www.GossamerSec.com

Assurance Activity Report (FDEAACPP20E/FDEEECPP20E) for Curtiss-Wright Defense Solutions Data Transport System 1-Slot Software Encryption Layer V3.01.00

Version 0.3
03/21/2023

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Catonsville, MD 21228

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Document: AAR-VID11355

Revision History

Revision	Date	Authors	Summary
Version 0.1	02/22/2023	Gossamer	Initial draft
Version 0.2	03/14/2023	Gossamer	Addressed ECR comments
Version 0.3	03/21/2023	Gossamer	Updated ST and User Guide references

The TOE Evaluation was Sponsored by:

Curtiss-Wright Defense Solutions
2600 Paramount Pl #200
Fairborn, OH 45324

Evaluation Personnel:

- John Messiha

Common Criteria Versions:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

Common Evaluation Methodology Versions:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017

Table of Contents

1. Introduction

This document presents evaluations results of the Curtiss-Wright DTS1 (SW Layer) FDEAAcPP20E/FDEEEcPP20E evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 CAVP Certificates

The TOE has the following CAVP certificates. The TOE has a single platform and the certificates were obtained on that platform.

The TOE uses its OpenSSL library (version 3.1.0 for CentOS 7.9) when verifying ECDSA P-384 w/ SHA-384 trusted update signatures.

SFR	Algorithm	NIST Standard	Cert#
FCS_COP.1(a) (Verify)	ECDSA P-384 w/ SHA-384 Verify	FIPS 186-4, ECDSA	A3313
FCS_COP.1(b) (Hash)	SHA-384 Hashing	FIPS 180-4	A3313

Table 1 OpenSSL Cryptographic Algorithms

The TOE uses its kernel cryptography (version 3.1.0 for CentOS 7.9) when doing AES-256 CBC ESSIV:SHA-256 data encryption/decryption.

SFR	Algorithm	NIST Standard	Cert#
FCS_COP.1(b) (Hash)	SHA-256 Hashing	FIPS 180-4	A3312
FCS_COP.1(f) (AES)	AES-256 XTS Encrypt/Decrypt	FIPS 197	A3312

Table 2 kernel Cryptographic Algorithms

The TOE uses its libgcrypt library (version 3.1.0 for CentOS 7.9) when doing key derivation and key management operations.

SFR	Algorithm	NIST Standard	Cert#
FCS_COP.1(b) (Hash)	SHA-256 Hashing	FIPS 180-4	A3311
FCS_COP.1(c) (Keyed Hash)	HMAC-SHA-256	FIPS 198-1 & 180-4	A3311
FCS_COP.1(g) (AES)	AES-256 CBC Encrypt/Decrypt	FIPS 197	A3311
FCS_RBG_EXT.1 (Random)	SHA-256 HMAC_DRBG	SP 800-90A	A3311

Table 3 libgcrypt Cryptographic Algorithms

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profile and Extended Packages. This section also describes the findings for each activity.

The evidence identified below was used to perform these Assurance Activities.

- Curtiss-Wright Defense Solutions Curtiss-Wright Defense Solutions Data Transport System 1-Slot Software Encryption Layer v3.01.00 Security Target, Version 1.7, 03/21/2023 (ST)
- Curtiss-Wright DTS1 CSfC 1-Slot Data Transport System (CSfC) User Guide, DOC0099-000-B4 (User Guide)

2.1 Cryptographic support (FCS)

2.1.1 Authorization Factor Acquisition (FDEAAcPP20E:FCS_AFA_EXT.1)

2.1.1.1 FDEAAcPP20E:FCS_AFA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall first examine the TSS to ensure that the authorization factors specified in the ST are described. For password-based factors the examination of the TSS section is performed as part of FCS_PCC_EXT.1 Evaluation Activities. Additionally in this case, the evaluator shall verify that the operational guidance discusses the characteristics of external authorization factors (e.g., how the authorization factor must be generated; format(s) or standards that the authorization factor must meet) that are able to be used by the TOE.

If other authorization factors are specified, then for each factor, the TSS specifies how the factors are input into the TOE.

KMD

The evaluator shall examine the Key Management Description to confirm that the initial authorization factors (submasks) directly contribute to the unwrapping of the BEV.

The evaluator shall verify the KMD describes how a submask is produced from the authorization factor (including any associated standards to which this process might conform), and verification is performed to ensure the length of the submask meets the required size (as specified in this requirement).

Section 6.1 of the ST states the TOE supports a password authorization factor. See FCS_PCC_EXT.1 for an assessment of the password. No other authorization factors are identified.

KMD - Section 7 explains the TOE uses PBKDFv2 to transform the operator's password into a 256-bit BEV, and then uses that BEV to AES decrypt the DEKs stored on the drive.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance includes instructions for all of the authorization factors. The AGD will discuss the characteristics of external authorization factors (e.g., how the authorization factor is generated; format(s) or standards that the authorization factor must meet, configuration of the TPM device used) that are able to be used by the TOE.

Section 5.1.2 of the User Guides discusses the password. This is the only authorization factor supported by the TOE. See FCS_PCC_EXT.1 for an assessment of the password.

Component Testing Assurance Activities: The password authorization factor is tested in FCS_PCC_EXT.1.

The evaluator shall also perform the following tests:

Test 1 [conditional]: If there is more than one authorization factor, ensure that failure to supply a required authorization factor does not result in access to the decrypted plaintext data.

Test 1: Not applicable as password is the only authorization factor.

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

2.1.2.1 FDEAAcPP20E:FCS_AFA_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS for a description of authorization factors and which of the factors are used to gain access to user data after the TOE entered a Compliant power

saving state. The TSS is inspected to ensure it describes that each authorization factor satisfies the requirements of FCS_AFA_EXT.1.1.

Section 6.1 of the ST states the TOE does not have any power-saving states beyond power-on and power-off. After transitioning from the power-off to the power-on state, the user must authenticate with a password before the TOE will allow data to be read from or written to the drive. The password is identified in FCS_AFA_EXT.1.1 and no other authorization factors exist.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation for a description of authorization factors used to access plaintext data when resuming from a Compliant power saving state.

Section 5.4 Software Encryption of the User Guide explains how to enter a password to use the software encryption layer. This step is required after the device has been powered off.

Component Testing Assurance Activities: The evaluator shall perform the following test:

- Enter the TOE into a Compliant power saving state

- Force the TOE to resume from a Compliant power saving state

- Release an invalid authorization factor and verify that access to decrypted plaintext data is denied

- Release a valid authorization factor and verify that access to decrypted plaintext data is granted.

Test – The evaluator first rebooted the TOE and logged on. To access the software encryption, the administrator must log into the software layer.

The results showed the filesystem was not available before login. The evaluator then attempted to login with an incorrect password. The login attempt was rejected and access was not granted.

Following that attempt, the evaluator logged on with a good password, was granted access and the evaluator could mount the encrypted filesystem.

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

2.1.3.1 FDEEEcPP20E:FCS_CKM.1.1(c)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes how the TOE obtains a DEK (either generating the DEK or receiving from the environment).

If the TOE generates a DEK, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. If the DEK is generated outside of the TOE, the evaluator checks to ensure that for each platform identified in the TOE the TSS, it describes the interface used by the TOE to invoke this functionality. The evaluator uses the description of the interface between the RBG and the TOE to determine that it requests a key greater than or equal to the required key sizes.

KMD

If the TOE received the DEK from outside the host platform, then the evaluator shall verify that the KMD describes how the TOE unwraps the DEK.

If the TOE received the DEK from outside the host platform, then the evaluator shall examine the TSS to determine that the DEK is sent wrapped using the appropriate encryption algorithm.

Section 6.1 of the ST explains the TOE can generate 256-bit DEKs onboard using its SHA-256 HMAC_DRBG. Because the DRBG has a security strength of 256 bits, the DEKs generated are sufficient for the TOE's 256-bit AES data encryption/decryption.

KMD – All DEKs are generated internally.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall configure the TOE to ensure the functionality of all selections.

The test is met by configuring the TOE in the evaluated configuration using the instrumented build. When the evaluator follows the setup instructions, at the step where it's supposed to generate a DEK, we see the instrumented build dump a DEK to the console. This demonstrates the TOE can be configured to generate a DEK. Examples of the DEK being dumped to the console are found are in FDEAAcPP20E:FCS_CKM.4(d).

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

2.1.4.1 FDEAAcPP20E:FCS_CKM.4.1(a)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS provides a high level description of how keys stored in volatile memory are destroyed. The valuator to verify that TSS outlines:

- if and when the TSF or the Operational Environment is used to destroy keys from volatile memory;

- if and how memory locations for (temporary) keys are tracked;

- details of the interface used for key erasure when relying on the OE for memory clearing.

KMD

The evaluator shall check to ensure the KMD lists each type of key, its origin, possible memory locations in volatile memory.

Section 6.1 of the ST explains that the TOE has 4GB of RAM, and this serves as the working memory in which the TOE temporarily stores working copies of key material (for example, the Derived Key [DerKey], which is derived from the user's password and salt using PBKDFv2 and the DEKs currently in use (if any). The TOE clears keys from memory by a removal of power.

Additionally, the TOE stores encrypted DEKs in a header for the encrypted drive or drive partitions. The TOE clears these keys by an internal call using the CRYPT_WIPE_RANDOM pattern, which draws random data from the TOE's HMAC_DRBG.

KMD - The KMD includes a table that identifies each key, its derivation, its storage location and when it is destroyed.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation if the TOE depends on the Operational Environment for memory clearing and how that is achieved.

The TOE does not depend on its operational environment for memory clearing.

Component Testing Assurance Activities: There are no test evaluation activities for this SFR.

There are no test evaluation activities for this SFR.

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

2.1.5.1 FDEEEcPP20E:FCS_CKM.4.1(a)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: TSS

The evaluator shall verify the TSS provides a high level description of how keys stored in volatile memory are destroyed. The valuator to verify that TSS outlines:

- if and when the TSF or the Operational Environment is used to destroy keys from volatile memory;

- if and how memory locations for (temporary) keys are tracked;

- details of the interface used for key erasure when relying on the OE for memory clearing.

KMD

The evaluator shall check to ensure the KMD lists each type of key, its origin, possible memory locations in volatile memory.

See the discussion in FDEAAcPP20:FCS_CKM.4(a).

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation if the TOE depends on the Operational Environment for memory clearing and how that is achieved.

The TOE does not depend on its operational environment for memory clearing.

Component Testing Assurance Activities: None Defined

2.1.6 Cryptographic Key Destruction (Software TOE, 3rd Party Storage) (FDEAAcPP20E:FCS_CKM.4(d))

2.1.6.1 FDEAAcPP20E:FCS_CKM.4.1(d)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator shall check to ensure the TSS lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) and description in the TSS.

The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement. If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

Section 6.1 of the ST explains that the TOE has 4GB of RAM, and this serves as the working memory in which the TOE temporarily stores working copies of key material (for example, the Derived Key [DerKey], which is derived from the user's password and salt using PBKDFv2 and the DEKs currently in use (if any). The TOE clears keys from memory by a removal of power.

Additionally, the TOE stores encrypted DEKs in a header for the encrypted drive or drive partitions. The TOE clears these keys by an internal call using the CRYPT_WIPE_RANDOM pattern, which draws random data from the TOE's HMAC_DRBG.

Component Guidance Assurance Activities: There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, it is assumed the drive supports the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. It is assumed the operating system and file system of the OE support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion.

