PP-Module for File Encryption

Version: 1.0

2019-07-25

National Information Assurance Partnership

Revision History

Version	Date	Comment
v 1.0	2019-07-25	Initial Release

Contents

1 Introduction

1.1 Overview

The scope of the File Encryption PP-Module is to describe the security functionality of a file encryption product in terms of [CC] and to define functional and assurance requirements for such products. This PP-Module is intended for use with the following Base-PP

- Application Software Protection Profile, Version 1.3

This Base-PP is valid because a file encryption product is a 3rd party application or application included with a operating system.

File encryption is the process of encrypting individual files or sets of files (or volumes, or containers, etc.) on an end user device and permitting access to the encrypted data only after proper authentication is provided. Encryption products that conform to this PP-Module must render information inaccessible to anyone (or, in the case of other software on the machine, anything) that does not have the proper authentication credential. The encrypted files may be on a local machine or may be sent to other devices.

The foremost security objective of file encryption is to force an adversary to perform a cryptographic exhaust against a prohibitively large key space. Technology is changing at a rapid rate and the definition of mobile devices and traditional laptop/PC devices is quickly merging. Requirements can diverge slightly for Mobile vs Laptop/PC and the Evaluation Activities will describe any differences. Either of these use cases may be an enterprise managed file encryption client. Some of the security functionality may be provided by the OE. The vendor is required to provide configuration guidance (AGD_PRE, AGD_OPE) to correctly install and administer the TOE for every operational environment supported.

The data that is to be secured by the encryption product is encrypted using a File Encryption Key (FEK). A file encryptor may have zero or more Key Encryption Keys (KEKs) that protect (encrypt) the FEK. The number of keys and the types of keys may vary, but the design should follow one of the following models:

1. Condition a Password/Passphrase directly into a FEK
2. Condition a Password/Passphrase into a KEK that is used to encrypt the randomly generated FEK directly or through a chaining of more than one KEK (these KEKs would be randomly generated).
3. Use a software certificate or an external token to protect the FEK.

From a terminology standpoint, a KEK is either a symmetric key or an asymmetric key pair, and is used for both encryption and decryption of the FEK. If a distinction needs to be made between the public key (which encrypts the FEK) and the private key (which decrypts the FEK), this is done in the requirements and the evaluation activities.

The TOE may be capable of supporting multiple users with different authorization factors, such that different users are able to use the same platform and not be able to read each other's encrypted files. The TOE may also support the ability for users to share an encrypted file without sharing an authorization factor, but this is not required. In order to claim this capability, the TOE must allow sharing of at least one encrypted resource among different users of the TOE who possess different authorization factors (e.g., two different smart cards, two different passwords, one using a password and another using a smart card). If this capability is supported, then the ST author adds FIA_FCT_EXT.1.1.

Authorization
One or more authorization factors must be established before data can be encrypted. This authorization factor(s) must be presented to the file encryption product in order for the user to request that the product decrypt the data. Authorization factors may be uniquely associated with individual users or may be associated with a community of users. The TOE is not required to support multiple types of authorization factors (e.g., both passphrases and external authorization factors). If the ST author defines additional authorization factors, they must be fully documented and cannot diminish the strength of the passphrase and/or external token authorization factors.

The password/passphrase authorization factors must be conditioned such that they are at least the same size (bit length) as the key they are protecting. While this PP-Module does not dictate how these authentication factors are created, a good operational practice is for an administrator to generate the password or passphrase to ensure sufficient entropy. Passphrases are preferred over passwords, since it is easier for users to remember and type in a sequence of words than recall a password and type in a long string of random characters.

Administration
The base requirements of the TOE do not require the TOE to maintain an administrative role. Typically, administrators possess privilege to invoke functionality on the TOE that is not available to general users. For stand alone file encryption products, however, once the product is installed there should be little need for administrative involvement. For enterprise managed file encryption products, the TOE may be remotely administered.

Data Authentication (optional)
Because modification of ciphertext data for certain modes of encryption will enable unidentified plaintext manipulation, care must be taken by the TOE to mitigate against forged or maliciously modified ciphertext data. The PP-Module defines requirements for how the TOE must provide data authentication services, allowing the TOE to implement authenticated block cipher, keyed hash function or asymmetric signing features. Depending on the implementation, the TOE will be responsible for meeting at least one of the aforementioned requirements. In all cases, unsuccessful authentication of the data should not allow the user to see the decrypted ciphertext and notification should be provided to the user if such an event were to occur.

A keyed hashing service may also be used to accomplish data authentication. This will involve using an approved keyed hashing service in accordance with FCS_COP.1(4) and proper protection of the File Authentication Key (FAK); the FAK being the secret value used as input to the keyed hash function. FAKs should be numerically different from the FEK, but will be protected in all of the same manners as the FEK. The primary requirement dictating implementation of data authentication using a keyed hash function is FDP_AUT_EXT.2.

Lastly, asymmetric signing in conjunction with a secure hash function may be used to authenticate the data. The implementation must use an approved signing algorithm in accordance with FCS_COP.1(3) (from the [AppPP]) and an approved secure hashing function in accordance with FCS_COP.1(2) (from the [AppPP]). The primary requirement addressing data authentication via asymmetric signing is FDP_AUT_EXT.3.

The TOE and Its Supporting Environment:
Since the TOE is purely a software solution, it must rely on the TOE Operational Environment (system hardware, firmware, and operating system) for its execution domain and its proper usage. The vendor is expected to provide sufficient installation and configuration instructions (for each platform listed in the ST) to identify Operational Environment(s) with the necessary features and to provide instructions for how to configure it correctly and securely.

The PP-Module contains requirements (Section 5 that must be met by either the TOE or the platform on which it operates. A "platform" is defined as a separate entity whose functions may be used by the TOE, but is not part of the TOE. A third-party library used by the TOE is not considered part of the TOE's "platform", but (for instance) cryptographic functionality that is built into an Operating System on which the TOE executes can be considered part of the platform.

Likewise, an external entity (such as a smart card) that performs cryptographic operations with respect to the FEK would also be considered a part of the "TOE Platform".

The ST author will make the appropriate selection based on where that element is implemented. It is allowable for some elements in a component to be implemented by the TOE, while other elements in that same component may be implemented by the platform; in these cases, further guidance is given in the application notes and Supporting Documentation.

In some cases, the TOE vendor will have to provide specific configuration guidance for the Operational Environment to enable the TOE to meet its security objectives. These include:
For non-mobile systems:

- Instructions for how to configure the operational environment so that the system powers down completely after a period of user inactivity for every operating system that the product supports.
- Instructions for how to disable power managed state (e.g., hibernate/sleep) capabilities.

For mobile systems:

- Instructions for how to configure the operational environment to provide necessary behavior in support of TOE functionality when transition to a locked state after inactivity period and manually engaging the lock functionality.

It should be noted that if the TOE possesses the capability to correctly protect information in one or more of an underlying platform's power managed modes, they can use the FDP_PM_EXT.1 requirement in Appendix A.

Authorized users of the TOE are those users possessing valid authorization factors for the TOE. While some of these functions specified in the PP-Module might be considered "administrative" functions for other types of TOEs, for file encryption products it is the expectation that all of these functions can be performed by the end user of the software.

1.2 Terms

1.2.1 Common Criteria Terms

Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.
Protection Profile Configuration (PP-Configuration)	A comprehensive set of security requirements for a product type that consists of at least one Base-PP and at least one PP-Module.
Protection Profile Module (PP-Module)	An implementation-independent statement of security needs for a TOE type complementary to one or more Base Protection Profiles.
Security Assurance Requirement (SAR)	A requirement to assure the security of theTOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by theTOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation. In this case, file encryption software and its supporting documentation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how aTOE satisfies the SFRs in aST.

1.2.2 Technical Terms

Administrator	Authorized Users with higher privileges and typically handle configuration and management functions, such as configuring and updating the TOE.
Authorization factor (AF)	A value submitted by the user, present on the host, or present on a separate protected hardware physical device used to establish that the user (and potentially the host) is in the community authorized to use the TOE. The authorization factors are used to generate the KEK. Note that these AFs are not used to establish the particular identity of the user.
Authorized	A user who has been provided Authorization factors by the administrator to use the TOE.

