bit**] that meet the following: NIST SP 800-132.
(TD0929 applied)

5.1.1.29 Extended: Cryptographic Operation (Random Bit Generation) (FDEAAcPP20E:FCS_RBG_EXT.1)

FDEAAcPP20E:FCS_RBG_EXT.1.1

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

FDEAAcPP20E:FCS_RBG_EXT.1.2

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.

5.1.1.30 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) - per TD0760 (FDEAAcPP20E:FCS_SNI_EXT.1)

FDEAAcPP20E:FCS_SNI_EXT.1.1
The TSF shall [***use salts that are generated by a [DRBG as specified in FCS_RBG_EXT.1]***].

FDEAAcPP20E:FCS_SNI_EXT.1.2
The TSF shall use [***no nonces***].

FDEAAcPP20E:FCS_SNI_EXT.1.3
The TSF shall [***create IVs in the following matter [CBC: IVs shall be non-repeating and unpredictable]***]. (TD0760 applied)

5.1.1.31 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) (FDEEEcPP20E:FCS_SNI_EXT.1)

FDEEEcPP20E:FCS_SNI_EXT.1.1
The TSF shall use [***use salts that are generated by a [DRBG as specified in FCS_RBG_EXT.1]***].

FDEEEcPP20E:FCS_SNI_EXT.1.2
The TSF shall use [***no nonces***].

FDEEEcPP20E:FCS_SNI_EXT.1.3
The TSF shall create IVs in the following manner [***CBC: IVs shall be non-repeating and unpredictable***].

5.1.1.32 Validation (FDEAAcPP20E:FCS_VAL_EXT.1)

FDEAAcPP20E:FCS_VAL_EXT.1.1
The TSF shall perform validation of the [***intermediate key***] using the following methods: [***decrypt a known value using the [intermediate key] as specified in FCS_COP.1(f) and compare it against a stored known value***].

FDEAAcPP20E:FCS_VAL_EXT.1.2
The TSF shall require validation of the BEV prior to forwarding the BEV to the EE.

FDEAAcPP20E:FCS_VAL_EXT.1.3
The TSF shall [***perform a key sanitization of the DEK upon a [configurable number] of consecutive failed validation attempts, require power cycle/reset the TOE after [5] of consecutive failed validation attempts***].

5.1.1.33 Validation (FDEEEcPP20E:FCS_VAL_EXT.1)

FDEEEcPP20E:FCS_VAL_EXT.1.1
The TSF shall perform validation of the BEV using the following method(s): [***decrypt a known value using the [BEV] as specified in FCS_COP.1(f) and compare it against a stored known value***]

FDEEEcPP20E:FCS_VAL_EXT.1.2
The TSF shall require the validation of the BEV prior to allowing access to TSF data after exiting a Compliant power saving state.

FDEEEcPP20E:FCS_VAL_EXT.1.3
The TSF shall [***perform a key sanitization of the DEK upon a [configurable number] of consecutive failed validation attempts, require power cycle/reset the TOE after [5] of consecutive failed validation attempts***]

5.1.2 User data protection (FDP)

5.1.2.1 Protection of Data on Disk (FDEEEcPP20E:FDP_DSK_EXT.1)

FDEEEcPP20E:FDP_DSK_EXT.1.1
The TSF shall perform Full Drive Encryption in accordance with FCS_COP.1(f), such that the drive contains no plaintext protected data.

FDEEEcPP20E:FDP_DSK_EXT.1.2
The TSF shall encrypt all protected data without user intervention.

5.1.3 Security management (FMT)

5.1.3.1 Management of Functions Behavior - per TD0765 (FDEAAcPP20E:FMT_MOF.1)

FDEAAcPP20E:FMT_MOF.1.1
The TSF shall restrict the ability to [modify the behaviour of] the functions [use of Compliant power saving state] to [authorized users].

5.1.3.2 Specification of Management Functions - per TD0767 (FDEAAcPP20E:FMT_SMF.1)

FDEAAcPP20E:FMT_SMF.1.1
The TSF shall be capable of performing the following management functions: [
a) forwarding requests to change the DEK to the EE,
b) forwarding requests to cryptographically erase the DEK to the EE,
c) allowing authorized users to change authorization values or set of authorization values used within the supported authorization method,
d) initiate TOE firmware/software updates,
e) [***no other functions***]
(TD0767 applied)

5.1.3.3 Specification of Management Functions (FDEEEcPP20E:FMT_SMF.1)

FDEEEcPP20E:FMT_SMF.1.1
The TSF shall be capable of performing the following management functions: a) change the DEK, as specified in FCS_CKM.1, when reprovisioning or when commanded, b) erase the DEK, as specified in FCS_CKM.4(a), c) initiate TOE firmware/software updates, d) [***no other functions***].

5.1.3.4 Security Roles (FDEAAcPP20E:FMT_SMR.1)

FDEAAcPP20E:FMT_SMR.1.1
The TSF shall maintain the roles [authorized user].