It is assumed that if a RAID array is being used, only set-ups that support TRIM are utilized. It is assumed if the drive is connected via PCI-Express, the operating system supports TRIM over that channel. It is assumed the drive is healthy and contains minimal corrupted data and will be end of life before a significant amount of damage to drive health occurs, it is assumed there is a risk small amounts of potentially recoverable data may remain in damaged areas of the drive.

Finally, it is assumed the keys are not stored using a method that would be inaccessible to TRIM, such as being contained in a file less than 982 bytes which would be completely contained in the master file table.

The TOE is able to perform memory clearing as it has access to the hardware resources needed. The User Guide, in Section 5.4 does provide a warning that states that in order for the memory to be clear, an SSH connection must be used when setting keys and passwords.

Component Testing Assurance Activities: Test 1: Applied to each key held as plaintext in volatile memory and subject to destruction by overwrite by the TOE (whether or not the plaintext value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Cause the TOE to stop the execution but not exit.

5. Cause the TOE to dump the entire memory of the TOE into a binary file.

6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

7. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a miniscule chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is found the test fails.

The following tests apply only to selection a), since the TOE in this instance has more visibility into what is happening within the underlying platform (e.g., a logical view of the media). In selection b), the TOE has no visibility into the inner workings and completely relies on the underlying platform, so there is no reason to test the TOE beyond test 1.

For selection a), the following tests are used to determine the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.

Test 2: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.

5. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for Use Case 1 test 1 above), and if a fragment is found in the repeated test then the test fails.

Test 3: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:

1. Record the logical storage location of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

Test 1 – The evaluator received a developer build from Curtiss Wright. The evaluator then used that build to run a series of memory dump tests that dumped the memory on the device. The evaluator took the memory dumps and searched those dumps with a hex search tool to search for known keys. The evaluator was unable to find any of the keys in the dump files.

Test 2 – The evaluator received a developer build from Curtiss Wright. The evaluator then used that build the run a series of flash dump tests that dumped the searched flash on the device. The evaluator searched the flash with a hex search tool to search for known keys. The evaluator then cleared the keys and was unable to locate them on subsequent searches.

Test 3 – This was tested as part test case 1 where the passphrase was searched. In this test case, the evaluator demonstrated the memory was overwritten with zeroes.

2.1.7 Cryptographic Key Destruction (Software TOE, 3rd Party Storage) (FDEEEcPP20E:FCS_CKM.4(d))

2.1.7.1 FDEEEcPP20E:FCS_CKM.4.1(d)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator shall check to ensure the TSS lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) and description in the TSS.

The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement. If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

See the description in FDEAAcPP20E:FCS_CKM.4(d)

Component Guidance Assurance Activities: There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or

circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, it is assumed the drive supports the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. It is assumed the operating system and file system of the OE support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion.

It is assumed that if a RAID array is being used, only set-ups that support TRIM are utilized. It is assumed if the drive is connected via PCI-Express, the operating system supports TRIM over that channel. It is assumed the drive is healthy and contains minimal corrupted data and will be end of life before a significant amount of damage to drive health occurs, it is assumed there is a risk small amounts of potentially recoverable data may remain in damaged areas of the drive.

Finally, it is assumed the keys are not stored using a method that would be inaccessible to TRIM, such as being contained in a file less than 982 bytes which would be completely contained in the master file table.

See the description in FDEAAcPP20E:FCS_CKM.4(d)

Component Testing Assurance Activities: Test 1: Applied to each key held as plaintext in volatile memory and subject to destruction by overwrite by the TOE (whether or not the plaintext value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Cause the TOE to stop the execution but not exit.

5. Cause the TOE to dump the entire memory of the TOE into a binary file.

6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

7. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a miniscule chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is found the test fails.

The following tests apply only to selection a), since the TOE in this instance has more visibility into what is happening within the underlying platform (e.g., a logical view of the media). In selection b), the TOE has no visibility into the inner workings and completely relies on the underlying platform, so there is no reason to test the TOE beyond test 1.

For selection a), the following tests are used to determine the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.

Test 2: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.

5. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for Use Case 1 test 1 above), and if a fragment is found in the repeated test then the test fails.

Test 3: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:

1. Record the logical storage location of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

See the description in FDEAAcPP20E:FCS_CKM.4(d).

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

2.1.8.1 FDEAAcPP20E:FCS_CKM_EXT.4.1(a)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when then should be expected to be destroyed.

KMD

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside and when the keys and key material are no longer needed.

The evaluator shall verify the KMD includes a key lifecycle, that includes a description where key material reside, how the key material is used, how it is determined that keys and key material are no longer needed, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM.4(a) for the destruction.

Section 6.1 of the ST explains that the TOE has 4GB of RAM, and this serves as the working memory in which the TOE temporarily stores working copies of key material (for example, the Dervied Key [DerKey], which is derived from the user's password and salt using PBKDFv2 and the DEKs currently in use (if any). The TOE actively clears (overwriting userspace keys with zeros, using its crypt_memzero() function and clearing keys [DEKs] from kernel memory using the secure data flag for device-mapper that forces the kernel to wipe all ioctl buffers with possible key data) these values when no longer needed. The TOE clears the DerKey from memory immediately after the operation for which its needed, while DEKs will be held in kernel memory while the drive data is unlocked. If the user logs out, then the TOE will clear any in-use DEKs from kernel memory.

Additionally, the TOE stores encrypted DEKs in a partition header for the each encrypted drive or drive partitions. The TOE clears these keys by through an internal call using the CRYPT_WIPE_RANDOM pattern, which draws random data from the TOE's HMAC_DRBG.

KMD - The KMD includes a table that identifies each key, its derivation, its storage location and when it is destroyed. The material in the KMD matches the material in the TSS.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.9.1 FDEEEcPP20E:FCS_CKM_EXT.4.1(a)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when then should be expected to be destroyed.

KMD

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside and when the keys and key material are no longer needed.

The evaluator shall verify the KMD includes a key lifecycle, that includes a description where key material reside, how the key material is used, how it is determined that keys and key material are no longer needed, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM.4(a) for the destruction.

See the description for FDEAAcPP20:FCS_CKM_EXT.4(a).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.10.1 FDEAAcPP20E:FCS_CKM_EXT.4.1(B)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS provides a description of what keys and key material are destroyed when entering any Compliant power saving state.

KMD

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside.

The evaluator shall verify the KMD includes a key lifecycle that includes a description where key material resides, how the key material is used, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM.4(d) for the destruction. (TD0345 applied)

The TOE has no Compliant power saving states other than power on and off (G3). See FDEAAcPP20:FCS_CKM_EXT.4(a).1 for a description of key destruction and lifecycle.

Component Guidance Assurance Activities: The evaluator shall validate that guidance documentation contains clear warnings and information on conditions in which the TOE may end up in a non-Compliant power saving state indistinguishable from a Compliant power saving state. In that case it must contain mitigation instructions on what to do in such scenarios.

The User Guide identifies the shutdown state in the CLI section 12.3. There are no other states (including sleep) so the administrator should be clear about whether the device is powered off or on. Further, the Startup section of the User Guide identifies the LED lights that are on when the TOE is powered on.

Component Testing Assurance Activities: None Defined

2.1.11 CRYPTOGRAPHIC KEY AND KEY MATERIAL DESTRUCTION (POWER MANAGEMENT) (FDEEEcPP20E:FCS_CKM_EXT.4(B))

2.1.11.1 FDEEEcPP20E:FCS_CKM_EXT.4.1(B)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS provides a description of what keys and key material are destroyed when entering any Compliant power saving state.

KMD

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside.

The evaluator shall verify the KMD includes a key lifecycle that includes a description where key material resides, how the key material is used, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM_EXT.6 for the destruction. (TD0345 applied)

The TOE has no Compliant power saving states other than power on and off (G3). See FDEAAcPP20:FCS_CKM_EXT.4(a).1 for a description of key destruction and lifecycle.

Component Guidance Assurance Activities: The evaluator shall validate that guidance documentation contains clear warnings and information on conditions in which the TOE may end up in a non-Compliant power saving state indistinguishable from a Compliant power saving state. In that case it must contain mitigation instructions on what to do in such scenarios.

See FDEAAcPP20E:FCS_CKM_EXT.4(b)

Component Testing Assurance Activities: None Defined

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

2.1.12.1 FDEEEcPP20E:FCS_CKM_EXT.6.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TOE's keychain in the TSS/KMD and verify all keys subject to destruction are destroyed according to one of the specified methods.

The TSS identifies how each key is destroyed. See the discussion in FCS_CKM.4(d)

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.13.1 FDEAAcPP20E:FCS_COP.1.1(a)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes the overall flow of the signature verification. This should at least include identification of the format and general location (e.g., 'firmware on the hard drive device' rather than 'memory location 0x00007A4B') of the data to be used in verifying the digital signature; how the data received from the operational environment are brought on to the device; and any processing that is performed that is not part of the digital signature algorithm (for instance, checking of certificate revocation lists).

Section 6.1 of the ST explains the overall flow of the signature verification. The TOE utilizes ECDSA P-384 w/ SHA-384 signatures to verify the authenticity of firmware updates. Upon receiving a candidate update and the accompanying signature file, the TOE uses an embedded public key (see FPT_TUD_EXT.1 below for the location) to verify the ECDSA signature against the received image. The verification uses SHA-384 and follows the FIPS 186-4 ECDSA format.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Each section below contains the tests the evaluators must perform for each type of digital signature scheme. Based on the assignments and selections in the requirement, the evaluators choose the specific activities that correspond to those selections.

It should be noted that for the schemes given below, there are no key generation/domain parameter generation testing requirements. This is because it is not anticipated that this functionality would be needed in the end device, since the functionality is limited to checking digital signatures in delivered updates. This means that the domain parameters should have already been generated and encapsulated in the hard drive firmware or on-board non-volatile storage. If key generation/domain parameter generation is required, the evaluation and validation scheme must be consulted to ensure the correct specification of the required evaluation activities and any additional components.

The following tests are conditional based upon the selections made within the SFR.

The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's authentic and unauthentic signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

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

2.1.14.1 FDEEEcPP20E:FCS_COP.1.1(a)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes the overall flow of the signature verification. This should at least include identification of the format and general location (e.g., 'firmware on the hard drive device' rather than 'memory location 0x00007A4B') of the data to be used in verifying the digital signature; how the data received from the operational environment are brought on to the device; and

any processing that is performed that is not part of the digital signature algorithm (for instance, checking of certificate revocation lists).

See the discussion in FDEAAcPP20E:FCS_COP.1(a)

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Each section below contains the tests the evaluators must perform for each type of digital signature scheme. Based on the assignments and selections in the requirement, the evaluators choose the specific activities that correspond to those selections.

It should be noted that for the schemes given below, there are no key generation/domain parameter generation testing requirements. This is because it is not anticipated that this functionality would be needed in the end device, since the functionality is limited to checking digital signatures in delivered updates. This means that the domain parameters should have already been generated and encapsulated in the hard drive firmware or on-board non-volatile storage. If key generation/domain parameter generation is required, the evaluation and validation scheme must be consulted to ensure the correct specification of the required evaluation activities and any additional components.

The following tests are conditional based upon the selections made within the SFR.