User	
Data Encryption	The process of encrypting all user data written to volatile memory.
Deterministic Random Bit Generator (DRBG)	A cryptographic algorithm that produces a sequence of bits from a secret initial seed value. Without knowledge of the seed value, the output sequence should be unpredictable up to the security level of the DRBG.
Entropy Source	This cryptographic function provides a seed for a random bit generator by accumulating the outputs from one or more noise sources. The functionality includes a measure of the minimum work required to guess a given output and tests to ensure that the noise sources are operating properly.
File/Set of files	The user data that is selected to be encrypted, which can include individual file encryption (with a FEK per file) or a set of files encrypted with a single FEK.
File Authentication Key (FAK)	The secret value used as input when a keyed hash function is used to perform data authentication.
File Encryption Key (FEK)	The key that is used by the encryption algorithm to encrypt the selected user data on the host machine.
Key Chaining	The method of using multiple layers of encryption keys to protect data. A top layer key encrypts a lower layer key which encrypts the data; this method can have any number of layers.
Key Encryption Key (KEK)	The key that is used to encrypt another key.
Keying Material	The KEK, FEK, authorization factors and random numbers or any other values from which keys are derived.
Key Sanitization	A method of sanitizing encrypted data by securely overwriting the key, as described in the key destruction requirement, that was encrypting the data.
Noise Source	The component of an RBG that contains the non-deterministic, entropy-producing activity.
Operational Environment	Hardware and software that are outside the TOE boundary that support the TOE functionality and security policy, including the platform, its firmware, and the operating system.
Password	A short string of characters used for authorization to the data on the device.
Passphrase	A string of words that may be used for authorization to the data on the device.
Primary Key Chain	The direct key chain from the authorization factor to the FEK.
Random Bit Generator (RBG)	A cryptographic function composed of an entropy source and DRBG that is invoked for random bits needed to produce keying material.
Sensitive Data	Any data of which the compromise with respect to loss, misuse, or unauthorized access to or modification of could adversely affect the interest of the TOE user.
Shutdown	Power down or unintentional loss of power of the TOE or platform.
Supplemental Key Chain	Other key chains that add protection or functionality without compromising the security of the primary key chain.
System files	Files that reside on the host machine that are used in the operation of the file encryption software.
Temporary File	A file created by an application for short term storage of sensitive data.
Trusted Host	Source/destination host configured and maintained to provide the TOE with appropriate IT security commensurate with the value of the user data protected by the TOE.
Unauthorized User	A user who has not been authorized to use the TOE and decrypt encrypted user data.
User Data	All data that originate on the host, or is derived from data that originate on the host, excluding system files and signed firmware updates from the TOE manufacturer.
Volatile memory	Memory that loses its content when power is turned off.
Zeroize	This term is used to make a distinction between dereferencing a memory location and actively overwriting it with a constant. Keying material needs to be overwritten when it is no longer needed.

1.3 Compliant Targets of Evaluation

This PP-Module specifically addresses encryption of a set of data. This PP-Module addresses the primary threat that an unauthorized user will obtain access to a host machine containing encrypted information and be able to extract the sensitive data through the process of decryption. The Target of Evaluation (TOE) defined in this PP-Module is an encryption product that will inherently encrypt all of that data that the user selects to encrypt. For ease of explanation, "file" will frequently be used to refer to the object that is encrypted (however, it could be any number of things - folders, volumes, containers, etc.).

1.3.1 TOE Boundary

The application, which consists of the software provided by its vendor, is installed onto the platform(s) it operates on. It executes on the platform, which may be an operating system, hardware environment, a software based execution environment, or some combination of these. Those platforms may themselves run within other environments, such as virtual machines or operating systems, that completely abstract away the underlying hardware from the application. The TOE is not accountable for security functionality that is implemented by platform layers that are abstracted away. Some evaluation activities are specific to the particular platform on which the application runs, in order to provide precision and repeatability. The only platforms currently recognized by [AppPP] and this module are those specified in SFR Evaluation Activities. To test on a platform for which there are no EAs, a Vendor should contact NIAP with recommended EAs. NIAP will determine if the proposed platform is appropriate for the PP and accept, reject, or develop EAs as necessary in coordination with the technical community.

The TOE includes any software in the application installation package, even those pieces that may extend or modify the functionality of the underlying platform, such as kernel drivers. Some platforms come bundled with file encryption product and these too should be considered subject to the requirements defined in this document although the expectation of formal Common Criteria evaluation depends upon the national scheme. BIOS and other firmware, the operating system kernel, and other systems software (and drivers) provided as part of the platform are outside the scope of this document.

1.4 Use Cases

[USE CASE 1] Unmanaged Endpoint
: The traditional ability to encrypt files without external management and power down the machine and know the data is securely protected.

[USE CASE 2] Managed Endpoint
: The traditional ability to encrypt files and power down the machine and know the data is securely protected, while communicating with an Enterprise Management server.

[USE CASE 3] Encrypted Distribution
: The ability to encrypt a file on a machine and then send the encrypted file securely using a non-encrypted data in transit method.

2 Conformance Claims

Conformance Statement

This PP-Module inherits exact conformance as required from the specified Base-PP and as defined in the [CC] and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs (dated May 2017). This PP-Module is conformant to Parts 2 (extended) and 3 (extended) of Common Criteria Version 3.1, Revision 5 [CC]. The following PPs and PP-Modules are allowed to be specified in a PP-Configuration with this PP-Module.

- PP-Module for VPN Client, Version 2.1
- PP-Module for File Encryption Enterprise Management, Version 1.0

If claiming compliance to a PP-Configuration that includes multiple PP-Modules, the ST author must ensure any duplicative SFRs are iterated using unique identifiers. This will allow the reader to easily determine which iteration applies to each TOE component

3 Security Problem Description

The primary asset that is being protected is the sensitive user data stored on a system. The threat model thus focuses on a host machine that has been compromised by an unauthorized user. This section addresses threats to the TOE only.

3.1 Threats

A threat consists of a threat agent, an asset, and an adverse action of that threat agent on that asset. The model in this PP-Module only addresses risks that arise from the host machine being compromised by an unauthorized user.

For this PP-Module, the TOE is not expected to defend against all threats related to malicious software that may reside in user data files. For instance, the TOE is not responsible for detecting malware in the data selected by the user for encryption (that is a responsibility of the host environment). Once the file encryption product is operational in a host system, the threats against the data from potentially malicious software on the host are also not in the threat model of this PP-Module. For example, there are no requirements in this PP-Module addressing a malicious host capturing a password-based authorization factor, nor a malicious process reading the memory of an application program that is operating on a decrypted file.

Note that this PP-Module does not repeat the threats identified in the [AppPP], though they all apply given the conformance and hence dependence of this PP-Module on the [AppPP].

Note also that while the [AppPP] contains only threats to the ability of the TOE to provide its security functions, this PP-Module focuses on threats to resources in the operational environment. Together the threats of [AppPP] and those defined in this PP-Module define the comprehensive set of security threats addressed by a file encryption TOE.

T.UNAUTHORIZED_DATA_ACCESS
: **Unauthorized Data Access:** An attacker has access to an account that is not permitted to decrypt files or has no access and uses forensic tools for examination.

T.MANAGEMENT_ACCESS
: **Management Access:** An authorized user may perform sensitive management functions without authorization or a legitimate user may lack the ability to perform necessary security operations due to a lack of supported management functionality.

T.KEYING_MATERIAL_COMPROMISE
: **Compromise of Keying Material:** An attacker exploits a weakness in the random number generation, plaintext keys, and other keying material to decrypt an encrypted file.

T.UNSAFE_AUTHFACTOR_VERIFICATION
: **Flawed Authentication Factor Verification:** An attacker exploits a flaw in the validation or conditioning of the authorization factor.

T.KEYSPACE_EXHAUST
: **Brute Force Attack:** An attacker is able to brute force the keyspace of the algorithms used to force disclosure of sensitive data.

T.PLAINTEXT_COMPROMISE
: **Plaintext Compromise:** An attacker is able to uncover plaintext remains with forensic tools.

3.2 Assumptions

These assumptions are made on the Operational Environment in order to be able to ensure that the security functionality specified in the PP-Module can be provided by the TOE. If the TOE is placed in an Operational Environment that does not meet these assumptions, the TOE may no longer be able to provide all of its security functionality.

A.AUTH_FACTOR
: An authorized user will be responsible for ensuring that all externally derived authorization factors have sufficient strength and entropy to reflect the sensitivity of the data being protected. This can apply to password- or passphrase-based, ECC CDH, and RSA authorization factors.

A.EXTERNAL_FEK_PROTECTION
: External entities that implement ECC CDH or RSA that are used to encrypt and decrypt a FEK have the following characteristics:
- meet national requirements for the cryptographic mechanisms implemented
- require authentication via a pin or other mechanisms prior to allowing access to protected information (the decrypted FEK, or the private key)
- implement anti-hammer provisions where appropriate (for example, when a pin is the authentication factor).