FDEAAcPP20E:FMT_SMR.1.2
The TSF shall be able to associate users with roles.

5.1.4 Protection of the TSF (FPT)

5.1.4.1 Protection of Key and Key Material - per TD0769 (FDEAAcPP20E:FPT_KYP_EXT.1)

FDEAAcPP20E:FPT_KYP_EXT.1.1
The TSF shall
[***only store keys in non-volatile memory when wrapped, as specified in FCS_COP.1(d), or encrypted, as specified in FCS_COP.1(g) or FCS_COP.1(e)***].
(TD0769 applied)

5.1.4.2 Protection of Key and Key Material - per TD0769 (FDEEEcPP20E:FPT_KYP_EXT.1)

FDEEEcPP20E:FPT_KYP_EXT.1.1

The TSF shall
[***only store keys in non-volatile memory when wrapped, as specified in FCS_COP.1(d), or encrypted, as specified in FCS_COP.1(g) or FCS_COP.1(e)***].
(TD0769 applied)

5.1.4.3 Power Saving States (FDEAAcPP20E:FPT_PWR_EXT.1)

FDEAAcPP20E:FPT_PWR_EXT.1.1

The TSF shall define the following Compliant power saving states: [***G3***].

5.1.4.4 Power Saving States (FDEEEcPP20E:FPT_PWR_EXT.1)

FDEEEcPP20E:FPT_PWR_EXT.1.1

The TSF shall define the following Compliant power saving states: [***G3***].
(TD0464 applied)

5.1.4.5 Timing of Power Saving States (FDEAAcPP20E:FPT_PWR_EXT.2)

FDEAAcPP20E:FPT_PWR_EXT.2.1

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

5.1.4.6 Timing of Power Saving States (FDEEEcPP20E:FPT_PWR_EXT.2)

FDEEEcPP20E:FPT_PWR_EXT.2.1

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

5.1.4.7 TSF Testing (FDEAAcPP20E:FPT_TST_EXT.1)

FDEAAcPP20E:FPT_TST_EXT.1.1

The TSF shall run a suite of the following self-tests [***during initial start-up (on power on), at the conditions before the function is first invoked***] to demonstrate the correct operation of the TSF: [
Power up tests
Integrity Test: Crypto library,
AES CBC 256 bit Encrypt + Decrypt KATs,
AES XTS 256 Encrypt + Decrypt KATs,
SHA-512 KAT,
HMAC-SHA-512 KAT,
DRBG KAT,
DRBG Health Tests.
Conditional tests:
DRBG Health Tests,
Software Upgrade verification: RSA Signature Verification].

5.1.4.8 TSF Testing (FDEEEcPP20E:FPT_TST_EXT.1)

FDEEEcPP20E:FPT_TST_EXT.1.1

The TSF shall run a suite of the following self-tests [***during initial start-up (on power on), at the conditions before the function is first invoked***] to demonstrate the correct operation of the TSF: [
Power up tests
Integrity Test: Crypto library,

AES CBC 256 bit Encrypt + Decrypt KATs,
AES XTS 256 Encrypt + Decrypt KATs,
SHA-256/512 KAT,
RSA sign/verify KAT,
HMAC-SHA-512 KAT,
Conditional tests:
Software Upgrade verification: RSA Signature Verification].

5.1.4.9 Trusted Update (FDEAAcPP20E:FPT_TUD_EXT.1)

FDEAAcPP20E:FPT_TUD_EXT.1.1
Refinement: The TSF shall provide authorized users the ability to query the current version of the TOE [***software***].
FDEAAcPP20E:FPT_TUD_EXT.1.2
Refinement: The TSF shall provide authorized users the ability to initiate updates to TOE [***software***].
FDEAAcPP20E: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.

5.1.4.10 Trusted Update (FDEEEcPP20E:FPT_TUD_EXT.1)

FDEEEcPP20E:FPT_TUD_EXT.1.1
Refinement: The TSF shall provide authorized users the ability to query the current version of the TOE [***software***].
FDEEEcPP20E:FPT_TUD_EXT.1.2
Refinement: The TSF shall provide authorized users the ability to initiate updates to TOE [***software***].
FDEEEcPP20E: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.

5.2 TOE Security Assurance Requirements

The SARs for the TOE are the components as specified in Part 3 of the Common Criteria. Note that the SARs have effectively been refined with the assurance activities explicitly defined in association with both the SFRs and SARs.

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic Functional Specification
AGD: Guidance documents	AGD_OPE.1: Operational User Guidance
	AGD_PRE.1: Preparative Procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM Coverage
ATE: Tests	ATE_IND.1: Independent Testing - Conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability Survey

Table 3 Assurance Components

5.2.1 Development (ADV)

5.2.1.1 Basic Functional Specification (ADV_FSP.1)

ADV_FSP.1.1d
The developer shall provide a functional specification.