The following tests may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's authentic and unauthentic signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

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

2.1.15.1 FDEAAcPP20E:FCS_COP.1.1(b)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Section 6.1 of the TSS explains that the TOE's kernel, gcrypt, and OpenSSL libraries provide the SHA-256 and SHA-384 algorithms and use those algorithms as part of ESSIV:SHA-256 IV generation, PBKDFv2 password-based key derivation, and trusted update signature verification respectively

Component Guidance Assurance Activities: The evaluator checks the operational guidance documents to determine that any system configuration necessary to enable required hash size functionality is provided.

No configuration is necessary for setting the required hash.

Component Testing Assurance Activities: The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented test mode.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this cPP.

Short Messages Test Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 99*i, where 1 <= i <= m. For SHA-384 and SHA-512, the length of the i-th message is 1024 + 99*i, where 1 <= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 8*99*i, where 1 <= i <= m/8. For SHA-384 and SHA-512, the length of the i-th message is 1024 + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of the NIST Secure Hash Algorithm Validation System (SHAVS) (https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-ValidationProgram/documents/shs/SHAVS.pdf). The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

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

2.1.16.1 FDEEEcPP20E:FCS_COP.1.1(b)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

See the description in FDEAAcPP20E:FCS_COP.1(b).

Component Guidance Assurance Activities: The evaluator checks the operational guidance documents to determine that any system configuration necessary to enable required hash size functionality is provided.

See the description in FDEAAcPP20E:FCS_COP.1(b).

Component Testing Assurance Activities: The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bitÂ¬oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bitÂ¬oriented vs. the byteÂ¬oriented test mode.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this cPP.

Short Messages Test Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 99*i, where 1 <= i <= m. For SHA-512, the length of the i-th

message is 1024 + 99*i, where 1 <+ i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. For SHA-256, the length of the i-th message is 512 + 8*99*i, where 1 <= i <= m/8. For SHA-512, the length of the i-th message is 1024 + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. (TD0233 applied).

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates

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

2.1.17.1 FDEAAcPP20E:FCS_COP.1.1(c)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If HMAC was selected:

The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

If CMAC was selected:

The evaluator shall examine the TSS to ensure that it specifies the following values used by the CMAC function: key length, block cipher used, block size (of the cipher), and output MAC length used.

Section 6.1 of the TSS explains that the TOE implements HMAC-SHA-256 using 256-bit keys, the SHA-256 hash algorithm, a 512-bit block size, and an output MAC length of 256 bits.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If HMAC was selected:

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key using a known good implementation.

If CMAC was selected:

For each of the supported parameter sets, the evaluator shall compose at least 15 sets of test data. Each set shall consist of a key and message data. The test data shall include messages of different lengths, some with partial blocks as the last block and some with full blocks as the last block. The test data keys shall include cases for which subkey K1 is generated both with and without using the irreducible polynomial R_b, as well as cases for which subkey K2 is generated from K1 both with and without using the irreducible polynomial R_b. (The subkey generation and polynomial R_b are as defined in SP800-38E.) The evaluator shall have the TSF generate CMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating CMAC tags with the same key using a known good implementation.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

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

2.1.18.1 FDEEEcPP20E:FCS_COP.1.1(c)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If HMAC was selected:

The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

If CMAC was selected:

The evaluator shall examine the TSS to ensure that it specifies the following values used by the CMAC function: key length, block cipher used, block size (of the cipher), and output MAC length used.

See the description in FDEAAcPP20E:FCS_COP.1(c).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If HMAC was selected:

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key using a known good implementation.

If CMAC was selected:

For each of the supported parameter sets, the evaluator shall compose at least 15 sets of test data. Each set shall consist of a key and message data. The test data shall include messages of different lengths, some with partial blocks as the last block and some with full blocks as the last block. The test data keys shall include cases for which subey K1 is generated both with and without using the irreducible polynomial R_b, as well as cases for which subkey K2 is generated from K1 both with and without using the irreducible polynomial R_b. (The subkey generation and polynomial R_b are as defined in SP800-38E.) The evaluator shall have the TSF generate CMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating CMAC tags with the same key using a known good implementation.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates

2.1.19 Cryptographic operation (AES Data Encryption/Decryption) (FDEAAcPP20E:FCS_COP.1(f))

2.1.19.1 FDEAAcPP20E:FCS_COP.1.1(f)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for encryption.

Section 6.1 of the TSS explains the TOE uses an AES CBC kernel implementation dedicated to drive encryption/decryption. The implementation uses AES- 256 bit keys.

Component Guidance Assurance Activities: If multiple encryption modes are supported, the evaluator examines the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

The administrator does not have to select a mode or key size so no guidance is required.

Component Testing Assurance Activities: The following tests are conditional based upon the selections made in the SFR.

AES-CBC Tests

For the AES-CBC tests described below, the plaintext, ciphertext, and IV values shall consist of 128-bit blocks. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known-good implementation.

These tests are intended to be equivalent to those described in NIST's AES Algorithm Validation Suite (AESAVS) (http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf). Known answer values tailored to exercise the AES-CBC implementation can be obtained using NIST's CAVS Algorithm Validation Tool or from NIST's ACPV service for automated algorithm tests (acvp.nist.gov), when available. It is not recommended that evaluators use values obtained from static sources such as the example NIST's AES Known Answer Test Values from the AESAVS document, or use values not generated expressly to exercise the AES-CBC implementation.

AES-CBC Known Answer Tests

KAT-1 (GFSBox):

To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different plaintext values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros.

To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different ciphertext values for each selected key size and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using a key value of all zeros and an IV of all zeros.

KAT-2 (KeySBox):

To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros.

To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the plaintext that results from AES-CBC decryption of an all-zeros ciphertext using the given key and an IV of all zeros.

KAT-3 (Variable Key):

To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of keys for each selected key size (as described below) and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using each key and an IV of all zeros.

Key i in each set shall have the leftmost i bits set to ones and the remaining bits to zeros, for values of i from 1 to the key size. The keys and corresponding ciphertext are listed in AESAVS, Appendix E.

To test the decrypt functionality of AES-CBC, the evaluator shall use the same keys as above to decrypt the ciphertext results from above. Each decryption should result in an all-zeros plaintext.

KAT-4 (Variable Text):

To test the encrypt functionality of AES-CBC, for each selected key size, the evaluator shall supply a set of 128-bit plaintext values (as described below) and obtain the ciphertext values that result from AES-CBC encryption of each plaintext value using a key of each size and IV consisting of all zeros.

Plaintext value i shall have the leftmost i bits set to ones and the remaining bits set to zeros, for values of i from 1 to 128. The plaintext values are listed in AESAVS, Appendix D.

To test the decrypt functionality of AES-CBC, for each selected key size, use the plaintext values from above as ciphertext input, and AES-CBC decrypt each ciphertext value using key of each size consisting of all zeros and an IV of all zeros.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting nine i-block messages for each selected key size, for 2 <= i <= 10. For each test, the evaluator shall supply a key, an IV, and a plaintext message of length i blocks, and encrypt the message using AES-CBC. The resulting ciphertext values shall be compared to the results of encrypting the plaintext messages using a known good implementation.

The evaluator shall test the decrypt functionality by decrypting nine i-block messages for each selected key size, for 2 <= i <= 10. For each test, the evaluator shall supply a key, an IV, and a ciphertext message of length i blocks, and

decrypt the message using AES-CBC. The resulting plaintext values shall be compared to the results of decrypting the ciphertext messages using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality for each selected key size using 100 3-tuples of pseudo-random values for plaintext, IVs, and keys.

The evaluator shall supply a single 3-tuple of pseudo-random values for each selected key size. This 3-tuple of plaintext, IV, and key is provided as input to the below algorithm to generate the remaining 99 3-tuples, and to run each 3-tuple through 1000 iterations of AES-CBC encryption.

Input: PT, IV, Key

Key[0] = Key

IV[0] = IV

PT[0] = PT

for i = 1 to 100

Output Key[i], IV[i], PT[0]

for j = 1 to 1000

if j == 1

CT[1] = AES-CBC-Encrypt(Key[i], IV[i], PT[1])

PT[2] = IV[i]

else

CT[j] = AES-CBC-Encrypt(Key[i], PT[j])

PT[j+1] = CT[j-1]

Output CT[1000]

If KeySize == 128 Key[i+1] = Key[i] xor CT[1000]

If KeySize == 256 Key[i+1] = Key[i] xor ((CT[999] << 128) | CT[1000])

IV[i+1] = CT[1000]

PT[0] = CT[999]

The ciphertext computed in the 1000th iteration (CT[1000]) is the result for each of the 100 3-tuples for each selected key size. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as above, exchanging CT and PT, and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

XTS-AES Test

The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

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

2.1.20.1 FDEEEcPP20E:FCS_COP.1.1(f)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for encryption.

See the description in FDEEEcPP20E_FCS_COP.1(f)

Component Guidance Assurance Activities: If multiple encryption modes are supported, the evaluator examines the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

See the description in FDEEEcPP20E_FCS_COP.1(f)

Component Testing Assurance Activities: The following tests are conditional based upon the selections made in the SFR.

AES-CBC Tests

For the AES-CBC tests described below, the plaintext, ciphertext, and IV values shall consist of 128-bit blocks. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known-good implementation.

These tests are intended to be equivalent to those described in NIST's AES Algorithm Validation Suite (AESAVS) (http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf). Known answer values tailored to exercise the AES-CBC implementation can be obtained using NIST's CAVS Algorithm Validation Tool or from NIST's ACPV service for automated algorithm tests (acvp.nist.gov), when available. It is not recommended that evaluators use values obtained from static sources such as the example NIST's AES Known Answer Test Values from the AESAVS document, or use values not generated expressly to exercise the AES-CBC implementation.

AES-CBC Known Answer Tests

KAT-1 (GFSBox):

To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different plaintext values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros.

To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different ciphertext values for each selected key size and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using a key value of all zeros and an IV of all zeros.

KAT-2 (KeySBox):

To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros.

To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the plaintext that results from AES-CBC decryption of an all-zeros ciphertext using the given key and an IV of all zeros.

KAT-3 (Variable Key):

To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of keys for each selected key size (as described below) and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using each key and an IV of all zeros.

Key i in each set shall have the leftmost i bits set to ones and the remaining bits to zeros, for values of i from 1 to the key size. The keys and corresponding ciphertext are listed in AESAVS, Appendix E.

To test the decrypt functionality of AES-CBC, the evaluator shall use the same keys as above to decrypt the ciphertext results from above. Each decryption should result in an all-zeros plaintext.

KAT-4 (Variable Text):

To test the encrypt functionality of AES-CBC, for each selected key size, the evaluator shall supply a set of 128-bit plaintext values (as described below) and obtain the ciphertext values that result from AES-CBC encryption of each plaintext value using a key of each size and IV consisting of all zeros.

Plaintext value i shall have the leftmost i bits set to ones and the remaining bits set to zeros, for values of i from 1 to 128. The plaintext values are listed in AESAVS, Appendix D.

To test the decrypt functionality of AES-CBC, for each selected key size, use the plaintext values from above as ciphertext input, and AES-CBC decrypt each ciphertext value using key of each size consisting of all zeros and an IV of all zeros.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting nine i-block messages for each selected key size, for 2 <= i <= 10. For each test, the evaluator shall supply a key, an IV, and a plaintext message of length i blocks, and encrypt the message using AES-CBC. The resulting ciphertext values shall be compared to the results of encrypting the plaintext messages using a known good implementation.