A.SHUTDOWN
: An authorized user will not leave the machine in a mode where sensitive information persists in non-volatile storage.

A.STRONG_OE_CRYPTO
: All cryptography implemented in the Operational Environment and used by the TOE will meet the requirements listed in this PP-Module. This includes generation of external token authorization factors by a RBG.

A.FILE_INTEGRITY
: When the file is in transit, it is not modified, otherwise if that possibility exists, the appropriate selections in Appendix B are chosen for Data Authentication.

3.3 Organizational Security Policies

There are no Organizational Security Policies for the PP-Module.

4 Security Objectives

The Security Problem described in Section 3 Security Problem Description will be addressed by a combination of cryptographic capabilities. Compliant TOEs will provide security functionality that addresses threats to the TOE. The following subsections provide a description of the security objectives required to meet the threats previously discussed. The description of these security objectives are in addition to that described in the [AppPP].

4.1 Security Objectives for the TOE

O.KEY_MATERIAL_PROTECTION

Protection of Key Material: The TOE must ensure that sensitive plaintext key material used in performing its operations is cleared once it is no longer needed. Key material must be identified; its use and intermediate storage areas must also be identified; and then those storage areas must be cleared in a timely manner and without interruptions. For example, authorization factors are only needed until the KEK is formed; at that point, volatile memory areas containing the authorization factors should be cleared.

Addressed by: FCS_CKM_EXT.4, FPT_KYP_EXT.1

O.FEK_SECURITY

Encryption Using a Strong FEK and KEK: In order to ensure that brute force attacks are infeasible, the TOE must ensure that the cryptographic strength of the keys and authorization factors used to generate and protect the keys is sufficient to withstand attacks in the near-to-mid-term future. Password/passphrase conditioning requirements are also levied to help ensure that a brute force attack against these authorization factors (when used) has a similar level of resistance.

Addressed by: FCS_COP.1(1) (from Base-PP), FCS_CKM_EXT.2, FCS_IV_EXT.1, FCS_KYC_EXT.1, FCS_VAL_EXT.1, FCS_CKM_EXT.3 (selection-based), FCS_CKM_EXT.6 (selection-based), FCS_COP.1(5) (selection-based), FCS_COP.1(6) (selection-based), FCS_COP.1(7) (selection-based), FCS_KDF_EXT.1 (selection-based), FCS_SMC_EXT.1 (selection-based), FCS_VAL_EXT.2 (selection-based)

O.WIPE_MEMORY

Removal of Plaintext Data: To address the threat of unencrypted copies of data being left in non-volatile memory or temporary files where it may be accessed by an unauthorized user, the TOE will ensure that plaintext data it creates is securely erased when no longer needed. The TOE's responsibility is to utilize the appropriate TOE platform method for secure erasure, but the TOE is not responsible for verifying that the secure erasure occurred as this will be the responsibility of the TOE platform.

Addressed by: FDP_PRT_EXT.1, FDP_PRT_EXT.2, FDP_PRT_EXT.3 (optional)

O.PROTECT_DATA

Protection of Data: The TOE will encrypt data to protect the data from unauthorized access. Encrypting the file or set of files will protect the user data even when low-level tools that bypass operating system protections such as discretionary and mandatory access controls are available to an attacker. Users that are authorized to access the data must provide authorization factors to the TOE in order for the data to be decrypted and provided to the user.

The TOE will also optionally include data authentication functionality to protect data from unauthorized modification.

Addressed by: FCS_COP.1(1) (from Base-PP), FCS_IV_EXT.1, FDP_PRT_EXT.1, FDP_PRT_EXT.2, FCS_CKM_EXT.5 (optional), FCS_COP_EXT.1 (optional), FDP_AUT_EXT.1 (optional), FDP_AUT_EXT.2 (optional), FDP_AUT_EXT.3 (optional), FDP_PM_EXT.1 (optional), FDP_PRT_EXT.3 (optional), FIA_FCT_EXT.1 (optional)

O.SAFE_AUTHFACTOR_VERIFICATION

Safe Authentication Factor Verification: In order to avoid exposing information that would allow an attacker to compromise or weaken any factors in the chain keys generated or protected by the authorization factors, the TOE will verify the valid authorization factor prior to the FEK being used to decrypt the data being protected.

Addressed by: FCS_VAL_EXT.1, FIA_AUT_EXT.1

O.MANAGE

The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.

Addressed by: FMT_MEC_EXT.1, FMT_SMF.1(2)

4.2 Security Objectives for the Operational Environment

The Operational Environment of the TOE implements technical and procedural measures to assist the TOE in correctly providing its security functionality (which is defined by the security objectives for the TOE). The security objectives for the Operational Environment consist of a set of statements describing the goals that the Operational Environment should achieve. This section defines the security objectives that are to be addressed by the IT domain or by non-technical or procedural means. The assumptions identified in Section 3 are incorporated as security objectives for the environment.

OE.AUTHORIZATION_FACTOR_STRENGTH

An authorized user will be responsible for ensuring that all externally derived authorization factors have sufficient strength and entropy to reflect the sensitivity of the data being protected. This can apply to password or passphrase based, ECC CDH, and RSA authorization factors.

OE.POWER_SAVE

The non-mobile operational environment must be configurable so that there exists at least one mechanism that will cause the system to enter a safe power state (A.SHUTDOWN). Any such mechanism (e.g., sleep, hibernate) that does not conform to this requirement must be capable of being disabled. The mobile operational environment must be configurable such that there exists at least one mechanism that will cause the system to lock upon a period of time.

OE.STRONG_ENVIRONMENT_CRYPTO

The Operating environment will provide a cryptographic function capability that is commensurate with the requirements and capabilities of the TOE.

4.3 Security Objectives Rationale

This section describes how the assumptions, threats, and organization security policies map to the security objectives.

Threat, Assumption, or OSP	Security Objectives	Rationale
T.UNAUTHORIZED_DATA_ACCESS	O.PROTECT_DATA	The threat T.UNAUTHORIZED_DATA_ACCESS i countered by O.PROTECT_DATA as tl provides for encryption of data.
T.MANAGEMENT_ACCESS	O.MANAGE	The threat T.UNAUTHORIZED_DATA_ACCESS i countered by O.MANAGE as this ensu proper management functionality.
T.KEYING_MATERIAL_COMPROMISE	O.KEY_MATERIAL_PROTECTION	The threat T.KEYING_MATERIAL_COMPROMIS countered by O.KEY_MATERIAL_PROTECTION as provides for protection of keys.
T.UNSAFE_AUTHFACTOR_VERIFICATION	O.SAFE_AUTHFACTOR_VERIFICATION	The threat T.UNSAFE_AUTHFACTOR_VERIFIC is countered by O.SAFE_AUTHFACTOR_VERIFICATI this provides for properly supported authentication factors.
T.KEYSPACE_EXHAUST	O.FEK_SECURITY	The threat T.KEYSPACE_EXHAUST is countered by O.FEK_SECURITY as thi makes brute force attacks infeasible.
T.PLAINTEXT_COMPROMISE	O.WIPE_MEMORY	The threat T.PLAINTEXT_COMPROMI countered by O.WIPE_MEMORY as th provides data cleanup.
A.AUTH_FACTOR	OE.AUTHORIZATION_FACTOR_STRENGTH	The operational environment objective OE.AUTHORIZATION_FACTOR_STR is realized through A.AUTH_FACTOR.
A.EXTERNAL_FEK_PROTECTION	OE.STRONG_ENVIRONMENT_CRYPTO	The operational environment objective OE.STRONG_ENVIRONMENT_CRYP realized through A.EXTERNAL_FEK_PROTECTION.
A.SHUTDOWN	OE.POWER_SAVE	The operational environment objective OE.POWER_SAVE is realized through A.SHUTDOWN.
A.STRONG_OE_CRYPTO	OE.STRONG_ENVIRONMENT_CRYPTO	The operational environment objective OE.STRONG_ENVIRONMENT_CRYP realized through A.STRONG_OE_CRY
A.FILE_INTEGRITY	OE.STRONG_ENVIRONMENT_CRYPTO	The operational environment objective OE.STRONG_ENVIRONMENT_CRYP realized through A.STRONG_OE_CRY

5 Security Requirements

This chapter describes the security requirements which have to be fulfilled by the product under evaluation. Those requirements comprise functional components from Part 2 and assurance components from Part 3 of [CC]. The following notations are used:

- **Refinement** operation (denoted by **bold text** or ~~strikethrough text~~): is used to add details to a requirement (including replacing an assignment with a more restrictive selection) or to remove part of the requirement that is made irrelevant through the completion of another operation, and thus further restricts a requirement.
- **Selection** (denoted by *italicized text*): is used to select one or more options provided by the [CC] in stating a requirement.
- **Assignment** operation (denoted by *italicized text*): is used to assign a specific value to an unspecified parameter, such as the length of a password. Showing the value in square brackets indicates assignment.
- **Iteration** operation: are identified with a number inside parentheses (e.g. "(1)")

5.1 App PP Security Functional Requirements Direction

The TOE is expected to rely on some of the security functions implemented by the application as a whole and evaluated against [AppPP]. The following section describe any modifications that the ST author must make to the SFRs defined in the Base-PP in addition to what is mandated by section 5.2.