ADV_FSP.1.2d

The developer shall provide a tracing from the functional specification to the SFRs.

ADV_FSP.1.1c

The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.2c

The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.3c

The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.4c

The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ADV_FSP.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2e

The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

5.2.2 Guidance documents (AGD)

5.2.2.1 Operational User Guidance (AGD_OPE.1)

AGD_OPE.1.1d

The developer shall provide operational user guidance.

AGD_OPE.1.1c

The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2c

The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3c

The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4c

The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5c

The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AGD_OPE.1.6c

The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7c

The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.2 Preparative Procedures (AGD_PRE.1)

AGD_PRE.1.1d

The developer shall provide the TOE, including its preparative procedures.

AGD_PRE.1.1c

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2c

The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

AGD_PRE.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2e

The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

5.2.3 Life-cycle support (ALC)

5.2.3.1 Labelling of the TOE (ALC_CMC.1)

ALC_CMC.1.1d

The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.1.1c

The TOE shall be labelled with its unique reference.

ALC_CMC.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.2 TOE CM Coverage (ALC_CMS.1)

ALC_CMS.1.1d

The developer shall provide a configuration list for the TOE.

ALC_CMS.1.1c

The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.2c

The configuration list shall uniquely identify the configuration items.

ALC_CMS.1.1e

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Security Target (ASE)

5.2.4.1 Cryptographic operation (Hash Algorithm) (ASE_TSS.1(c))

ASE_TSS.1.1(c)

Refinement: The TOE summary specification shall describe how the TOE meets each SFR, including a proprietary Key Management Description (Appendix E), and [**Entropy Essay**].

5.2.5 Tests (ATE)

5.2.5.1 Independent Testing - Conformance (ATE_IND.1)

ATE_IND.1.1d
The developer shall provide the TOE for testing.

ATE_IND.1.1c
The TOE shall be suitable for testing.

ATE_IND.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.2e
The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.2.6 Vulnerability assessment (AVA)

5.2.6.1 Vulnerability Survey (AVA_VAN.1)

AVA_VAN.1.1d
The developer shall provide the TOE for testing.

AVA_VAN.1.1c
The TOE shall be suitable for testing.

AVA_VAN.1.1e
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.2e
The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3e
The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6. TOE Summary Specification

This chapter describes the security functions:

- Cryptographic support
- User data protection
- Security management
- Protection of the TSF

6.1 Context

6.1.1 Supported Authorization Factors

The TOE supports the following different authorization factors

1. Password – the user supplies a secret password
2. Smartcard – the user inserts their smartcard as well as the PIN needed to authenticate *to the smartcard*.

In all cases, the TOE, after receiving the authorization factor(s), transforms them using PBKDFv2, validates whether the authorization factor(s) is/are correct, and if correct, ultimately decrypts the DEK.

6.1.2 Core TOE Concepts

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

a) Installer. The TOE installer runs from a protected host OS installer that installs all TOE components, configures the drive, and then reboots to initialize user authorization factors and initialize encryption.

b) ESP. EFI System Partition (ESP) is a GUID partition table (GPT) partition with file allocation table (FAT) FAT32 file system located in the UEFI. The system firmware loads files from this folder to boot and load the TOE.

c) GUI. The TOE provides a local graphical user interface (GUI) for unlock via authentication factor(s) and TOE / user management.

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

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

   ii. Login User. Can unlock the drive.

e) Protected OS. The protected OS environment on the drive that is booted after successful TOE authentication.

6.1.3 Key Management

The following sections describe the fundamental key management aspects of the TOE. The Figures below depict the resulting keychains designed with sufficient strength to protect the 256-bit data encryption key (DEK).