The evaluator shall test the decrypt functionality by decrypting nine i-block messages for each selected key size, for 2 <= i <= 10. For each test, the evaluator shall supply a key, an IV, and a ciphertext message of length i blocks, and decrypt the message using AES-CBC. The resulting plaintext values shall be compared to the results of decrypting the ciphertext messages using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality for each selected key size using 100 3-tuples of pseudo-random values for plaintext, IVs, and keys.

The evaluator shall supply a single 3-tuple of pseudo-random values for each selected key size. This 3-tuple of plaintext, IV, and key is provided as input to the below algorithm to generate the remaining 99 3-tuples, and to run each 3-tuple through 1000 iterations of AES-CBC encryption.

Input: PT, IV, Key

Key[0] = Key

IV[0] = IV

PT[0] = PT

for i = 1 to 100

Output Key[i], IV[i], PT[0]

for j = 1 to 1000

if j == 1

CT[1] = AES-CBC-Encrypt(Key[i], IV[i], PT[1])

PT[2] = IV[i]

else

CT[j] = AES-CBC-Encrypt(Key[i], PT[j])

PT[j+1] = CT[j-1]

Output CT[1000]

If KeySize == 128 Key[i+1] = Key[i] xor CT[1000]

If KeySize == 256 Key[i+1] = Key[i] xor ((CT[999] << 128) | CT[1000])

IV[i+1] = CT[1000]

PT[0] = CT[999]

The ciphertext computed in the 1000th iteration (CT[1000]) is the result for each of the 100 3-tuples for each selected key size. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as above, exchanging CT and PT, and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

XTS-AES Test

The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

2.1.21 Cryptographic operation (Key Encryption) (FDEAAcPP20E:FCS_COP.1(g))

2.1.21.1 FDEAAcPP20E:FCS_COP.1.1(g)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for the key encryption.

KMD

The evaluator shall examine the vendor's KMD to verify that it includes a description of how key encryption will be used as part of the key chain.

Section 6.1 of the TSS states the TOE has a gcrypt AES CBC implementation used for key managements operations (decryption of the encrypted DEKs). This implementation uses AES- 256 bit keys.

KMD – The FCS_KYC_EXT.1/2 description explains the TOE uses PBKDFv2 to transform the operator's password into a 256-bit BEV, and then uses that BEV to AES decrypt the DEKs stored in the header(s) stored on the drive.

Component Guidance Assurance Activities: If multiple key encryption modes are supported, the evaluator examines the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

The administrator does not have to select a mode or key size so no guidance is required.

Component Testing Assurance Activities: The AES test should be followed in FCS_COP.1(f) Cryptographic Operation (AES Data Encryption/Decryption.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

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

2.1.22.1 FDEEEcPP20E:FCS_COP.1.1(g)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for the key encryption.

KMD

The evaluator shall examine the vendor's KMD to verify that it includes a description of how key encryption will be used as part of the key chain.

See the description in FDEAAcPP20E:FCS_COP.1(g).

Component Guidance Assurance Activities: If multiple key encryption modes are supported, the evaluator examines the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described.

See the description in FDEAAcPP20E:FCS_COP.1(g).

Component Testing Assurance Activities: The AES test should be followed in FCS_COP.1(f) Cryptographic Operation (AES Data Encryption/Decryption).

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

2.1.23 Cryptographic Key Derivation (FDEAAcPP20E:FCS_KDF_EXT.1)

2.1.23.1 FDEAAcPP20E:FCS_KDF_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 and SP 800-132.

KMD

The evaluator shall examine the vendor's KMD to ensure that all keys used are derived using an approved method and a description of how and when the keys are derived.

Section 6.1 of the TSS explains that the TOE uses 800-132 (PBKDFv2) using with HMAC-SHA-256 and a number of iterations and a 256-bit salt to transform the operator's password into a Derived Key for decrypting the encrypted DEKs. The number of iterations is determined by the specified number of milliseconds (2000 milliseconds) multiplied by the number of PBKDF operations per/second to achieve a delay specified by the administrator.

KMD – See TSS description for details

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.24 Cryptographic Key Derivation (FDEEEcPP20E:FCS_KDF_EXT.1)

2.1.24.1 FDEEEcPP20E:FCS_KDF_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 and SP 800-132.

KMD

The evaluator shall examine the vendor's KMD to ensure that all keys used are derived using an approved method and a description of how and when the keys are derived.

See the description in FDEAAcPP20E:FCS_KDF_EXT.1.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.25.1 FDEAAcPP20E:FCS_KYC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.25.2 FDEAAcPP20E:FCS_KYC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS contains a high-level description of the BEV sizes - that it supports BEV outputs of no fewer 128 bits for products that support only AES-128, and no fewer than 256 bits for products that support AES-256.

KMD

The evaluator shall examine the KMD describes a high level description of the key hierarchy for all authorizations methods selected in FCS_AFA_EXT.1 that are used to protect the BEV. The evaluator shall examine the KMD to ensure it describes the key chain in detail. The description of the key chain shall be reviewed to ensure it maintains a chain of keys using key wrap or key derivation methods that meet FCS_COP.1(d) and FCS_KDF_EXT.1.

The evaluator shall examine the KMD to ensure that it describes how the key chain process functions, such that it does not expose any material that might compromise any key in the chain. (e.g. using a key directly as a compare value against a TPM) This description must include a diagram illustrating the key hierarchy implemented and detail where all keys and keying material is stored or what it is derived from. The evaluator shall examine the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or the initial authorization value and the effective strength of the BEV is maintained throughout the key chain.

The evaluator shall verify the KMD includes a description of the strength of keys throughout the key chain.

Section 6.1 of the TSS states that the TOE supports a BEV key size of 256-bits. This is adequate as AES-256 is claimed elsewhere.

KMD – Figure 2 in the Section 6.1 of the TSS provides a key management diagram. The diagram shows step by step the lifecycle of the key chain. In each case, the algorithm and strength are identified and they match the claims in the SFRs. The evaluator is able to determine the key is not exposed and its strength remains at 256-bits.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.26.1 FDEEEcPP20E:FCS_KYC_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.26.2 FDEEEcPP20E:FCS_KYC_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: KMD

The evaluator shall examine the KMD to ensure it describes a high level key hierarchy and details of the key chain. The description of the key chain shall be reviewed to ensure it maintains a chain of keys using key wrap or key derivation methods that meet FCS_KDF_EXT.1, FCS_COP.1(d), FCS_COP.1(e), and/or FCS_COP.1(g).

The evaluator shall examine the KMD to ensure that it describes how the key chain process functions, such that it does not expose any material that might compromise any key in the chain. (e.g. using a key directly as a compare value against a TPM) This description must include a diagram illustrating the key hierarchy implemented and detail where all keys and keying material is stored or what it is derived from. The evaluator shall examine the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or knowledge of the BEV and the effective strength of the DEK is maintained throughout the Key Chain.

The evaluator shall verify the KMD includes a description of the strength of keys throughout the key chain.

See the description for FDEAACPP20:FCS_KYC_EXT.1.2.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.27 Cryptographic Password Construct and Conditioning (FDEAAcPP20E:FCS_PCC_EXT.1)

2.1.27.1 FDEAAcPP20E:FCS_PCC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes the manner in which the TOE enforces the construction of passwords, including the length, and requirements on characters (number and type). The evaluator also verifies that the TSS provides a description of how the password is conditioned and the evaluator ensures it satisfies the requirement.

KMD

The evaluator shall examine the KMD to ensure that the formation of the BEV and intermediary keys is described and that the key sizes match that selected by the ST author.

The evaluator shall check that the KMD describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the KMD contains a description of how the output of the hash function is used to form the submask that will be input into the function and is the same length as the BEV as specified above.

Section 6.1 of the TSS states that the TOE allows passwords up to 512 characters in length, and the TOE allows uppercase/lowercase letters, numbers, and ASCII printable characters. The TOE will reject a password containing other characters. The TOE conditions passwords by combining them with a 256-bit salt using PBKDFv2.

KMD – The KMD and TSS describe that the TOE uses 800-132 (PBKDFv2) with HMAC-SHA-256 and a number of iterations and a 256-bit salt to transform the operator's password into a Derived Key for decrypting the encrypted DEKs with 256-bit strength.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall also perform the following tests:

Test 1: Ensure that the TOE supports passwords/passphrases of a minimum length of 64 characters.

Test 2: If the TOE supports a password/passphrase length up to a maximum number of characters, n (which would be greater than 64), then ensure that the TOE will not accept more than n characters.

Test 3: Ensure that the TOE supports passwords consisting of all characters assigned and supported by the ST author.

Test 1 - The evaluator attempted to enable the software encryption with a 64-character passphrase. The passphrase was accepted as expected.

Test 2 – The TOE supports a passphrase of 512 characters. The evaluator attempted to enable the software encryption with a 512-character passphrase and it was accepted. The evaluator then attempted a passphrase of 513 characters and it was rejected.

Test 3 – The evaluator attempted to set the passphrase using all the printable ASCII characters and it the passphrase was accepted.

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

2.1.28.1 FDEAAcPP20E:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.28.2 FDEAAcPP20E:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For any RBG services provided by a third party, the evaluator shall ensure the TSS includes a statement about the expected amount of entropy received from such a source, and a full description of the processing of the output of the third-party source. The evaluator shall verify that this statement is consistent with the selection made in FCS_RBG_EXT.1.2 for the seeding of the DRBG. If the ST specifies more than one DRBG, the evaluator shall examine the TSS to verify that it identifies the usage of each DRBG mechanism.

The TOE does not use a third-party DRBG.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected DRBG mechanism(s), if necessary, and provides information regarding how to instantiate/call the DRBG for RBG services needed in this cPP.

The evaluated DRBG is used by default and no configuration is necessary.

Component Testing Assurance Activities: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable by the TOE, the evaluator shall perform 15 trials for each configuration. The evaluator shall verify that the instructions in the operational guidance for configuration of the RNG are valid.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

2.1.29 Random Bit Generation (FDEEEcPP20E:FCS_RBG_EXT.1)

2.1.29.1 FDEEEcPP20E:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.29.2 FDEEEcPP20E:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For any RBG services provided by a third party, the evaluator shall ensure the TSS includes a statement about the expected amount of entropy received from such a source, and a full description of the processing of the output of the third-party source. The evaluator shall verify that this statement is consistent with the selection made in FCS_RBG_EXT.1.2 for the seeding of the DRBG. If the ST specifies more than one DRBG, the evaluator shall examine the TSS to verify that it identifies the usage of each DRBG mechanism.

The TOE does not use a third-party DRBG.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected DRBG mechanism(s), if necessary, and provides information regarding how to instantiate/call the DRBG for RBG services needed in this cPP.

The evaluated DRBG is used by default and no configuration is necessary.

Component Testing Assurance Activities: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable by the TOE, the evaluator shall perform 15 trials for each configuration. The evaluator shall verify that the instructions in the operational guidance for configuration of the RNG are valid.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

This is addressed by CAVP testing. See Section 1.1 for a listing of CAVP algorithm certificates.

2.1.30 Cryptographic Operation (Salt, Nonce, and Initialization Vector Generation) (FDEAAcPP20E:FCS_SNI_EXT.1)

2.1.30.1 FDEAAcPP20E:FCS_SNI_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.30.2 FDEAAcPP20E:FCS_SNI_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.30.3 FDEAAcPP20E:FCS_SNI_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes how salts are generated. The evaluator shall confirm that the salt is generating using an RBG described in FCS_RBG_EXT.1 or by the Operational Environment. If external function is used for this purpose, the TSS should include the specific API that is called with inputs.