5.1.1 Modified SFRs

This PP-Module does not modify any SFRs defined by the App PP.

5.2 TOE Security Functional Requirements

The following section describes the SFRs that must be satisfied by any TOE that claims conformance to this PP-Module. These SFRs must be claimed regardless of which PP-Configuration is used to define the TOE.

5.2.1 Cryptographic Support (FCS)

FCS_CKM_EXT.2 File Encryption Key (FEK) Generation

FCS_CKM_EXT.2.1 The TSF shall [**selection**:

- *accept FEK from an enterprise management server*,
- *generate FEK cryptographic keys*
  *[**selection**:*
  - *using a Random Bit Generator as specified in FCS_RBG_EXT.1* ***(from [AppPP])*** *and with entropy corresponding to the security strength of AES key sizes of [**selection**: 128 bit, 256 bit],*
  - *derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6*

  ]

].

FCS_CKM_EXT.2.2 The TSF shall use a unique FEK for each file (or set of files) using the mechanism on the client as specified in FCS_CKM_EXT.2.1.

FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [**selection**:

- *For volatile memory, the destruction shall be executed by a [**selection**:*
  - *single overwrite consisting of [**selection**: a pseudo-random pattern using the TSF's RBG, zeroes, ones, new value of a key, [**assignment**: any value that does not contain any CSP]] ,*
  - *removal of power to the memory,*
  - *destruction of reference to the key directly followed by a request for garbage collection*

  ] ,
- *For non-volatile memory, the destruction shall be executed by [**selection**:*
  - *destruction of all KEKs protecting the target key, where none of the KEKs protecting the target key are derived ,*
  - *the invocation of an interface provided by the underlying platform that [**selection**:*
    - *logically addresses the storage location of the key and performs a [**selection**: single, [**assignment**: ST author defined multi-pass]] overwrite consisting of [**selection**: a pseudo-random pattern using the TSF's RBG, zeroes, ones, new value of a key, [**assignment**: any value that does not contain any CSP]] ,*
    - *instructs the underlying platform to destroy the abstraction that represents the key*

    ]

  ]

].

Application Note: The interface referenced in the requirement could take different forms, the most likely of which is an application programming interface to an OS kernel. There may be various levels of abstraction visible. For instance, in a given implementation that overwrites a key stored in non-volatile memory, the application may have access to the file system details and may be able to logically address specific memory locations. In another implementation that instructs the underlying platform to destroy the representation of a key stored in non-volatile memory, the application may simply have a handle to a resource and can only ask the platform to delete the resource, as may be the case with a platforms secure key store. The latter implementation should only be used for the most restricted access. The level of detail to which the TOE has access will be reflected in the TSS section of the ST. Several selections allow assignment of a 'value that does not contain any CSP'. This means that the TOE uses some other specified data not drawn from a source that may contain key material or reveal information about key material, and not being any of the particular values listed as other selection options. The point of the phrase 'does not contain any CSP' is to ensure that the overwritten data is carefully selected, and not taken from a general 'pool' that might contain current or residual data that itself requires confidentiality protection.

For the selection "destruction of all KEKs protecting target key, where none of the KEKs protecting the target key are derived", a key can be considered destroyed by destroying the key that protects the key. If a key is wrapped or encrypted it is not necessary to "overwrite" that key, overwriting the key that is used to wrap or encrypt the key used to encrypt/decrypt data, using the appropriate method for the memory type involved, will suffice. For example, if a product uses a Key Encryption Key (KEK) to encrypt a File Encryption Key (FEK), destroying the KEK using one of the methods in FCS_CKM_EXT.4 is sufficient, since the FEK would no longer be usable (of course, presumes the FEK is still encrypted and the KEK cannot be recovered or re-derived).

FCS_CKM_EXT.4.2 The TSF shall destroy all keys and key material when no longer needed.

Application Note: Keys, including intermediate keys and key material that are no longer needed are destroyed by using an approved method, FCS_CKM_EXT.4.1. Examples of keys are intermediate keys, submasks. There may be instances where keys or key material that are contained in persistent storage are no longer needed and require destruction. Base on their implementation, vendors will explain when certain keys are no longer needed. There are multiple situations in which key material is no longer necessary, for example, a wrapped key may need to be destroyed when a password is changed. However, there are instances when keys are allowed to remain in memory, for example, a device identification key. If a PIN was used for a smart card and managed by the TOE, ensuring that the PIN was properly destroyed must be addressed.

FCS_IV_EXT.1 Initialization Vector Generation

FCS_IV_EXT.1.1 The TSF shall [**selection**:

- *invoke platform-provided functionality to generate IVs*,
- *generate IVs with the following properties [**selection**:*
  - *CBC: IVs shall be non-repeating and unpredictable*,
  - *CCM: Nonce shall be non-repeating and unpredictable*,
  - *XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer,*
  - *GCM: IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^32 for a given secret key*

  ]

].

FCS_KYC_EXT.1 Key Chaining and Key Storage

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

- *a conditioned password as the [FEK]*,
- *[KEKs] originating from* ***[selection: one or more authorization factors(s), a file encryption enterprise management server]*** *to* ***[selection: the FEK(s), a file encryption enterprise management server]*** *using the following method(s): [**selection**:*
  - *utilization of the platform key storage*,
  - *utilization of platform key storage that performs key wrap with a TSF provided key,*
  - *implementation of key wrapping as specified in FCS_COP.1**(5)**,*
  - *implementation of key combining as specified in FCS_SMC_EXT.1*,
  - *implementation of key encryption as specified in FCS_COP.1**(7)**,*
  - *implementation of key transport as specified in FCS_COP.1**(6)**,*
  - *implementation of key derivation as specified in FCS_KDF_EXT.1*

  *] while maintaining an effective strength of [**selection**:*
  - *[**selection**: 128 bits, 256 bits] for symmetric keys ,*
  - *[**selection**: 128 bits, 192 bits, 256 bits] for asymmetric keys*

  ] commensurate with the strength of the FEK

] and [**selection**:

- *no supplemental key chains*,
- *other supplemental key chains that protect a key or keys in the primary key chain using the following method(s): [**selection**:*
  - *utilization of the platform key storage*,
  - *utilization of platform key storage that performs key wrap with a TSF provided key,*
  - *implementation of key wrapping as specified in FCS_COP.1**(5)**,*
  - *implementation of key combining as specified in FCS_SMC_EXT.1*,

- *implementation of key encryption as specified in FCS_COP.1**(7)**,*
- *implementation of key transport as specified in FCS_COP.1**(6)**,*
- *implementation of key derivation as specified in FCS_KDF_EXT.1*

]

].

Application Note: Key Chaining is the method of using multiple layers of encryption keys to ultimately secure the FEK. The number of intermediate keys will vary. This applies to all keys that contribute to the ultimate wrapping or derivation of the FEK. For the first selection, the ST author selects the method used for the keychain.

If the second option is chosen ("KEKs originating:") , then the ST author chooses all methods for production and protection of KEKs in the keychain from the options in the second selection. For this option, the ST author must also specify the strength of the keys in the keychain. It should be noted that "maintaining overall strength: commensurate with the overall strength of the FEK" is meant to cover the use case for this PP-Module of a powered-off device being recovered by an adversary, who subsequently attempts to recover the FEK through a compromise of the key chain.

The third selection in the requirement is used to select the types of keys used in the key chain (both symmetric and asymmetric keys are allowed). The bit sizes selected in the fourth and fifth selections are chosen by the ST author to be commensurate with the strength of the FEK in the following manner: for symmetric FEKs of 128 bits, the ST author can select any of the choices for both symmetric and asymmetric keys. For symmetric FEKs of 256 bits, the ST author selects 256 bits if the symmetric key option is chosen and 192 bits or 256 bits if the asymmetric key option is chosen.