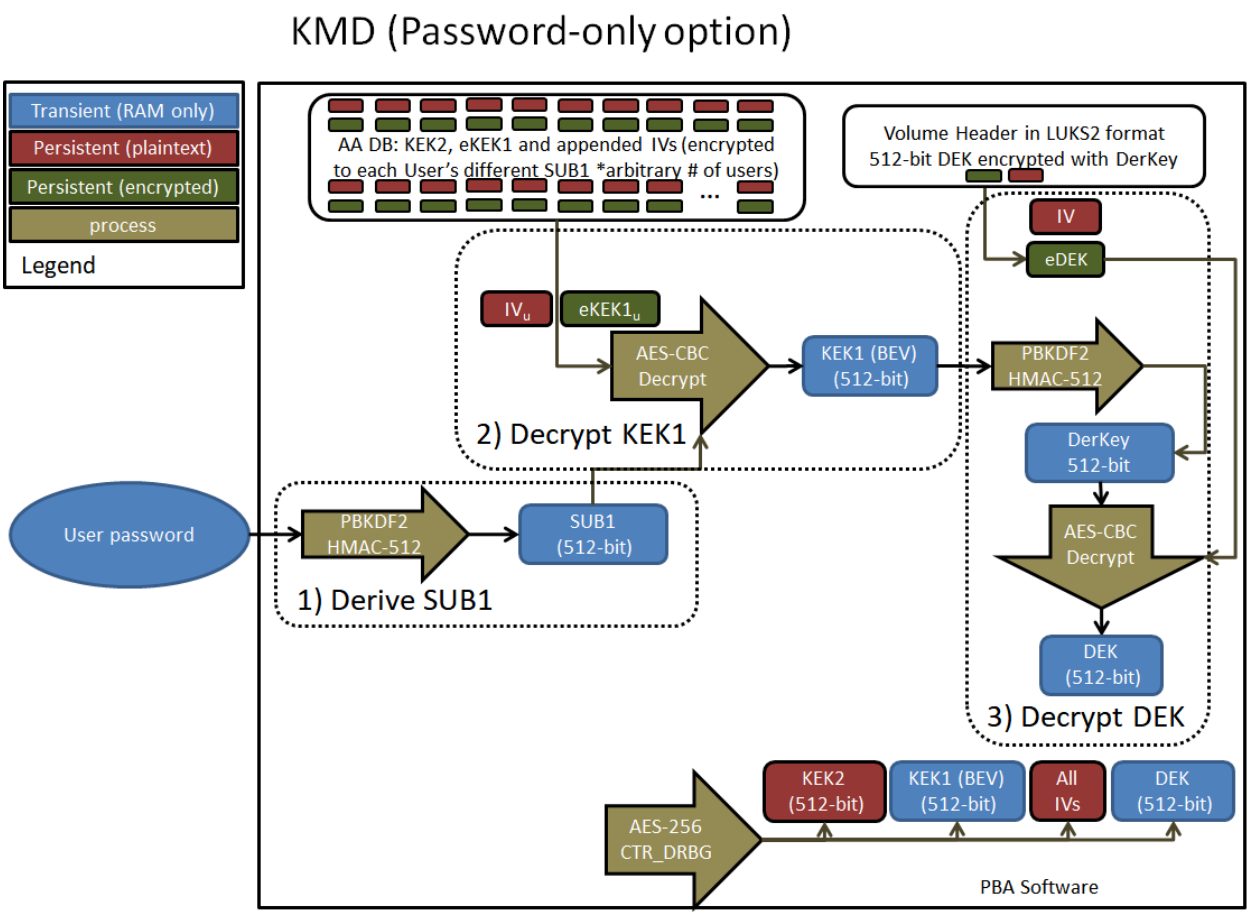

Figure 1 - KMD - Password only

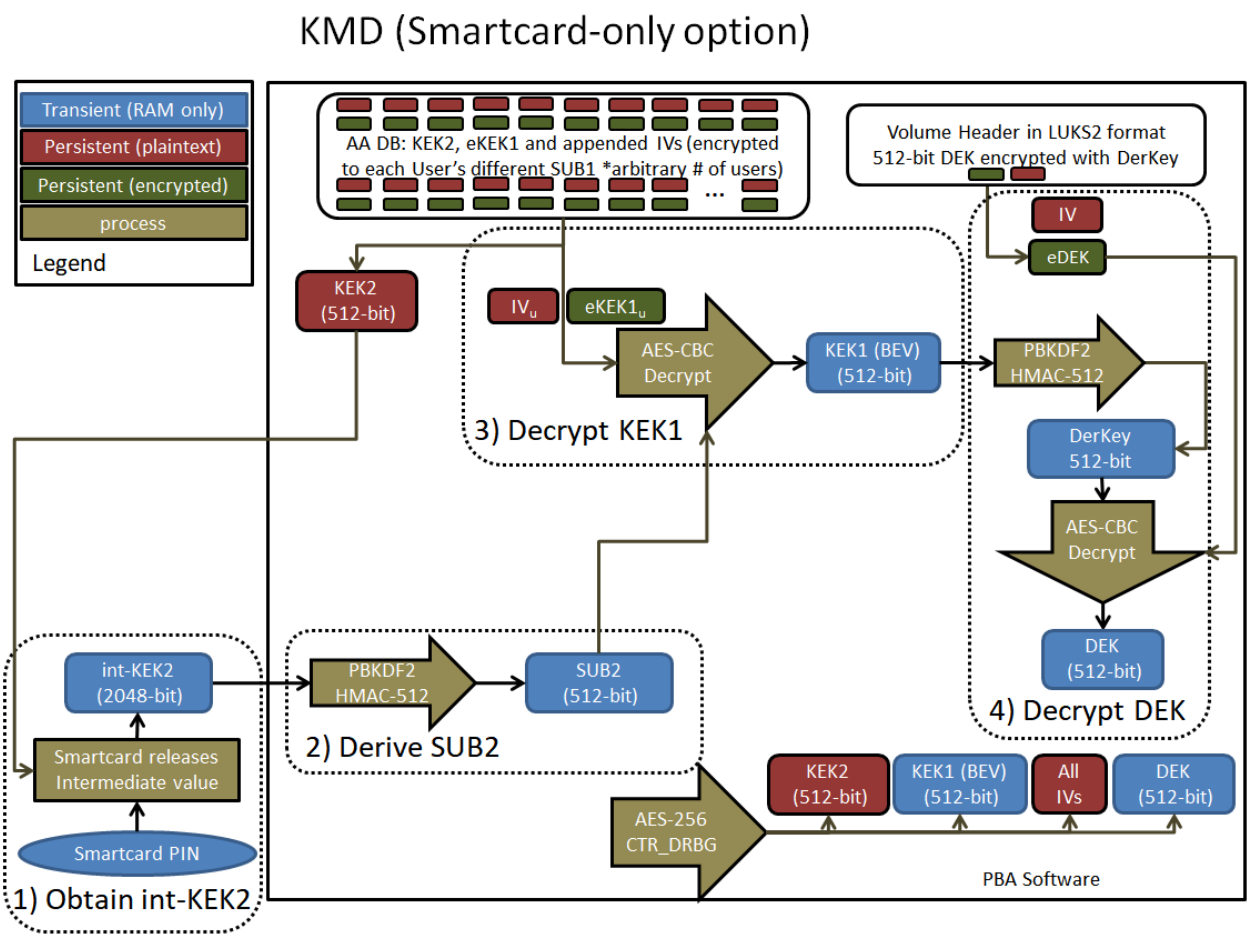

Figure 2 - KMD - Smartcard only

Name	Use	Type	Source	Storage
Password	Authentication factor	Password	Input by user	Volatile memory
KEK2	Authentication factor / pre-cursor	512-bit secret	Generated by DRBG	Non-volatile (plaintext)
int-KEK2	Authentication factor	2048-bit secret	Output from smartcard	Volatile memory
SUB1/2	Used to encrypt KEK1	256-bit AES CBC key (1st 256-bits of submask)	Derived from Authentication factors	Volatile memory
KEK1 (BEV)	Used as BEV sent to EE	512-bit secret value	Generated by DRBG, Decrypted w/ SUBx	Volatile memory & Non-volatile (encrypted)
DerKey	Used to decrypt DEK	256-bit AES CBC key (1st 256-bits of DerKey)	Derived from KEK1/BEV	Volatile memory
DEK	Data Encryption Key	2x256-bit AES XTS keys	Generated by DRBG, Decrypted w/ DerKey	Volatile memory & Non-volatile (encrypted)

Table 4 Key Table

6.1.4 Key Chain

As show in Figures 1, 2, and 3 the TOE uses a chain of submasks and key encryption keys (KEKs) to the DEK, depending on the authentication mechanism:

a) SUB1. 512-bit submask derived from the user's password via password-based key derivation function (PBKDF2).

b) SUB2. 512-bit submask derived (using PBKDF2) from the user's smartcard's protected credential (KEK2).

c) KEK1 (BEV) 512-bit. The TOE randomly generates this key and encrypts a copy using the SUB_x resulting from the user authorization factors in use:

d) KEK2. The TOE generates KEK2, a 512-bit random string, using its DRBG, asks the user's smartcard to encrypt the string, and saves the resulting encrypted blob (which the TOE will later present back to the smartcard during boot/unlock) in its database. The TOE does this for each user employing a smartcard. The user's smartcard possesses an RSA private key that the smartcard uses to encrypt the TOE generated int-KEK2 value. Upon boot, the TOE provides the smartcard both the user's PIN and the RSA encrypted KEK2 value and requests the smartcard decrypt and return the plaintext int-KEK2. Upon receiving back the 256-byte int-KEK2 value, the TOE uses PBKDF2 to derive SUB2.

e) DerKey: The TOE derives this key from the KEK1/BEV value using PBKDF2, and the DerKey encrypts the FDE DEK.

f) DEK: The TOE generates (using its DRBG) two random 256-bit Data Encryption Keys and uses the DerKey to encrypt/decrypt the DEK. The DEK used to access the encrypted drive.

6.1.5 Authentication / Drive Unlock Flow

At a high-level, the basic start-up and authentication flow is as follows:

a. When the TOE starts up, the UEFI portion of the TOE launches the GUI. The user then enters their username and password or inserts a smartcard.

b. Depending on the authentication method:

   1. For password-only, the TOE validates the username against the database.

   2. For smartcard-only, the smartcard PIN is authenticated.

c. The TOE performs PBKDF2 on the password to generate SUB1.

d. The TOE performs PBKDF2 on the smartcard output to generate SUB2.

e. The SUB is then used to encrypt/decrypt the KEK using AES-256-CBC.

f. The TOE passes the plaintext KEK as a BEV value and derives the DerKey from it.

g. The TOE uses the DerKey to decrypt (AES-256-CBC) the DEK.

6.2 Cryptographic support

FDEAAcPP20E:FCS_AFA_EXT.1:

The TOE supports the use of password-only or smartcard-only.