The evaluator shall ensure the TSS describes how nonces are created uniquely and how IVs and tweaks are handled (based on the AES mode). The evaluator shall confirm that the nonces are unique and the IVs and tweaks meet the stated requirements.

Section 6.1 of the TSS states that the TOE generates its salts using its SHA-256 HMAC_DRBG. The TOE generates its AES-CBC IVs using ESSIV:SHA256. The TOE generates no nonces but generates it 256-bit AES XTS tweaks (used for data partition encryption) using its HMAC_DRBG.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.1.31.1 FDEEEcPP20E:FCS_SNI_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.31.2 FDEEEcPP20E:FCS_SNI_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.31.3 FDEEEcPP20E:FCS_SNI_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes how salts are generated. The evaluator shall confirm that the salt is generating using an RBG described in FCS_RBG_EXT.1 or by the Operational Environment. If external function is used for this purpose, the TSS should include the specific API that is called with inputs.

The evaluator shall ensure the TSS describes how nonces are created uniquely and how IVs and tweaks are handled (based on the AES mode). The evaluator shall confirm that the nonces are unique and the IVs and tweaks meet the stated requirements.

See FDEAAcPP20:FCS_SNI_EXT.1.3.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.32 Validation (FDEAAcPP20E:FCS_VAL_EXT.1)

2.1.32.1 FDEAAcPP20E:FCS_VAL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.32.2 FDEAAcPP20E:FCS_VAL_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.32.3 FDEAAcPP20E:FCS_VAL_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine which authorization factors support validation.

The evaluator shall examine the TSS to review a high-level description if multiple submasks are used within the TOE, how the submasks are validated (e.g., each submask validated before combining, once combined validation takes place).

KMD

The evaluator shall examine the KMD to verify that it describes the methods the TOE employs to limit the number of consecutively failed authorization attempts.

The evaluator shall examine the vendor's KMD to ensure it describes how validation is performed. The description of the validation process in the KMD provides detailed information how the TOE validates the submasks.

The KMD describes how the process works, such that it does not expose any material that might compromise the submask(s).

Section 6.1 of the ST identifies passwords as the only authorization factor. A password is required when the machine is power cycled.

KMD - Section 6.1 of the ST explains the TOE validates the operator's password by first subjecting the password and salt to PBKDFv2 to form the Derived Key (DerKey). The TOE uses the DerKey to decrypt the masterKey stripes and reconstitutes the masterKey; however, before using the masterKey, the TOE first performs iterative HMAC-SHA-256 using the operator's password, the masterKey salt, masterKey iterations, and masterKey as inputs, and then compares the resulting value to the stored masterKey's digest stored in the header to ensure the two match.

If the TOE detects more than five incorrect passwords, then the TOE will block all subsequent attempts to validate the operator’s password (and not even attempt to validate the password). The TOE clears its counter upon a reboot.

Component Guidance Assurance Activities: [conditional] If the validation functionality is configurable, the evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established.

[conditional] If the validation functionality is specified by the ST Author, the evaluator shall examine the operational guidance to ensure it states the values that the TOE uses for limits regarding validation attempts.

Section 5.1.3 of the User Guide explains that an incorrect passphrase (is tolerated five times, each responding with an error message and attempt count value. Then subsequent attempts correct or not are rejected, a reboot is required to make another five attempts.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall determine the limit on the average rate of the number of consecutive failed authorization attempts. The evaluator will test the TOE by entering that number of incorrect authorization factors in consecutive attempts to access the protected data. If the limit mechanism includes any 'lockout' period, the time period tested should include at least one such period. Then the evaluator will verify that the TOE behaves as described in the TSS.

Test 2: For each validated authorization factor, ensure that when the user provides an incorrect authorization factor, the TOE prevents the BEV from being forwarded outside the TOE (e.g., to the EE).

Test 1 - The TSS states that after 5 failed attempts, the TOE must be rebooted before it will accept further passwords. The evaluator attempted to log into the software encryption using an incorrect password. The evaluator repeated this 5 times and received an incrementing counter each time. After the 5th failure, the evaluator entered the incorrect password again and the counter did not increment. The evaluator also attempted to enter the correct password but it was not accepted either. A reboot was required.

Test 2 – See FCS_AFA_EXT.2-t1. This test rebooted the TOE, demonstrated a bad password was not accepted and a good password was accepted for access to the TOE's data.

2.1.33 Validation (FDEEEcPP20E:FCS_VAL_EXT.1)

2.1.33.1 FDEEEcPP20E:FCS_VAL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.33.2 FDEEEcPP20E:FCS_VAL_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.33.3 FDEEEcPP20E:FCS_VAL_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine which authorization factors support validation.

The evaluator shall examine the TSS to review a high-level description if multiple submasks are used within the TOE, how the submasks are validated (e.g., each submask validated before combining, once combined validation takes place).

The evaluator shall also examine the TSS to determine that a subset or all of the authorization factors identified in the SFR can be used to exit from a Compliant power saving state.

KMD

The evaluator shall examine the KMD to verify that it described the method the TOE employs to limit the number of consecutively failed authorization attempts.

The evaluator shall examine the vendor's KMD to ensure it describes how validation is performed. The description of the validation process in the KMD provides detailed information how the TOE validates the BEV.

The KMD describes how the process works, such that it does not expose any material that might compromise the submask(s).

See the description in FDEAAcPP20E:FCS_VAL_EXT.1.3.

Component Guidance Assurance Activities: [conditional] If the validation functionality is configurable, the evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established.

[conditional] If ST Author assigned, the evaluator shall examine the operational guidance to ensure it states the values the TOE uses for limits regarding validation attempts. (TD0229 applied)

The evaluator shall verify that the guidance documentation states which authorization factors are allowed to exit a Compliant power saving state.

See the description in FDEAAcPP20E:FCS_VAL_EXT.1.3

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall determine the limit on the average rate of the number of consecutive failed authorization attempts. The evaluator will test the TOE by entering that number of incorrect authorization factors in consecutive attempts to access the protected data. If the limit mechanism includes any 'lockout' period, the time period tested should include at least one such period. Then the evaluator will verify that the TOE behaves as described in the TSS.

Test 2: The evaluator shall force the TOE to enter a Compliant power saving state, attempt to resume it from this state, and verify that only a valid authorization factor as defined by the guidance documentation is sufficient to allow the TOE to exit the Compliant power saving state.

See the description in FDEAAcPP20E:FCS_VAL_EXT.1.3.

2.2 User data protection (FDP)

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

2.2.1.1 FDEEEcPP20E:FDP_DSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.1.2 FDEEEcPP20E:FDP_DSK_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that the description is comprehensive in how the data is written to the disk and the point at which the encryption function is applied. The TSS must make the case that standard methods of accessing the disk drive via the host platforms operating system will pass through these functions.

For the cryptographic functions that are provided by the Operational Environment, the evaluator shall check the TSS to ensure it describes, for each platform identified in the ST, the interface(s) used by the TOE to invoke this functionality.

The evaluator shall verify the TSS in performing the evaluation activities for this requirement. The evaluator shall ensure the comprehensiveness of the description, confirms how the TOE writes the data to the disk drive, and the point at which it applies the encryption function.

The evaluator shall verify that the TSS describes the initialization of the TOE and the activities the TOE performs to ensure that it encrypts all the storage devices entirely when a user or administrator first provisions the TOE. The evaluator shall verify the TSS describes areas of the disk that it does not encrypt (e.g., portions associated with the Master Boot Records (MBRs), boot loaders, partition tables, etc.). If the TOE supports multiple disk encryptions, the evaluator shall examine the administration guidance to ensure the initialization procedure encrypts all storage devices on the platform.

KMD

The evaluator shall verify the KMD includes a description of the data encryption engine, its components, and details about its implementation (e.g. for hardware: integrated within the device's main SOC or separate co-processor, for software: initialization of the product, drivers, libraries (if applicable), logical interfaces for encryption/decryption, and areas which are not encrypted (e.g. boot loaders, portions associated with the Master Boot Record (MBRs), partition tables, etc.)). The evaluator shall verify the KMD provides a functional (block) diagram showing the main components (such as memories and processors) and the data path between, for hardware, the device's host interface and the device's persistent media storing the data, or for software, the initial steps needed to the activities the TOE performs to ensure it encrypts the storage device entirely when a user or administrator first provisions the product. The hardware encryption diagram shall show the location of the data encryption engine within the data path. The evaluator shall validate that the hardware encryption diagram contains enough detail showing the main components within the data path and that it clearly identifies the data encryption engine.

The evaluator shall verify the KMD provides sufficient instructions for all platforms to ensure that when the user enables encryption, the product encrypts all hard storage devices. The evaluator shall verify that the KMD describes the data flow from the device's host interface to the device's persistent media storing the data. The evaluator shall verify that the KMD provides information on those conditions in which the data bypasses the data encryption engine (e.g. read-write operations to an unencrypted Master Boot Record area).

The evaluator shall verify that the KMD provides a description of the platform's boot initialization, the encryption initialization process, and at what moment the product enables the encryption. The evaluator shall validate that the product does not allow for the transfer of user data before it fully initializes the encryption. The evaluator shall ensure the software developer provides special tools which allow inspection of the encrypted drive either in-band or out-of-band, and may allow provisioning with a known key.

Section 6.1 of the ST provides a diagram of the LUKS partition. Section 7 of the ST explains the data encryption engine is based on LUKS, and is comprised of both a userspace component and a kernel-level component. The userspace component handles derivation of the Derived Key from the user's password and the subsequent decryption of the DEK with the Derived Key. The kernel-level component receives the DEK from the userspace component and then encrypts/decrypts data written to/read from the encrypted partition/drive. The data encryption engine itself is a Network Attached Storage (NAS) device, where all executable code of the data encryption engine executes within a dedicated processor, with its own dedicated Flash memory. While the TOE's

does not encrypt its internal dedicated Flash memory, it provides no access this memory, and only exposes the encrypted Removable Memory Cartridge (drive) to network-attached clients. The TOE ensures that access to the RMC/drive is always encrypted, and does not permit plaintext access to protected partitions or drive. Because the TOE utilizes a dedicated processor and dedicated internal Flash, the TOE only provides access to the RMC/drive once fully initialized and after receiving the administrator's password.

The User Guide describes the TOE's initialization process and setup for the SW-layer. The TOE maintains a separate, unencrypted, internal Flash chip to house its CentOS-based firmware that is beyond the RMC drive that the TOE encrypts. If the administrator configures the RMC drive for use as a raw block device, then the TOE encrypts the entire drive (with a small area reserved for the LUKS header). Otherwise, if the administrator chooses to partition the RMC drive, then the drive's partition table and LUKS headers for each partition will be in plaintext, with all partition data encrypted.

KMD – See above description

Component Guidance Assurance Activities: The evaluator shall review the AGD guidance to determine that it describes the initial steps needed to enable the FDE function, including any necessary preparatory steps. The guidance shall provide instructions that are sufficient, on all platforms, to ensure that all hard drive devices will be encrypted when encryption is enabled.

Section 5.4 of the User Guide explains how to establish the software layer encryption. It explains how to select the entire hard drive or how to partition the drive and encrypt the partitions. There is only one hardware platform so it is sufficiently addressed.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: Write data to random locations, perform required actions and compare:

- Ensure TOE is initialized and, if hardware, encryption engine is ready;

- Provision TOE to encrypt the storage device. For SW Encryption products, or hybrid products use a known key and the developer tools.