If a supplemental keychain is used, then the ST author selects the second option in the sixth selection and then chooses the method by which these keys are protected. Keys in the supplemental key chain may be of any size, as they only provide additional protection to the primary key chain. Compromise (according the PP-Module use case) of the secondary key chain cannot circumvent the protection provided by the primary keychain.

If the selections where the TOE implements KEKs are chosen for the primary or supplemental key chains then FCS_CKM_EXT.3 shall be included.

The selections for an enterprise management server permit the key chain may originate or terminate from an enterprise management server.

The server may provide a key needed to start a chain or the server may receive a key that ends a chain.

The key management internal to the server is not evaluated here. This permits the enterprise management server to function in the middle of a larger key chain.

FCS_VAL_EXT.1 Validation

FCS_VAL_EXT.1.1 The TSF shall perform validation of the [*user*] by [**selection**:

- *receiving assertion of the subject's validity from [**assignment**: Operational Environment component responsible for authentication]*,
- *validating the [**selection**: submask, intermediate key] using the following methods: [**selection**:*
  - *key wrap as specified in FCS_COP.1**(5)**,*
  - *hash the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1**(2) (from [AppPP])** and compare it to a stored hash,*
  - *decrypt a known value using the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1**(1) (from [AppPP])** and compare it against a stored known value*

  ]

].

FCS_VAL_EXT.1.2 The TSF shall require validation of the [*user*] prior to [*decrypting any FEK*].

Application Note: Two iterations of this SFR are also defined in PP-Module for File Encryption Enterprise Management. If the TOE also claims this module, the ST author should iterate these SFRs by using "/FE" and "/EM" as unique identifiers for the iterations. This will allow the reader to easily determine which iteration applies to each TOE component.

5.2.2 User Data Protection (FDP)

FDP_PRT_EXT.1 Protection of Selected User Data

FDP_PRT_EXT.1.1 The TSF shall perform encryption and decryption of the user-selected file (or set of files) in accordance with FCS_COP.1(1) **(from [AppPP])**.

Application Note: This is the primary requirement for encrypting and decrypting the protected resources (files and sets of files).

FDP_PRT_EXT.1.2 The TSF shall [**selection**: *invoke platform-provided functionality*, *implement functionality*] to ensure that all sensitive data created by the TOE when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory when the data is no longer needed according to FCS_CKM_EXT.4.

Application Note: The intent is that the TSF controls the use and clearing of any data that it manipulates that is not needed by the user (e.g. a temporary file created in non-volatile memory during the encryption/decryption process would be destroyed as soon as the process is completed). This should not prevent expected usage (e.g. the TOE may create a decrypted copy of a file as requested by the user). The TSF is also not responsible for temporary files that non-TSF application creates (for example, a text editor may create a "checkpoint" file when editing a file that is protected by the TOE; the TOE does not have to try to keep track of or clean up these "checkpoint" files). An optional requirement on cleaning up the temporary files created by non-TSF application when operating on files protected by the TOE is FDP_PRT_EXT.3.1. While these data sets are not keys, they can follow the same deletion procedures described in FCS_CKM_EXT.4.

FDP_PRT_EXT.2 Destruction of Plaintext Data

FDP_PRT_EXT.2.1 The TSF shall [**selection**: *invoke platform-provided functionality*, *implement functionality*] to ensure that all original plaintext data created when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory according to FCS_CKM_EXT.4 upon completion of the decryption/encryption operation.

Application Note: This is the primary requirement for encrypting and decrypting the protected resources (file or set of files).

For FDP_PRT_EXT.2.1, the intent is that the TSF controls the use and clearing of any data that it manipulates. It needs to ensure that no plaintext data from encrypted resources remains after the TSF has finished operating on that resource. In the context of FDP_PRT_EXT.2.1, the TSF has completed the decryption operation after it has decrypted the file or set of files for use by an application, and completed the encryption operation after it has encrypted the file or set of files for storage in the file system.

While these data sets are not keys, they can follow the same deletion procedures described in FCS_CKM_EXT.4.

5.2.3 Identification and Authentication (FIA)

FIA_AUT_EXT.1 Subject Authorization

FIA_AUT_EXT.1.1 The TSF shall [**selection**: *implement platform-provided functionality to provide user authorization*, *provide user authorization*] based on [**selection**:

- *a password authorization factor conditioned as defined in FCS_CKM_EXT.6*,
- *an external smart card factor that is at least the same bit-length as the FEK(s), and is protecting a submask that is [**selection**: generated by the TOE (using the RBG as specified in FCS_RBG_EXT.1 **(from [AppPP])**), generated by the platform] protected using RSA with key size [**selection**: 3072 bits, 4096 bits] with user presence proved by presentation of the smart card and [**selection**: no PIN, an OE defined PIN, a configurable PIN]* ,
- *an external USB token factor that is at least the same security strength as the FEK(s), and is providing a submask generated by the [**selection**: TOE (using the RBG as specified in FCS_RBG_EXT.1 **(from [AppPP])**), platform]*

].

Application Note: If the ST author selects "provide user authorization", the selection-based requirement FCS_VAL_EXT.2 must also be claimed.

This requirement specifies what authorization factors the TOE accepts from the user. A password entered by the user is one authorization factor that the TOE must be able to condition, as specified in FCS_CKM_EXT.6. Another option is a smart card authorization factor, with the differentiating feature being how the value is generated - either by the TOE's RBG or by the platform. An external USB token may also be used, with the submask value generated either by the TOE's RBG or by the platform.

The TOE may accept any number of authorization factors, and these are categorized as "submasks". The ST author selects the authorization factors they support, and there may be multiple methods for a selection.

Use of multiple authorization factors is preferable; if more than one authorization factor is used, the submasks produced must be combined using FCS_SMC_EXT.1.

5.2.4 Security Management (FMT)

FMT_SMF.1(2) Specification of File Encryption Management Functions

FMT_SMF.1.1(2) The TSF shall be capable of performing the following management functions: **[selection:**

- ***configure cryptographic functionality*,**
- ***change authentication factors*,**
- ***perform a cryptographic erase of the data by the destruction of FEKs or KEKs protecting the FEKs as described in FCS_CKM_EXT.4.1*,**
- ***configure the number of failed validation attempts required to trigger corrective behavior* ,**
- ***configure the corrective behavior to issue in the event of an excessive number of failed validation attempts*,**

- ***[assignment: other management functions provided by the TSF]***

].

Application Note: The intent of this requirement is to express the management capabilities that may be included in the TOE. Several common options are given:

- If the TOE provides configurability of the cryptographic functions (for example, key size of the FEK)-even if the configuration is the form of parameters that may be passed to cryptographic functionality implement on the TOE platform--then "configure cryptographic functionality" will be included, and the specifics of the functionality offered can either be written in this requirement as bullet points, or included in the TSS.
- If the TOE uses stored FEKS or KEKs(the FEK is not directly derived from a password) , then "perform a cryptographic erase of the data by the destruction of FEKs or KEKs protecting the FEKs as described in FCS_CKM_EXT.4.1" will be included. This function should be able to be triggered by a user or admin and may be triggered by uninstallation.
- If "other management functions" are assigned, a validation authority must be consulted to ensure the evaluation activities and other functionality requirements that may be needed are appropriately specified so that the ST can claim conformance to this PP-Module.

This list is in addition to the list of functions as specified in, FMT_SMF.1 in the AppPP.

5.2.5 Protection of the TSF (FPT)

FPT_KYP_EXT.1 Protection of Keys and Key Material

FPT_KYP_EXT.1.1 The TSF shall [**selection**:

- *not store keys in non-volatile memory*,
- *store keys in non-volatile memory only when [**selection**:*
  - *wrapped, as specified in FCS_COP.1(5)*,
  - *encrypted, as specified in FCS_COP.1(7)*,
  - *the plaintext key is stored in the underlying platform's keystore as specified by FCS_STO_EXT.1.1* ***(from [AppPP])***,
  - *the plaintext key is not part of the key chain as specified in FCS_KYC_EXT.1,*
  - *the plaintext key will no longer provide access to the encrypted data after initial provisioning,*
  - *the plaintext key is a key split that is combined as specified in FCS_SMC_EXT.1 and another contribution to the split is [**selection**:*
    - *wrapped as specified in FCS_COP.1(5)*,
    - *encrypted as specified in FCS_COP.1(7)*,
    - *derived as specified in FCS_KDF_EXT.1.1 and not stored in non-volatile memory,*
    - *supplied by the enterprise management server*

    ] ,
  - *the plaintext key is stored on an external storage device for use as an authorization factor.,*
  - *the plaintext key is used to encrypt a key as specified in FCS_COP.1(7) or wrap a key as specified in FCS_COP.1(5) that is already encrypted as specified in FCS_COP.1(7) or wrapped as specified in FCS_COP.1(5)*

  ]

].