The TOE supports an external smartcard factor that is at least the same bit-length as the DEK (256-bit) – the int-KEK2 value released by the smartcard is 2048-bits in length. The TOE uses a standard PC/SC Interface to communicate with a Smartcard and obtain the smartcard authentication factor.

FDEAAcPP20E:FCS_AFA_EXT.2:

The TOE supports both passwords or a credential for the smartcard. The TOE requires the user (re)present his or her authentication factor(s) after a power cycle (power off followed by power on).

FDEAAcPP20E:FCS_CKM.1(b):

The TOE uses its AES-256 CTR_DRBG to generate KEK1 (BEV value) and DEK (both 512-bit values). The TOE stores the KEK1 encrypted with a key derived from each user's authentication factor (SUB1 or SUB2) and stores the DEK encrypted with the DerKey (a key derived from the KEK1/BEV).

FDEEEcPP20E:FCS_CKM.1(c):

The TOE generates two 256-bit AES-XTS key Data Encryption Keys (together which form the 512-bit DEK) using its AES-256 CTR_DRBG. The TOE accesses its internal DRBG whenever keys or IVs are needed. The TOE does not generate DEKs outside of its TOE boundary nor does it receive a DEK from outside.

FDEAAcPP20E:FCS_CKM.4(a):

The TOE erases cryptographic keys and key material from volatile memory when transitioning to a Compliant power saving state with a single overwrite consisting of zeroes and a second, single, overwrite consisting of ones as specified in FCS_CKM.4(d).

Note: The TSF (not the Operational Environment) is used to destroy keys from volatile memory.

FDEEEcPP20E:FCS_CKM.4(a):

The TOE erases cryptographic keys and key material from volatile memory when the transitioning to the power-off states with a single overwrite consisting of zeros, or alternatively overwrite with a new key value, or finally, with a removal of power to the memory as specified in FCS_CKM_EXT.6 and FCS_CKM_EXT.4(b).

FDEAAcPP20E/FDEEEcPP20E:FCS_CKM.4(d):

For the TOE volatile memory, key destruction is executed by a single overwrite consisting of zeroes followed by a single overwrite of ones, immediately following the operation requiring the key is completed.

For the TOE non-volatile memory, the TOE GUI may be used to forward requests to cryptographically erase the DEK to the encryption engine (EE) by uninstalling the TOE or erasing the entire disk.

Additional details regarding how keys are managed in volatile and non-volatile memory are provided in the Key Management Description (KMD).

FDEAAcPP20E/FDEEEcPP20E:FCS_CKM_EXT.4(a):

At a high level, the TOE keys are no longer needed when power is removed from memory, when the TOE erases the disk, or when the TOE is uninstalled. All intermediate keys are destroyed after their use in the chain. For example, for password-only authentication, the user's SUB1 is destroyed subsequent to the decryption of the KEK1.

FDEAAcPP20E/FDEEEcPP20E:FCS_CKM_EXT.4(b):

Transitioning into the compliant power saving state automatically triggers the destruction of all keys and keying material from volatile memory.

Additional details regarding key destruction when entering a Compliant power saving state are provided in the KMD.

FDEEEcPP20E:FCS_CKM_EXT.6:

The TOE subjects plaintext keys in RAM (volatile memory) to clearing through overwriting the memory location with zeros (when no longer needed) or through removal of power to the memory. Similarly, the TOE provides destruction of persistently stored (non-volatile memory) encrypted keys through a call into the OS to erase the LUKS2 header and pre-boot database.

FDEAAcPP20E:FCS_COP.1:

The TOE UEFI component uses the following its UEFI FIPS OpenSSL 3.5.1 cryptographic algorithms in support of its AA functionality. The TOE's UEFI component was developed in the TianoCore EDK II 202502 UEFI environment.

SFR	Algorithm	NIST Standard	Cert#
FCS_COP.1(a) (Verify)	RSA 4096 w/ SHA-512 Verify	FIPS 186-4, RSA	A7807
FCS_COP.1(b) (Hash)	SHA-512 Hashing	FIPS 180-4	A7807
FCS_COP.1(c) (Keyed Hash)	HMAC-SHA-512	FIPS 198-1 & 180-4	A7807
FCS_COP.1(g) (AES-CBC)	AES-256 CBC Encrypt/Decrypt	FIPS 197	A7807
FCS_COP.1(f) (AES-XTS)	AES-256 XTS Encrypt/Decrypt	FIPS 197	A7807
FCS_CKM.1(b) FCS_CKM.1(c) FCS_RBG_EXT.1 (Random)	AES-256 CTR_DRBG	SP 800-90A	A7807

Table 5 SW-FDE (AA) OpenSSL Cryptographic Algorithms

The TOE filesystem driver also uses its version 1.9 of its OS Driver for Encryption Engine (EE) functionality.