- Determine a random character pattern of at least 64 KB;

- Retrieve information on what the device TOE's lowest and highest logical address is for which encryption is enabled.

Test 2: Write pattern to storage device in multiple locations:

- For HW Encryption, randomly select several logical address locations within the device's lowest to highest address range and write pattern to those addresses;

- For SW Encryption, write the pattern using multiple files in multiple logical locations.

Test 3: Verify data is encrypted:

- For HW Encryption:

-- engage device's functionality for generating a new encryption key, thus performing an erase of the key per FCS_CKM.4(a);

-- Read from the same locations at which the data was written;

-- Compare the retrieved data to the written data and ensure they do not match

- For SW Encryption, using developer tools;

-- Review the encrypted storage device for the plaintext pattern at each location where the file was written and confirm plaintext pattern cannot be found.

-- Using the known key, verify that each location where the file was written, the plaintext pattern can be correctly decrypted using the key.

-- If available in the developer tools, verify there are no plaintext files present in the encrypted range.

Test 1 – The evaluator ensured the TOE was configured. The evaluator them created a 64K file of random data. The evaluator also calculated the size of the disk and the number of sectors to use in two following tests.

Test 2 - The evaluator logged onto the device as the root user and wrote the random file created in test case 1 to 3 locations on the disk. The evaluator picked the first location since it was just past the LUKS header. The second location is a random location in the middle of the disk and the third location is at the end of the disk.

Test 3 – After the evaluator wrote data to three locations, the evaluator read the data back from those locations and confirmed it could be decrypted and was as expected. The evaluator then unmounted the software encryption and decrypted it. The evaluator then re-encrypted the partition and mounted it so it would be available for access. The evaluator the read the same areas of disk to ensure the data had been overwritten and wrote the 3 known areas to files. The evaluator then compared the files had changed from the original random file. Examination of the encrypted areas is in 3.5.1.

2.3 Security management (FMT)

2.3.1 Management of Functions Behavior (FDEAACPP20E:FMT_MOF.1)

2.3.1.1 FDEAAcPP20E:FMT_MOF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If support for Compliant power saving state(s) are claimed in the ST, the evaluator shall ensure the TSS describes how these are managed and shall ensure that TSS describes how only privileged users (administrators) are allowed to manage the states.

Section 6.3 of the ST states the TOE provides the Compliant power-saving state G3, mechanical off. Only the authorized administrator can issue the shutdown command.

Component Guidance Assurance Activities: The evaluator to check if guidance documentation describes which authorization factors are required to change Compliant power saving state behavior and properties.

An administrator must have logged onto the TOE with a password to issue the shutdown command. The shutdown command is described in the CLI section of the User Guide (12.3).

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator presents a privileged authorization credential to the TSF and validates that changes to Compliant power saving state behavior and properties are allowed.

Test 2: The evaluator presents a non-privileged authorization credential to the TSF and validates that changes to Compliant power saving state behavior are not allowed.

Test 1 - The evaluator logged onto the TOE and issued the shutdown command. The TOE transitioned to the shutdown state.

Test 2 - The evaluator used an incorrect password to log onto the TOE. The evaluator was unable to issue a command to attempt to transition the power state and the only function available was the login prompt.

2.3.2 Specification of Management Functions (FDEAAcPP20E:FMT_SMF.1)

2.3.2.1 FDEAAcPP20E:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If item a) is selected in FMT_SMF_.1.1: The evaluator shall ensure the TSS describes how the TOE sends the request to the EE to change the DEK.

If item b) is selected in FMT_SMF_.1.1: The evaluator shall ensure the TSS describes how the TOE sends the request to the EE to cryptographically erase the DEK.

If item c) is selected in FMT_SMF_.1.1: The evaluator shall ensure the TSS describes the methods by which users may change the set of all authorization factor values supported.

If item d) is selected in FMT_SMF_.1.1: The evaluator shall ensure the TSS describes the process to initiate TOE firmware/software updates.

If item e) is selected in FMT_SMF_.1.1: If power saving states can be managed, the evaluator shall ensure that the TSS describes how this is performed, including how the TOE supports disabling certain power saving states if more than one are supported. If additional management

functions are claimed in the ST, the evaluator shall ensure the TSS describes the additional functions.

Section 6.3 of the ST explains the management functions. The TOE provides each of the required management services with no additional ones. Because the TOE fulfills the AA and EE requirements together, the TOE need not "forward" requests to change the DEK or cryptographically erase the DEK. Instead, the TOE provides an administrator command that will decrypt and erase the DEK ("**`rmcctl -D`**") and a command to create a new partition ("**`rmcctl -s 0 --part 2 50% 50% --force`**"). The TOE supports changing of the authorization factors (the administrator can remove a partition and recreate it to change the associated password). The User Guide describes the TOE's "Field Update" process, which consists of securely copying the new update image and signature file to the TOE and then executing the "`fupdate`" command, after which the TOE will detect the new update, verify the signature, and (if the signature verifies successfully) install the update. The TOE does not provide any manageable power-saving states.

Component Guidance Assurance Activities: If item a) and/or b) is selected in FMT_SMF.1.1: The evaluator shall examine the operational guidance to ensure that it describes how the functions for A and B can be initiated by the user.

If item c) is selected in FMT_SMF_.1.1: The evaluator shall examine the operational guidance to ensure that it describes how selected authorization factor values are changed.

If item d) is selected in FMT_SMF_.1.1: The evaluator shall examine the operational guidance to ensure that it describes how to initiate TOE firmware/software updates.

If item e) is selected in FMT_SMF_.1.1: Default Authorization Factors: It may be the case that the TOE arrives with default authorization factors in place. If it does, then the selection in section E must be made so that there is a

mechanism to change these authorization factors. The operational guidance shall describe the method by which the user changes these factors when they are taking ownership of the device. The TSS shall describe the default authorization factors that exist.

Disable Key Recovery: The guidance for disabling this capability shall be described in the AGD documentation.

Power Saving: The guidance shall describe the power saving states that are supported by the TSF, how these states are applied, how to configure when these states are applied (if applicable), and how to enable/disable the use of specific power saving states (if applicable).

Section 5.4 of the User Guide explains how to establish the software encryption. It also explains that if the passphrase is changed, the data will be wiped on the device. The same section of the User Guide provides the command for decrypting the software layer. The CLI reference section provides the fupdate command for initiating an update on the TOE

Component Testing Assurance Activities: If item a) and/or b) is selected in FMT_SMF.1.1: The evaluator shall verify that the TOE has the functionality to forward a command to the EE to change and cryptographically erase the DEK. The actual testing of the cryptographic erase will take place in the EE.

If item c) is selected in FMT_SMF.1.1: The evaluator shall initialize the TOE such that it requires the user to input an authorization factor in order to access encrypted data.

Test 1: The evaluator shall first provision user authorization factors, and then verify all authorization values supported allow the user access to the encrypted data. Then the evaluator shall exercise the management functions to change a user's authorization factor values to a new one. Then he or she will verify that the TOE denies access to the user's encrypted data when he or she uses the old or original authorization factor values to gain access.

If item d) is selected in FMT_SMF.1.1: The evaluator shall verify that the TOE has the functionality to initiate TOE firmware/software updates.

If item e) is selected in FMT_SMF.1.1: If additional management functions are claimed, the evaluator shall verify that the additional features function as described.

Test 2 (conditional): If the TOE provides default authorization factors, the evaluator shall change these factors in the course of taking ownership of the device as described in the operational guidance. The evaluator shall then confirm that the (old) authorization factors are no longer valid for data access.

Test 3 (conditional): If the TOE provides key recovery capability whose effects are visible at the TOE interface, then the evaluator shall devise a test that ensures that the key recovery capability has been or can be disabled following the guidance provided by the vendor.

Test 4 (conditional): If the TOE provides the ability to configure the power saving states that are entered by certain events, the evaluator shall devise a test that causes the TOE to enter a specific power saving state, configure the TSF so that this activity causes a different state to be entered, repeat the activity, and observe the new state is entered as configured.

Test 5 (conditional): If the TOE provides the ability to disable the use of one or more power saving states, the evaluator shall devise a test that enables all supported power saving states and demonstrates that the TOE can enter into each of these states. The evaluator shall then disable the supported power saving states one by one, repeating the same set of actions that were performed at the start of the test, and observe each time that when a power saving state is configured to no longer be used, none of the behavior causes the disabled state to be entered.

Test 1, Option A and B - The procedure for changing the DEK is related to changing the passphrase. In the next test case, FMT_SMF.1-OptionC-t1, the evaluator issues the "rmcctl –D" command which decrypts and erases the DEK. The evaluator then re-enables software encryption with a new passphrase and a new DEK is generated. See the next test for results for an example of the "rmcctl –D" command.

Test 1, Option C – The evaluator enabled software encryption with the passphrase set. The evaluator then enabled the partition and demonstrated the passphrase was valid. The evaluator then decrypted the partition. Next the evaluator re-encrypted the partition using a new passphrase. The evaluator then enabled the partition with the new passphrase and exited. The evaluator then attempted to enable the partition with the old passphrase and was rejected as expected.

Option D - Updates are tested in FPT_TUD_EXT.1

Option E – No additional functions are claimed.

Test 2 – Not Applicable. The TOE does not support default credentials for encryption. When the encryption is invoked the first time, a passphrase must be provided.

Test 3 – Not applicable. The TOE does not support a key recovery capability.

Test 4 - Not applicable. The TOE does not support configuring the power saving state by certain events.

Test 5 – Not applicable. The TOE does not support configuring the power saving states

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

2.3.3.1 FDEEEcPP20E:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Option A: The evaluator shall ensure the TSS describes how the TOE changes the DEK.

Option B: The evaluator shall ensure the TSS describes how the TOE cryptographically erases the DEK.

Option C: The evaluator shall ensure the TSS describes the process to initiate TOE firmware/software updates.

Option D: If additional management functions are claimed in the ST, the evaluator shall verify that the TSS describes those functions.

KMD

Option D: If the TOE offers the functionality to import an encrypted DEK, the evaluator shall ensure the KMD describes how the TOE imports a wrapped DEK and performs the decryption of the wrapped DEK.

See the description for FDEAAcPP20:FMT_SMF.1.1.

Component Guidance Assurance Activities: Option A: The evaluator shall review the AGD guidance and shall determine that the instructions for changing a DEK exist. The instructions must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate or re-generate the DEK.

Option C: The evaluator shall examine the operational guidance to ensure that it describes how to initiate TOE firmware/software updates.

Option D: Default Authorization Factors: It may be the case that the TOE arrives with default authorization factors in place. If it does, then the selection in item D must be made so that there is a mechanism to change these authorization factors. The operational guidance shall describe the method by which the user changes these factors when they are taking ownership of the device. The TSS shall describe the default authorization factors that exist.

Disable Key Recovery: The guidance for disabling this capability shall be described in the AGD documentation.

See the description for FDEAAcPP20:FMT_SMF.1.1.

Component Testing Assurance Activities: Option A and B: The evaluator shall verify that the TOE has the functionality to change and cryptographically erase the DEK (effectively removing the ability to retrieve previous user data).

Option C: The evaluator shall verify that the TOE has the functionality to initiate TOE firmware/software updates.

Option D: If additional management functions are claimed, the evaluator shall verify that the additional features function as described.