Application Note: The plaintext key storage in non-volatile memory is allowed for several reasons. If the keys exist within protected memory that is not user accessible on the TOE or OE, the only methods that allow it to play a security relevant role for protecting the FEK is if it is a key split or providing additional layers of wrapping or encryption on keys that have already been protected.

6 Consistency Rationale

6.1 Application Software Protection Profile

6.1.1 Consistency of TOE Type

6.1.2 Consistency of Security Problem Definition

The threats defined by this PP-Module (see section 3.1) supplement those defined in the App PP as follows:

PP-Module Threat	Consistency Rationale
T.UNAUTHORIZED_DATA_ACCESS	This threat is a variation on T.PHYSICAL_ACCESS defined in the Base-PP. In this case, the "sensitive data at rest" is the data that the TOE is intended to protect.
T.MANAGEMENT_ACCESS	This threat is a variation on T.LOCAL_ATTACK defined in the Base-PP. The Base-PP does not define access-controlled management functions so this PP-Module goes beyond it by specifying misuse of the management interface, or inability to fully use the management interface, as threats to the TSF.
T.KEYING_MATERIAL_COMPROMISE	This threat is a specific example of T.PHYSICAL_ACCESS defined in the Base-PP. Specifically, this PP-Module defines a method of maliciously gaining access to sensitive data at rest that is particular to the technology type of this PP-Module.
T.UNSAFE_AUTHFACTOR_VERIFICATION	This threat is a specific example of T.PHYSICAL_ACCESS defined in the Base-PP. Specifically, this PP-Module defines a method of maliciously gaining access to sensitive data at rest that is particular to the technology type of this PP-Module.
T.KEYSPACE_EXHAUST	This threat is a specific example of T.PHYSICAL_ACCESS defined in the Base-PP. Specifically, this PP-Module defines a method of maliciously gaining access to sensitive data at rest that is particular to the technology type of this PP-Module.
T.PLAINTEXT_COMPROMISE	This threat is a specific example of T.PHYSICAL_ACCESS defined in the Base-PP. Specifically, this PP-Module defines a method of maliciously gaining access to sensitive data at rest that is particular to the technology type of this PP-Module.

6.1.3 Consistency of Objectives

The objectives for the TOEs are consistent with the App PP based on the following rationale:

PP-Module TOE Objective	Consistency Rationale
O.KEY_MATERIAL_PROTECTION	This objective is consistent with the Base-PP because the Base-PP includes the O.PROTECTED_STORAGE objective. The protection and timely destruction of key materials is consistent with the intent of that objective.
O.FEK_SECURITY	This objective is consistent with the Base-PP because it is a method of supporting the Base-PP's O.PROTECTED_STORAGE objective that is specific to this technology type.
O.WIPE_MEMORY	This objective is consistent with the Base-PP because it is a method of supporting the Base-PP's O.PROTECTED_STORAGE objective that is specific to this technology type.
O.PROTECT_DATA	This objective is consistent with the Base-PP because it is a method of supporting the Base-PP's O.PROTECTED_STORAGE objective that is specific to this technology type.
O.SAFE_AUTHFACTOR_VERIFICATION	This objective is consistent with the Base-PP because it is a method of supporting the Base-PP's O.PROTECTED_STORAGE objective that is specific to this technology type.
O.MANAGE	The Base-PP does not define functionality for protected administrative access to the TSF so this objective relates solely to material beyond the scope of the Base-PP.

The objectives for the TOE's Operational Environment are consistent with the App PP based on the following rationale:

PP-Module Operational Environment Objective	Consistency Rationale
OE.AUTHORIZATION_FACTOR_STRENGTH	This objective is consistent with the Base-PP because this functionality is beyond the scope of what the Base-PP defines. Therefore, the use and strength of external authorization factors does not affect the ability of any Base-PP SFRs or objectives to be satisfied.

OE.POWER_SAVE	This objective is consistent with the Base-PP because it is an extension of the Base-PP's OE.PLATFORM objective that is specific to this technology type.
OE.STRONG_ENVIRONMENT_CRYPTO	This objective is consistent with the Base-PP because the Base-PP allows for the TOE to use platform-provided cryptography.

6.1.4 Consistency of Requirements

This PP-Module identifies several SFRs from the App PP that are needed to support File Encryption functionality. This is considered to be consistent because the functionality provided by the App is being used for its intended purpose. The PP-Module also identifies a number of modified SFRs from the App PP as well as new SFRs that are used entirely to provide functionality for File Encryption. The rationale for why this does not conflict with the claims defined by the App PP are as follows:

PP-Module Requirement	Consistency Rationale
Modified SFRs	
This PP-Module does not modify any requirements when the App PP is the base.	
Mandatory SFRs	
FCS_CKM_EXT.2	This SFR describes behavior that is not in scope of the Base-PP. It is consistent with the Base-PP because it may use the same random bit generation function defined in the Base-PP.
FCS_CKM_EXT.4	This SFR extends the cryptographic functionality defined in the Base-PP by specifying a method for key destruction. It is consistent with the Base-PP because keys generated by the Base-PP portion of the TOE may also be destroyed in the manner specified by this SFR.
FCS_IV_EXT.1	This SFR defines how IVs for AES keys must be generated. This is consistent with the Base-PP because it supplements the key generation methods specified by the Base-PP SFR FCS_CKM.1(2).
FCS_KYC_EXT.1	The Base-PP defines how stored keys are protected. This SFR extends that functionality by defining the logical hierarchy of how keys are logically protected by other keys or other secret data.
FCS_VAL_EXT.1	This SFR goes beyond the functionality defined by the Base-PP by defining a method by which the TSF can validate the correctness of data input to it.
FDP_PRT_EXT.1	This SFR is consistent with the Base-PP because it is a specific application of the FCS_COP.1(1) function defined in the Base-PP.
FDP_PRT_EXT.2	This SFR relates to the destruction of key data, which is beyond the scope defined by the Base-PP and does not affect the ability of the Base-PP SFRs to be enforced.
FIA_AUT_EXT.1	This SFR defines how user requests to access protected data are authorized. It uses FCS_RBG_EXT.1 from the Base-PP in a manner consistent with its definition, but otherwise does not relate to functionality defined by the Base-PP.
FMT_SMF.1(2)	This SFR defines management functions for the TOE for functionality specific to this PP-Module. These functions are defined in addition to what the Base-PP defines for its own operation.
FPT_KYP_EXT.1	The Base-PP defines an SFR for secure storage of sensitive data. This SFR expands on that definition by describing the supported logical methods for storage of key data.
Optional SFRs	
FCS_CKM_EXT.5	This SFR supports the PP-Module's data authentication function, which does not relate to any functionality defined in the Base-PP.
FCS_COP_EXT.1	This SFR defines usage of AES functionality not defined by the Base-PP. However, this functionality is only used in certain situations that are specific to this PP-Module and do not affect the ability for any Base-PP SFRs to be enforced.
FDP_AUT_EXT.1	This SFR relates to data authentication, which does not relate to any functionality defined in the Base-PP.
FDP_AUT_EXT.2	This SFR relies on cryptographic functionality defined by the Base-PP. However, the function itself does not relate to any behavior defined in the Base-PP.
FDP_AUT_EXT.3	This SFR relies on cryptographic functionality defined by the Base-PP. However, the function itself does not relate to any behavior defined in the Base-PP.
FDP_PM_EXT.1	This SFR describes the behavior of the TSF when its host platform is in a locked or unpowered state, which does not relate to any functionality defined in the Base-PP.
FDP_PRT_EXT.3	This SFR relates to the PP-Module's file encryption capability. This goes beyond the sensitive data protection defined in the Base-PP but does not prevent the Base-PP functions from being enforced.
FIA_FCT_EXT.1	This SFR relates to the use of authorization factors, which does not relate to any behavior described in the Base-PP.
FIA_FCT_EXT.2	This SFR relates to key sharing, which does not relate to any behavior described in the Base-PP.