SFR	Algorithm	NIST Standard	Cert#
FCS_COP.1(a) (Verify)	RSA 4096 w/ SHA-256 Verify	FIPS 186-4, RSA	A7808
FCS_COP.1(b) (Hash)	SHA-256/512 Hashing	FIPS 180-4	A7808
FCS_COP.1(c) (Keyed Hash)	HMAC-SHA-512	FIPS 198-1 & 180-4	A7808
FCS_COP.1(g) (AES-CBC)	AES-256 CBC Encrypt/Decrypt	FIPS 197	A7808
FCS_COP.1(f) (AES-XTS)	AES-256 XTS Encrypt/Decrypt	FIPS 197	A7808

Table 6 SW-FDE (EE) OS Driver Cryptographic Algorithms

FDEAAcPP20E/FDEEEcPP20E:FCS_COP.1(a):

The TOE performs signature verification using RSA 4096 certificates and specifically,

- Cigent signs the trusted UEFI component of the TOE with an RSA 4096-bit key and SHA-512 message hashing
- Cigent signs the Windows installer, kernel volume driver, service, and UI with RSA 4096 w/ SHA-256 digest
- Cigent's Windows kernel volume driver is signed by Microsoft using RSA 4096 w/ SHA-256 digest.

FDEAAcPP20E/FDEEEcPP20E:FCS_COP.1(b):

The TOE makes use of SHA-512 for the following:

a) Digital signature verification

b) PBKDF (using HMAC) of authorization factors and of the BEV

The TOE makes use of SHA-256 for the following:

a) Digital signature verification

FDEAAcPP20E/FDEEEcPP20E:FCS_COP.1(c):

The TOE implements HMAC-SHA-512 with the following characteristics:

a) Key length. 512 bits.

b) Block size. 1024 bits.

c) MAC length. 512 bits.

FDEAAcPP20E/FDEEEcPP20E:FCS_COP.1(f):

The TOE uses AES-256 CBC for key encryption decryption (of both KEK1 and the DEK). The TOE uses AES-256 XTS (as specified in IEEE 1619) to encrypt drive data. All keys are 256-bits in size; however, XTS uses two, 256-bit keys, totaling 512-bits. Additionally, where the TOE derives a 512-bit submask (from PBKDF2 w/ HMAC-SHA-512), the TOE uses the first 256-bits as the AES CBC key.

FDEAAcPP20E/FDEEEcPP20E:FCS_COP.1(g):

The TOE uses AES-256 bit CBC keys to protect persistently stored keys in the keychain (KEK1 and the DEK).

FDEAAcPP20E/FDEEEcPP20E:FCS_KDF_EXT.1:

The TOE conditions passwords and smartcard protected submasks via PBKDF2 using HMAC-SHA-512 with 120,842 iterations, resulting in a 512-bit submask in accordance with National Institute of Standards and Technology (NIST) special publication (SP) 800-132. For smartcard authentication, the TOE accepts an RNG generated submask in accordance with NIST SP 800-108.

The TOE also uses PBKDFv2 to HMAC-SHA-512 (with 124,432 iterations) condition the imported KEK1/BEV submask and thereby derive the DerKey.

FDEAAcPP20E:FCS_KYC_EXT.1:

The TOE supports conditioning of the passphrase and smartcard authorization factors via PBKDFv2 to derive the SUBx submasks (each submask has a size of 512 bits). The TOE uses the first 256-bits of the submask to encrypt/decrypt the KEK1/BEV value. Additional details on the TOE key chain are provided in the KMD.

FDEEEcPP20E:FCS_KYC_EXT.2:

The TOE accepts a BEV of 512 bits from the AA, maintaining a chain of intermediate keys originating from the BEV to the DEK and using the following methods: the TOE uses PBKDFv2 to derive the DerKey which the TOE uses to for key encryption/decryption of the DEK.

The TOE maintains an effective strength of 256 bits for symmetric keys

FDEAAcPP20E:FCS_PCC_EXT.1:

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

a) Minimum Length (8 – 128)

b) Require at least one uppercase

c) Require at least one lowercase

d) Require at least one numeric

e) Require at least one of the following special characters: ("~", "!", "@", "#", "$", "^", "&", "*", "(", ")", "_", "-", "+", "=", "[", "]", ":", "<", ">", ".")

Passwords are conditioned via PBKDF2 using HMAC-SHA-512 with 120,842 iterations, resulting in a 512-bit submask in accordance with NIST SP 800-132.

FDEAAcPP20E:FCS_RBG_EXT.1:

The TOE uses a software-based random bit generator (AES-256 CTR_DRBG) that complies with NIST SP 800-90A for all cryptographic operations. The DRBG is seeded by a single software-based entropy/noise source from a Jitterentropy entropy source. All entropy is extracted, processed, and accumulated by OpenSSL.

The Jitter software noise source provides full entropy output, thus a 256-bit seed contains 256 bits of entropy.