See the description for FDEAAcPP20:FMT_SMF.1.1.

2.3.4 Security Roles (FDEAAcPP20E:FMT_SMR.1)

2.3.4.1 FDEAAcPP20E:FMT_SMR.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.2 FDEAAcPP20E:FMT_SMR.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There are no TSS evaluation activities for this SFR. Evaluation of this SFR is performed as part of evaluating FMT_MOF.1 and FMT_SMF.1.

Evaluation of this SFR is performed as part of evaluating FMT_MOF.1 and FMT_SMF.1.

Component Guidance Assurance Activities: There are no guidance evaluation activities for this SFR. Evaluation of this SFR is performed as part of evaluating FMT_MOF.1 and FMT_SMF.1.

Evaluation of this SFR is performed as part of evaluating FMT_MOF.1 and FMT_SMF.1.

Component Testing Assurance Activities: There are no test evaluation activities for this SFR. Evaluation of this SFR is performed as part of evaluating FMT_MOF.1 and FMT_SMF.1.

Evaluation of this SFR is performed as part of evaluating FMT_MOF.1 and FMT_SMF.1.

2.4 Protection of the TSF (FPT)

2.4.1 Protection of Key and Key Material (FDEAAcPP20E:FPT_KYP_EXT.1)

2.4.1.1 FDEAAcPP20E:FPT_KYP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the method by which intermediate keys are generated using submask combining.

KMD

The evaluator shall verify the KMD to ensure it describes the storage location of all keys and the protection of all keys stored in non-volatile memory. The description of the key chain shall be reviewed to ensure the selected method is followed for the storage of wrapped or encrypted keys in non-volatile memory and plaintext keys in non-volatile memory meet one of the criteria for storage.

(TD0458 applied)

Section 6.4 of the ST states that the TOE stores encrypted DEKs in the header of each drive partition.

KMD – The KMD has a table showing that the DEK is stored in the partition header and is protected using AES CBC encryption as identified in the FCS_COP.1(c) requirement

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.2 Protection of Key and Key Material (FDEEEcPP20E:FPT_KYP_EXT.1)

2.4.2.1 FDEEEcPP20E:FPT_KYP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the method by which intermediate keys are generated using submask combining.

KMD

The evaluator shall verify the KMD to ensure it describes the storage location of all keys and the protection of all keys stored in non-volatile memory. The description of the key chain shall be reviewed to ensure the selected method is followed for the storage of wrapped or encrypted keys in non-volatile memory and plaintext keys in non-volatile memory meet one of the criteria for storage.

(TD0458 applied)

See FDEAAcPP20_FPT_KYP_EXT.1.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.3 Power Saving States (FDEAAcPP20E:FPT_PWR_EXT.1)

2.4.3.1 FDEAAcPP20E:FPT_PWR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall validate the TSS contains a list of Compliant power saving states.

Section 6.4 of the ST states that the TOE provides the Compliant power-saving state G3, mechanical off. The TOE enters this state when the user shuts off the device or when the administrator issues the shutdown command. The TOE must be fully rebooted from this state.

Component Guidance Assurance Activities: The evaluator shall ensure that guidance documentation contains a list of Compliant power saving states. If additional power saving states are supported, then the evaluator shall validate that the guidance documentation states how non-Compliant power states are disabled.

The TOE supports power on and power off. The User Guide provides the CLI command for shutting down the TOE.

Component Testing Assurance Activities: The evaluator shall confirm that for each listed compliant state all key/key materials are removed from volatile memory by using the test defined in FCS_CKM.4(d).

See the test result for FDEAAcPP20E:FCS_CKM.4(d).

2.4.4 Power Saving States (FDEEEcPP20E:FPT_PWR_EXT.1)

2.4.4.1 FDEEEcPP20E:FPT_PWR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall validate the TSS contains a list of Compliant power saving states.

See FDEAAcPP20_FPT_PWR_EXT.1.

Component Guidance Assurance Activities: The evaluator shall ensure that guidance documentation contains a list of Compliant power saving states. If additional power saving states are supported, then the evaluator shall validate that the guidance documentation states how the use of non-Compliant power savings states are disabled. (TD0460 applied)

See FDEAAcPP20_FPT_PWR_EXT.1.

Component Testing Assurance Activities: The evaluator shall confirm that for each listed Compliant state all key/key materials are removed from volatile memory by using the test defined in FCS_CKM_EXT.6. (TD0345 applied)

See the test result for FDEAAcPP20E:FCS_CKM.4(d).

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

2.4.5.1 FDEAAcPP20E:FPT_PWR_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall validate that the TSS contains a list of conditions under which the TOE enters a Compliant power saving state.

Section 6.4 of the ST states that the TOE provides the Compliant power-saving state G3, mechanical off. The TOE enters this state when the user shuts off the device or when the administrator issues the shutdown command. The TOE must be fully rebooted from this state.

Component Guidance Assurance Activities: The evaluator shall check that the guidance contains a list of conditions under which the TOE enters a Compliant power saving state. Additionally, the evaluator shall verify that the guidance documentation states whether unexpected power-loss events may result in entry to a non-Compliant power saving state and, if that is the case, validate that the documentation contains information on mitigation measures.

The TOE supports power on and power off. The User Guide provides the CLI command for shutting down the TOE. If the TOE unexpectedly loses power, it transitions to the shutdown state. The TOE does not support any non-Complaint states.

Component Testing Assurance Activities: The evaluator shall trigger each condition in the list of identified conditions and ensure the TOE ends up in a compliant power saving state by running the test identified in FCS_CKM.4(d).

See the test result for FDEAAcPP20E:FCS_CKM.4(d).

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

2.4.6.1 FDEEEcPP20E:FPT_PWR_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall validate that the TSS contains a list of conditions under which the TOE enters a Compliant power saving state.

See FDEAAcPP20_FPT_PWR_EXT.2.

Component Guidance Assurance Activities: The evaluator shall check that the guidance contains a list of conditions under which the TOE enters a Compliant power saving state. Additionally, the evaluator shall verify that the guidance documentation provides information on how long it is expected to take for the TOE to fully transition into the Compliant power saving state (e.g. how many seconds for the volatile memory to be completely cleared).

See FDEAAcPP20_FPT_PWR_EXT.2.

Component Testing Assurance Activities: The evaluator shall trigger each condition in the list of identified conditions and ensure the TOE ends up in a Complaint power saving state by running the test identified in FCS_CKM_EXT.6. (TD0345 applied)

See the test result for FDEAAcPP20E:FCS_CKM.4(d).

2.4.7 TSF Testing (FDEAAcPP20E:FPT_TST_EXT.1)

2.4.7.1 FDEAAcPP20E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the known-answer self-tests for cryptographic functions.

The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TOE and the method by which the TOE tests those functions. The evaluator shall verify that the TSS includes each of these functions, the method by which the TOE verifies the correct operation of the function. The evaluator shall verify that the TSF data are appropriate for TSF Testing. For example, more than blocks are tested for AES in CBC mode, output of AES in GCM mode is tested without truncation, or 512-bit key is used for testing HMAC-SHA-512.

If FCS_RBG_EXT.1 is implemented by the TOE and according to NIST SP 800-90, the evaluator shall verify that the TSS describes health tests that are consistent with section 11.3 of NIST SP 800-90.

If any FCS_COP functions are implemented by the TOE, the TSS shall describe the known-answer self-tests for those functions.

The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TSF, the method by which those functions are tested. The TSS will describe, for each of these

functions, the method by which correct operation of the function/component is verified. The evaluator shall determine that all of the identified functions/components are adequately tested on start-up.

Section 6.4 of the ST identifies that the TOE includes the following power-up Known Answer Tests (KATs) to ensure that each of its cryptographic algorithms operates correctly.

- OpenSSL - ECDSA sign/verify test
- OpenSSL – SHA-384 hashing test
- OpenSSL – integrity test
- kernel – SHA-256 hashing test
- kernel – AES-256 CBC encrypt/decrypt test
- kernel – integrity test
- libgcrypt – SHA hashing tests
- libgcrypt – HMAC-SHA tests
- libgcrypt – AES-256 CBC encrypt/decrypt test
- libgcrypt – SHA-256 HMAC_DRBG test
- libgcrypt – integrity test

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.8 TSF Testing (FDEEEcPP20E:FPT_TST_EXT.1)

2.4.8.1 FDEEEcPP20E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the known-answer self-tests for cryptographic functions.

The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TOE and the method by which the TOE tests those functions. The evaluator shall verify that the TSS includes each of these functions, the method by which the TOE verifies the correct operation of the function. The evaluator shall verify that the TSF data are appropriate for TSF Testing. For example, more than blocks are

tested for AES in CBC mode, output of AES in GCM mode is tested without truncation, or 512-bit key is used for testing HMAC-SHA-512.

If FCS_RBG_EXT.1 is implemented by the TOE and according to NIST SP 800-90, the evaluator shall verify that the TSS describes health tests that are consistent with section 11.3 of NIST SP 800-90.

If any FCS_COP functions are implemented by the TOE, the TSS shall describe the known-answer self-tests for those functions.

The evaluator shall verify that the TSS describes, for some set of non-cryptographic functions affecting the correct operation of the TSF, the method by which those functions are tested. The TSS will describe, for each of these functions, the method by which correct operation of the function/component is verified. The evaluator shall determine that all of the identified functions/components are adequately tested on start-up.

See FDEAAcPP20E:FPT_TST_EXT.1

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.9 Trusted Update (FDEAAcPP20E:FPT_TUD_EXT.1)

2.4.9.1 FDEAAcPP20E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.9.2 FDEAAcPP20E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.9.3 FDEAAcPP20E:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes information stating that an authorized source signs TOE updates and will have an associated digital signature. The evaluator shall examine the TSS contains a definition of an authorized source along with a description of how the TOE uses public keys for the update verification mechanism in the Operational Environment. The evaluator ensures the TSS contains details on the protection and maintenance of the TOE update credentials.

If the Operational Environment performs the signature verification, then the evaluator shall examine the TSS to ensure it describes, for each platform identified in the ST, the interface(s) used by the TOE to invoke this cryptographic functionality.

Section 6.4 of the ST states the TOE can display its current firmware version and has the ability to update its firmware using signed updates. The TOE will verify the signature on a firmware upgrade (using its OpenSSL library in conjunction with the embedded /root/fupdate/cwdts_publickey.pem key to verify the ECDSA P-384 with SHA-384 signature) before installing it, and will reject any update with an invalid signature.

Component Guidance Assurance Activities: The evaluator ensures that the operational guidance describes how the TOE obtains vendor updates to the TOE; the processing associated with verifying the digital signature of the updates (as defined in FCS_COP.1(a)); and the actions that take place for successful and unsuccessful cases.

The fupdate CLI command in the User Guide (sections 12.3 and 7.9) explains the update process. It states the digital signature is verified before an update is performed. If the digital signature cannot be verified, the update operation is aborted. The User Guide instructs the administrator to contact the vendor for updates.

Component Testing Assurance Activities: The evaluators shall perform the following tests (if the TOE supports multiple signatures, each using a different hash algorithm, then the evaluator performs tests for different combinations of authentic and unauthentic digital signatures and hashes, as well as for digital signature alone):

Test 1: The evaluator performs the version verification activity to determine the current version of the TOE. After the update tests described in the following tests, the evaluator performs this activity again to verify that the version correctly corresponds to that of the update.