Selection-based SFRs	
FCS_CKM_EXT.3	This SFR relates to how KEKs are made available to the TSF, which are used for functionality that does not relate to the Base-PP.
FCS_CKM_EXT.6	This SFR defines a key derivation method based on passphrase conditioning. It uses the FCS_RBG_EXT.1 SFR from the Base-PP in its intended manner but otherwise does not relate to the Base-PP's functionality.
FCS_COP.1(5)	This SFR defines usage of AES functionality not defined by the Base-PP. However, this functionality is only used in certain situations that are specific to this PP-Module and do not affect the ability of any Base-PP SFRs to be enforced.
FCS_COP.1(6)	This SFR defines key transport functionality that is outside the scope of the original cryptographic operations defined in the Base-PP.
FCS_COP.1(7)	This SFR defines key encryption functionality that is outside the scope of the original cryptographic operations defined in the Base-PP.
FCS_KDF_EXT.1	This SFR defines key transport functionality. It uses random bit generation and keyed-hash message authentication functionality from the Base-PP as they are intended but is otherwise outside the scope of the original cryptographic operations defined in the Base-PP.
FCS_SMC_EXT.1	This SFR relates to submask combining as a method of generating intermediate keys. Key hierarchy functionality is outside the scope of the Base-PP.
FCS_VAL_EXT.2	This SFR goes beyond the functionality defined by the Base-PP by defining a method by which the TSF can take security-relevant action if some data input to it is invalid.
Objective SFRs	
This PP-Module does not define any objective requirements.	

Appendix A - Optional SFRs

FCS_CKM_EXT.5 File Authentication Key (FAK) Support

FCS_CKM_EXT.5.1 The TSF shall use a FAK to authenticate sensitive data when a cryptographic, keyed hashing function is used for data authentication and shall be supported in the following manner: [**selection**:

- *A FAK conditioned from a password/passphrase shall never be stored in non-volatile memory*,
- *a FAK will be stored in non-volatile memory encrypted with a KEK as specified in FCS_COP.1(5) using authorization factors as specified in FIA_AUT_EXT.1*

].

FCS_CKM_EXT.5.2 The TSF shall create a unique FAK for each file (or set of files) using the mechanism on the client as specified in FCS_RBG_EXT.1 **(from [AppPP])**.

FCS_CKM_EXT.5.3 The FAKs must be generated by the TOE as specified in FDP_AUT_EXT.2.9.

FCS_CKM_EXT.5.4 The TSF will not write FAKs to non-volatile memory.

FCS_CKM_EXT.5.5 The FAK shall be protected in a manner conformant to FCS_COP_EXT.1.

Application Note: The intent of this requirement is to describe the different methods that a FAK can be created and formed.

FCS_CKM_EXT.5.1 details how a FAK is stored.

FCS_CKM_EXT.5.2 requires that each resource to be encrypted has a unique FAK, and that this FAK is generated by the TSF. If the encrypted resource is a set of files encrypted under one FAK, additional requirements on the initialization vectors and cipher modes must be adhered to in Section 4.2.

FCS_COP_EXT.1 FAK Encryption/Decryption Support

FCS_COP_EXT.1.1 The FAK shall be protected in the same manner as the FEK, in accordance with FCS_COP.1(5).

Application Note: The intent of this requirement is to clarify that, if a FAK is to be used, it should be treated as sensitive as the FEK, and thus, follow the same encryption and decryption practices.

FDP_AUT_EXT.1 Authentication of Selected User Data

FDP_AUT_EXT.1.1 The TSF shall perform authentication of the user-selected file (or set of files) and provide notification to the user if modification had been detected.

FDP_AUT_EXT.1.2 The TSF shall implement a data authentication method based on [**selection**: *cryptographic keyed hashing service and verification in accordance with FDP_AUT_EXT.2*, *asymmetric signing and verification in accordance with FDP_AUT_EXT.3*].

Application Note: This is the primary requirement for authentication of the protected resources (files and sets of files). It is highly encouraged for vendors to utilize a keyed hashing service or asymmetric signing mechanism to ensure data authentication, as these are the only two implementations noted in this PP-Module that prevent decryption if authentication is unsuccessful. Using modes such as XTS or CBC will require additional data authentication measures to be added, such as a keyed hash function or asymmetric signing, because these modes do not come inherently packaged with data authentication or a way to signal to the user that data has been modified.
Specific tests are performed in FDP_AUT_EXT.2 or FDP_AUT_EXT.3 depending on the selection made in FDP_AUT_EXT.1.2.

FDP_AUT_EXT.2 Data Authentication Using cryptographic Keyed-Hash Functions

FDP_AUT_EXT.2.1 The TSF shall use a cryptographic, keyed hash function in accordance with FCS_COP.1(4) **(from [AppPP])**.

FDP_AUT_EXT.2.2 The TSF shall use a File Authentication Key (FAK) in accordance with FCS_COP_EXT.1 and FCS_CKM_EXT.5 as the secret key to the keyed hash function.

FDP_AUT_EXT.2.3 The TSF shall use the entirety of the ciphertext file as the message input to the keyed hash function.

FDP_AUT_EXT.2.4 The TSF shall concatenate the output of the keyed hash function, the Message Authentication Code (MAC).

FDP_AUT_EXT.2.5 The TSF shall authenticate the encrypted file prior to decryption.

FDP_AUT_EXT.2.6 The TSF shall authenticate the data by comparing the keyed hash output of the ciphertext against the stored MAC.

FDP_AUT_EXT.2.7 The TSF shall notify the user of an unsuccessful authentication and prevent decryption of the ciphertext.

FDP_AUT_EXT.2.8 During verification, the TSF shall verify the MAC is at the end of the ciphertext file.

FDP_AUT_EXT.2.9 The FAK will be generated using a RBG that meets FCS_RBG_EXT.1 **(from [AppPP])**.

Application Note: The intent of this requirement is to specify the correct way of using a keyed hash function to authenticate the data, and enable authentication of data. FAKs are considered cryptographic keys and are subject to destruction per FCS_CKM_EXT.4.1.

FDP_AUT_EXT.3 Data Authentication Using Asymmetric Signing and Verification

FDP_AUT_EXT.3.1 The TSF shall use a secure hash function in accordance with FCS_COP.1(2) **(from [AppPP])** with the entire ciphertext file as input to create a hash.

FDP_AUT_EXT.3.2 The TSF shall use a cryptographic signing function in accordance with FCS_COP.1(3) **(from [AppPP])** and must use the hash generated in accordance with FDP_AUT_EXT.3.1 as input to the signing process. Additionally, use of ephemeral key for signing purposes is prohibited.

FDP_AUT_EXT.3.3 The TSF shall use a public and private key pair generated in accordance with FCS_CKM.1(1) **(from [AppPP])** and must use this key pair as part of the cryptographic signing process in accordance with FDP_AUT_EXT.3.2.

FDP_AUT_EXT.3.4 The TSF shall authenticate the ciphertext data prior to decryption.

FDP_AUT_EXT.3.5 The TSF shall notify the user of an unsuccessful authentication and prevent decryption of the ciphertext if such an event were to occur.

FDP_AUT_EXT.3.6 The TSF shall append the signature to the end of the ciphertext file.

FDP_AUT_EXT.3.7 During verification, the TSF shall verify the signature is at the end of the ciphertext file.

Application Note: The intent of this requirement is to specify the secure way of using a cryptographic signing and hashing function as part of the data authentication mechanism.

FDP_PM_EXT.1 Protection of Data in Power Managed States

FDP_PM_EXT.1.1 The TSF shall protect all data selected for encryption during the transition to the [**assignment**: *powered-down state(s) or locked system states for which this capability is provided*] state as per FDP_PRT_EXT.1.1.

FDP_PM_EXT.1.2 On the return to a powered-on state from the state(s) indicated in FDP_PM_EXT.1.1, the TSF shall authorize the user in the manner specified in FIA_AUT_EXT.1.1 once before any protected data are decrypted.

FDP_PM_EXT.1.3 The TSF shall destroy all key material and authentication factors stored in plaintext when transitioning to a protected state as defined by FDP_PM_EXT.1.1.

Application Note: For the first assignment, the ST author fills in the state(s) using the same name used in the Operational Guidance for the state that is appropriately protected by the TOE.
It should be noted that it is not sufficient to use Operational Environment-based credentials to unlock the TOE from the indicated state; the intent is that returning from the indicated state is equivalent (from an authorization point of view) to returning from a completely powered-off state and re-opening the resources that are protected.

FDP_PRT_EXT.3 Protection of Third-Party Data

FDP_PRT_EXT.3.1 The TSF shall ensure that all temporary files created by [**selection**: *all applications*, *[**assignment**: subset of applications that can integrate with the FE]*]when decrypting/encrypting the user-selected file (or set of files) are removed or encrypted upon completion of the decryption/encryption operation.

Application Note: This requirement is to cover the detection and encryption of temporary files created by third party applications. If the FE provides a capability to allow specific applications to leverage it, the applications or method they would use to opt in may be included in the assignment.