FDEAAcPP20E/FDEEEcPP20E:FCS_SNI_EXT.1:

Salts and IVs are generated using the RBG as described in FCS_RBG_EXT.1, which ensures that they are unique.

The tweak values used for XTS are non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer (based upon the LBA of the drive sector being encrypted), thus ensuring each is unique.

The TOE does not make use of nonces.

FDEAAcPP20E/FDEEEcPP20E:FCS_VAL_EXT.1:

The TOE accepts authorization factors from users and validates them through the entire keychain. The keychain results in decryption of the DEK and the TOE's UEFI component subjects the DEK to PBKDF2 derivation (using a slightly different number of iterations [100,207] than used to derive the SUB keys) and compares that to the stored value. If validation of the authorization factors fails, the TOE rejects the user's authorization attempt and increments the number of consecutive failed authentication attempts. When the number of failed attempts exceeds a configurable value (default of 5), the TOE forces a system restart. Additionally, one can optionally configure the TOE to erase the drive after exceeding the failed attempt limit.

6.3 User data protection

FDEEEcPP20E:FDP_DSK_EXT.1:

An admin installs the TOE into an unencrypted system, and once installed the TOE's UEFI component resides in the plaintext UEFI area. The TOE encrypts the entire protected OS drive without user intervention using AES:XTS (as described in Section 6.2). The data encryption engine installed into the Windows 11 Operating System and consists of Windows file system filter driver (providing the volume encryption), a Windows service (to interact with the driver), and a system tray (providing a simple UI for administration). The TOE encrypts the entire Windows system drive (as only the UEFI and the TOE's UEFI pre-boot application lie in plaintext on the drive). The initial TOE configuration steps needed to ensure the TOE encrypts the storage device entirely when a user or administrator first provisions the product can be found in section 6.1.2a) and the Admin Guide.

6.4 Security management

FDEAAcPP20E:FMT_MOF.1:

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

FDEAAcPP20E:FMT_SMF.1:

The TOE GUI may be used to forward requests to cryptographically erase the DEK to the encryption engine (EE) via the GUI by uninstalling the TOE or erasing the entire disk.

Note: Changing the DEK is the same functionality as cryptographically erasing the DEK.

The TOE GUI may be used to configure the active authorization factor (either password or smartcard).

TOE updates may be performed by booting the host system and install an MSI installed to update the TOE software. An admin user must authenticate and choose to install the update.

FDEEEcPP20E:FMT_SMF.1:

The TOE is 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 update (initiated from the protected OS).

The TOE changes a DEK when re-provisioning or when commanded.

Software updates to the TOE are initiated through the protected host operating system.

FDEAAcPP20E:FMT_SMR.1:

The TOE restricts access to authorized users.

6.5 Protection of the TSF

FDEAAcPP20E/FDEEEcPP20E:FPT_KYP_EXT.1:

Keys are protected as described in the KMD. The TOE stores all keys encrypted with AES–CBC as per FCS_COP.1(g) and stored in the TOE's database (for the UEFI component) or in the LUKS2 drive header (for the filesystem driver).

FDEAAcPP20E/FDEEEcPP20E:FPT_PWR_EXT.1:

The TOE represents software executing on host computer and as such, supports only a single Compliant power saving state:

a) G3. In this state, the computing system is completely off and it does not consume any power. The system returns to the working state only after a complete reboot and hence the UEFI Pre-boot component of the TOE will be invoked/executed for authentication/authorization.

FDEAAcPP20E/FDEEEcPP20E:FPT_PWR_EXT.2:

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

FDEAAcPP20E:FPT_TST_EXT.1:

The TOE's UEFI component runs the following Power on Self-Tests:

- Power on Self-Tests:
  - AES CBC Encrypt/Decrypt KATs,
  - AES XTS Encrypt/Decrypt KATs,
  - SHA-512 KAT,
  - HMAC-SHA-512 KAT,
  - DRBG KAT,
  - DRBG Health Tests (consistent with the requirements in section 11.3 of NIST SP 800-90A)
- Conditional tests:
  - DRBG Health Tests (consistent with the requirements in section 11.3 of NIST SP 800-90A),
  - Software Upgrade verification: RSA Signature Verification.

FDEEEcPP20E:FPT_TST_EXT.1:

The TOE's filesystem driver runs the following Power on Self-Tests:

- Power on Self-Tests:
  - AES CBC Encrypt/Decrypt KATs,
  - AES XTS Encrypt/Decrypt KATs,
  - SHA-256/512 KAT,
  - HMAC-SHA-512 KAT.

FDEAAcPP20E/FDEEEcPP20E:FPT_TUD_EXT.1:

Update files are digitally signed (RSA 4096 per FCS_COP.1(a)) by Cigent and TOE will verify the signature on the image update file (using its cryptographic library in conjunction with the vendor's signing key, stored internally, to verify the signature) before installing it, and will reject any update with an invalid signature. The TOE does not support automatic update credentials. Only authorized administrators may manually perform the update process from within the protected (host) operating system.