Test 2: The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that an update successfully installs on the TOE. The evaluator shall perform a subset of other evaluation activity tests to demonstrate that the update functions as expected.

Test 1 - The evaluator logged off the TOE and was able to observe the TOE version at the logon prompt. The evaluator installed an updated from the vendor in test 2. The evaluator observed changed after the update was installed.

Test 2 - The evaluator received an update from the vendor and followed the “Update Software/Firmware” section of the User Guide which describes using the fupdate command. The evaluator copied the update onto the device and then issued fupdate. After the update was installed, the evaluator performed a login to show the version. The evaluator noted the version had incremented. The evaluator then performed the password tests to ensure the product still worked as before.

2.4.10 Trusted Update (FDEEEcPP20E:FPT_TUD_EXT.1)

2.4.10.1 FDEEEcPP20E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.10.2 FDEEEcPP20E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.10.3 FDEEEcPP20E:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes information stating that an authorized source signs TOE updates and will have an associated digital signature. The evaluator shall examine the TSS contains a definition of an authorized source along with a description of how the TOE uses public keys for the update verification mechanism in the Operational Environment. The evaluator ensures the TSS contains details on the protection and maintenance of the TOE update credentials.

If the Operational Environment performs the signature verification, then the evaluator shall examine the TSS to ensure it describes, for each platform identified in the ST, the interface(s) used by the TOE to invoke this cryptographic functionality.

See FDEAAcPP20_FPT_TUD_EXT.1.

Component Guidance Assurance Activities: The evaluator ensures that the operational guidance describes how the TOE obtains vendor updates to the TOE; the processing associated with verifying the digital signature of the updates (as defined in FCS_COP.1(a)); and the actions that take place for successful and unsuccessful cases.

See FDEAAcPP20_FPT_TUD_EXT.1.

Component Testing Assurance Activities: The evaluators shall perform the following tests (if the TOE supports multiple signatures, each using a different hash algorithm, then the evaluator performs tests for different combinations of authentic and unauthentic digital signatures and hashes, as well as for digital signature alone):

Test 1: The evaluator performs the version verification activity to determine the current version of the TOE. After the update tests described in the following tests, the evaluator performs this activity again to verify that the version correctly corresponds to that of the update.

Test 2: The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that an update successfully installs on the TOE. The evaluator shall perform a subset of other evaluation activity tests to demonstrate that the update functions as expected.

See FDEAAcPP20_FPT_TUD_EXT.1.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the FDEEEcPP20E/FDEAAcPP20E that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: The evaluator shall check the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g., perform updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent, is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any supplementary information required by the cPP for aspects such as entropy analysis or cryptographic key management architecture1: no additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities. The interfaces that need to be evaluated are also identified by reference to the assurance activities listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any supplementary information required by the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the Evaluation Activities for each SFR also means that the tracing required in ADV_FSP.1.2D is treated as implicit, and no separate mapping information is required for this element.

However, if the evaluator is unable to perform some other required Evaluation Activity because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

The assurance activities from Supporting Documents of FDEEEcPP20/FDEAAcPP20 have been performed. The evaluator concluded adequate information was provided and the analysis of the evaluator is documented in the previous sections of this document.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: The evaluator shall check the requirements below are met by the operational guidance.

Operational guidance documentation shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

Operational guidance must be provided for every Operational Environment that the TOE supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target. This may be contained all in one document.

The contents of the operational guidance will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR.

In addition to SFR-related Evaluation Activities, the following information is also required.

- The operational guidance shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

- The TOE will likely contain security functionality that does not fall under the scope of evaluation under this cPP. The operational guidance shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

See the previous assurance activities summaries for a description of how the User Guide meets the requirements. The cryptographic engine does not need any configuration so the User Guide does not need any instructions to address it. The TOE is the software encryption part of the product and that is called out in 5.4.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: The evaluator shall check the requirements below are met by the preparative procedures.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR.

Preparative procedures shall be distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 Evaluation Activities for SFRs.

In addition to SFR-related Evaluation Activities, the following information is also required.

Preparative procedures must include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target). The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE itself).

Preparative procedures must be provided for every Operational Environment that the TOE supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target. This may be contained all in one document.

The preparative procedures must include

- instructions to successfully install the TSF in each Operational Environment; and

- instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and

- instructions to provide a protected administrative capability.

Section 5.4 explains how to configure the TOE. It has step by step instructions that the evaluator was able to use when setting up the TOE.

3.3 Life-cycle support (ALC)

The FDEEEcPP20/FDEAAcPP20 do not contain any specific AAs for the Life-Cycle Support assurance class.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

The evaluator shall prepare a test plan that covers all of the testing actions for ATE_IND.1 in the CEM and in the SFR-related Evaluation Activities. While it is not necessary to have one test case per test listed in an Evaluation

Activity, the evaluator must show in the test plan that each applicable testing requirement in the SFR-related Evaluation Activities is covered.

The test plan identifies the platforms to be tested, and for any platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition and configuration of each platform to be tested, and any setup actions that are necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of any cryptographic engine to be used (e.g. for cryptographic protocols being evaluated).

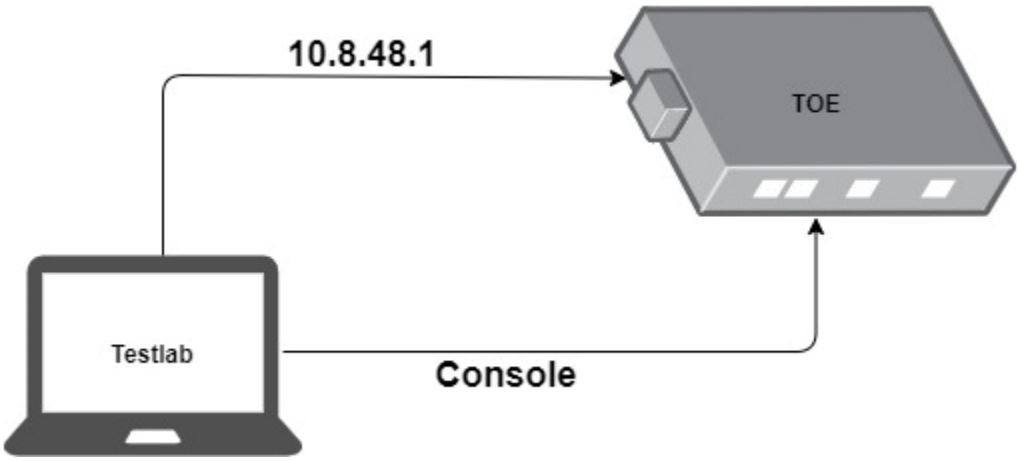
The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives, and the expected results.

The test report (which could just be an updated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure, so that a fix was then installed and then a successful re-run of the test was carried out, then the report would show a 'fail' result followed by a 'pass' result (and the supporting details), and not just the 'pass' result*.

*It is not necessary to capture failures that were due to errors on the part of the tester or test environment. The intention here is to make absolutely clear when a planned test resulted in a change being required to the originally specified test configuration in the test plan, to the evaluated configuration identified in the ST and operational guidance, or to the TOE itself.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results.

The TOE was made available at the Gossamer testing laboratory. When performing testing, the evaluator configured the TOE into CC mode as described in the User Guide. The following diagram shows the test configuration:



The evaluator used the following test tools:

- Windows 10,64-bit edition
- SSH Client – Putty version 0.74
- Standard Windows utilities (e.g., notepad, snip tool)
- HxD (Hexeditor) version 2.0 (used to examine dumped memory files)
- Gossamer developed test tools for binary searches

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components apply to all systems claimed in the ST, and should identify at a minimum the processors used by the TOE. Software components include any libraries used by the TOE, such as cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

The evaluator shall examine the documentation outlined below provided by the vendor to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

In addition to the activities specified by the CEM in accordance with Table 2 above, the evaluator shall perform the following activities.

The evaluator formulates hypotheses in accordance with process defined in Appendix A.1. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix

A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

The evaluator shall perform the CEM activity as specified, unless otherwise indicated:

AVA_VAN.1-1 The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.*

AVA_VAN.1-2 The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

AVA_VAN.1-3 Replace CEM work unit with activities outlined in Appendix A, Section A.1.

AVA_VAN.1-4 Replace the CEM work unit with the analysis activities on the list of potential vulnerabilities in Appendix A, section A.1, and documentation as specified in Appendix A, Section A.3.

AVA_VAN.1-5 Replace the CEM work unit with the activities specified in Appendix A, section A.2.

AVA_VAN.1-6 The CEM work unit is captured in Appendix A, Section A.3; there are no substantive differences.

AVA_VAN.1-7 The evaluator shall conduct penetration testing.

AVA_VAN.1-8 The evaluator shall record the actual results of the penetration tests.

AVA_VAN.1-9 Replace the CEM work unit with the reporting called for in Appendix A, Section A.3.

AVA_VAN.1-10 This work unit is not applicable for Type 1 and Type 2 flaws (as defined in Appendix A, Section A.1), as inclusion in this Supporting Document by the iTC makes any confirmed vulnerabilities stemming from these flaws subject to an attacker possessing a Basic attack potential. This work unit is replaced for Type 3 and Type 4 flaws by the activities defined in Appendix A, Section A.3.

AVA_VAN.1-11 Replace the CEM work unit with the reporting called for in Appendix A, Section A.3.

*If the iTC specifies any tools to be used in performing this analysis in section A.3.4, the following text is also included in this cell: 'The calibration of test resources specified in paragraph 1418 of the CEM applies to the tools listed in Appendix A, Section A.1.4.'

3.5.1.1 cPP Sourced Hypotheses

The FDEAAcPP20 has the following two flaw hypotheses:

1. In order to validate the AA is properly encrypting keying material (e.g., BEV, KEK, authorization submasks) in the readable part of the disk (e.g., shadow MBR), the evaluator should examine the disk using a tool to view the drive (e.g. WinHex) to look for material that exposes a key value.

2. When an authentication or recovery credential is changed, it is critical that the AA does not leave old keys/key chains/key material around. This process should also be monitored using a tool to view the drive

To address item 1, the evaluator performed this test as part of FCS_CKM.4(d) test 2.

To address item 2, the evaluator changed the passphrase and then searched for the key material again. No key material was found on the disk as expected.

The FDEEEcPP20 has the following one flaw hypothesis:

- During the software encryption installation process, it is possible that that encryption is interrupted (e.g., power is removed, etc.). The evaluator should verify that when the software encryption resumes and completes, that all of the user data is encrypted.

The evaluator re-partitioned the drive to include a smaller partition to use for searching. The evaluator then attempted to encrypt the smaller partition but power cycled the device before it could complete. The evaluator then encrypted the smaller partition since the status was not encrypted from the previous attempt. The evaluator wrote known data to the smaller partition. The evaluator then copied the smaller partition off and searched for the known data and was unsuccessful.

3.5.1.2 Public Search

The evaluator searched the National Vulnerability Database (https://web.nvd.nist.gov/vuln/search), Vulnerability Notes Database (http://www.kb.cert.org/vuls on 3/13/2023 with the following search terms: "disk encryption", "drive encryption", "key destruction", "key sanitization", "Opal management software", "SED management software", "Password caching", "Key caching", "Curtiss Wright", "DTS1", "Defense Solutions Data Transport System", "Linux Unified Key Setup", "LUKS", "Libgcrypt", "openssl", "CentOS", "kernel cryptography".