FIA_FCT_EXT.1 Multi-User Authorization

FIA_FCT_EXT.1.1 The TSF shall support the use of authorization factors from multiple users that result in unique KEKs.

FIA_FCT_EXT.1.2 The TSF shall support the ability of each user to have files protected by a key chain tied only to that user's credentials.

Application Note: FIA_FCT_EXT.1.1 requires the TSF to support multiple authorization factors to produce multiple KEKs, the intent is that the TSF supports a system where multiple users have access to files on the underlying platform, and that each user has an authorization factor so that they can protect their own files from other users. This should be accomplished via the methods detailed in FIA_FCT_EXT.1.2.

FIA_FCT_EXT.2 Authorized Key Sharing

FIA_FCT_EXT.2.1 The TSF shall support [**selection**: *Authorized User Key sharing via key transport as specified in FCS_COP.1(6)*, *Distribution of a shared key from an enterprise management Server*].

Application Note: While FIA_FCT_EXT.1 requires that each user has an authorization factor so that they can protect their own files from other users. FIA_FCT_EXT.2 created a mechanism to safely share files between users.

Appendix B - Selection-based SFRs

FCS_CKM_EXT.3 Key Encrypting Key (KEK) Support

This is a selection-based component. Its inclusion depends upon selection from FCS_KYC_EXT.1.1.

FCS_CKM_EXT.3.1 The TSF shall [**selection**:

- *accept KEK from an enterprise management server,*
- *generate KEK cryptographic keys*
  *[**selection**:*
  - *using a Random Bit Generator as specified in FCS_RBG_EXT.1* ***(from [AppPP])*** *and with entropy corresponding to the security strength of AES key sizes of [**selection**: 128 bit, 256 bit],*
  - *derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6*

  ]

].

Application Note: This requirement must be included in STs in which KEKs originating from is chosen in FCS_KYC_EXT.1.1.

FCS_CKM_EXT.6 Cryptographic Password/Passphrase Conditioning

This is a selection-based component. Its inclusion depends upon selection from FIA_AUT_EXT.1.1, FCS_CKM_EXT.2.1, FCS_CKM_EXT.3.1.

FCS_CKM_EXT.6.1 The TSF shall support a password/passphrase of up to [**selection**: *[**assignment**: maximum value supported by the platform]*, *[**assignment**: maximum password size, positive integer of 64 or more]*] characters used to generate a password authorization factor.

FCS_CKM_EXT.6.2 The TSF shall allow passwords to be composed of any combination of upper case characters, lower case characters, numbers, and the following special characters: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")", and [**selection**: *[**assignment**: other supported special characters]*, *no other characters*].

FCS_CKM_EXT.6.3 The TSF shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC-[**selection**: *SHA-256*, *SHA-384*, *SHA-512*], with [**selection**: *[**assignment**: positive integer of 4096 or more] iterations*, *value supported by the platform, greater than 1000*], and output cryptographic key sizes [**selection**: *128*, *256*] that meet the following: [*NIST SP 800-132*].

FCS_CKM_EXT.6.4 The TSF shall not accept passwords less than [**selection**: *a value settable by the administrator*, *[**assignment**: minimum password length accepted by the TOE, must be >= 1]*] and greater than the maximum password length defined in FCS_CKM_EXT.6.1.

FCS_CKM_EXT.6.5 The TSF shall generate all salts using an RBG that meets FCS_RBG_EXT.1 **(from [AppPP])** and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.6.3.

Application Note: The password/passphrase is represented on the host machine as a sequence of characters whose encoding depends on the TOE and the underlying OS. This sequence must be conditioned into a string of bits that is to be used as a KEK that is the same size as the FEK.

For FCS_CKM_EXT.6.1, the ST author assigns the maximum size of the password/passphrase it supports; it must support at least 64 characters or a length defined by the platform. The selection "maximum value supported by the platform" may only be selected if "implement platform-provided functionality to provide user authorization" was selected in FIA_AUT_EXT.1.

For FCS_CKM_EXT.6.2, the ST author assigns any other supported characters; if there are no other supported characters, they should select "no other characters".

For FCS_CKM_EXT.6.3, the ST author selects the parameters based on the PBKDF used by the TSF. The key cryptographic key sizes in FCS_CKM_EXT.6.3 are made to correspond to the KEK key sizes selected in FCS_CKM_EXT.3.

The password/passphrase must be conditioned into a string of bits that forms the submask to be used as input into the KEK. Conditioning is performed using one of the identified hash functions in accordance with the process described in NIST SP 800-132. SP 800-132 requires the use of a pseudo-random function (PRF) consisting of HMAC with an approved hash function.

Appendix A of SP 800-132 recommends setting the iteration count in order to increase the computation needed to derive a key from a password and, therefore, increase the workload of performing a password recovery attack. However, for this PP-Module, a minimum iteration count of 4096 is required in order to ensure that twelve bits of security is added to the password/passphrase value. A significantly higher value is recommended to ensure optimal security. If the platform is leveraged for authentication the value may be a minimum of 1000, this selection may only be selected if "implement platform-provided functionality to provide user authorization" was selected in FIA_AUT_EXT.1.

For FCS_CKM_EXT.6.4 If the minimum password length is settable, then ST author chooses "a value settable by the administrator for this component for FMT_SMF.1.2. If the minimum length is not settable, the ST author fills in the assignment with the minimum length the password must be (zero-length passwords are not allowed for compliant TOEs).

This requirement is selection dependent on FIA_AUT_EXT.1.1.

FCS_COP.1(5) Cryptographic operation (Key Wrapping)

This is a selection-based component. Its inclusion depends upon selection from FCS_KYC_EXT.1.1.

FCS_COP.1.1(5) The TSF shall **[selection:** ***use platform-provided functionality to perform Key Wrapping***, ***implement functionality to perform Key Wrapping*****]** in accordance with a specified cryptographic algorithm [*AES*] **in the following modes [selection:**

- ***Key Wrap***,
- ***Key Wrap with Padding***,
- ***GCM mode***,
- ***CCM mode***

] and cryptographic key sizes **[selection:** ***128 bits (AES)***, ***256 bits (AES)*****]** that meet the following: **[selection:**

- ***"NIST SP 800-38C"***,
- ***"NIST SP 800-38D"***,
- ***"NIST SP 800-38F"***

] and no other standards.

Application Note: This requirement is used in the body of the ST if the ST author chooses to use key wrapping in the key chaining approach that is specified in FCS_KYC_EXT.1.

FCS_COP.1(6) Cryptographic operation (Key Transport)

This is a selection-based component. Its inclusion depends upon selection from FCS_KYC_EXT.1.1.

FCS_COP.1.1(6) The TSF shall perform [*key transport*] in accordance with a specified cryptographic algorithm [*RSA*] **in the following modes [selection:** ***KTS-OAEP***, ***KTS-KEM-KWS*****]** and cryptographic key sizes [**selection**: *3072*, *4096*] bits that meet the following: [*NIST SP 800-56B, Revision 1*].

Application Note: This requirement is used in the body of the ST if the ST author chooses to use key transport in the key chaining approach that is specified in FCS_KYC_EXT.1.

FCS_COP.1(7) Cryptographic operation (Key Encryption)

This is a selection-based component. Its inclusion depends upon selection from FCS_KYC_EXT.1.1.

FCS_COP.1.1(7) The TSF shall **[selection:** ***use platform-provided functionality to perform Key Encryption***, ***perform key encryption and decryption*****]** in accordance with a specified cryptographic algorithm [*AES used in CBC mode*] and cryptographic key sizes **[selection:**

- ***128***,
- ***256***

] bits that meet the following: [*AES as specified in SP 800-38A*].

Application Note: This requirement is used in the body of the ST if the ST author chooses to use AES encryption/decryption for protecting the keys as part of the key chaining approach that is specified in FCS_KYC_EXT.1.

FCS_KDF_EXT.1 Cryptographic Key Derivation Function

This is a selection-based component. Its inclusion depends upon selection from FCS_KYC_EXT.1.1.

FCS_KDF_EXT.1.1 The TSF shall [*accept [**selection**: a submask generated by an RBG as specified in FCS_RBG_EXT.1* ***(from [AppPP])***, *a conditioned password, an imported submask] to derive an intermediate key, as defined in [**selection**:*

- *NIST SP 800-108 [**selection**: KDF in Counter Mode, KDF in Feedback Mode, KDF in Double-Pipeline Iteration Mode] ,*
- *NIST SP 800-132*

*] using the keyed-hash functions specified in FCS_COP.1***(4) (from [AppPP])***, such that the output is at least of equivalent security strength (in number of bits) to the [FEK]*].

Application Note: This requirement is used in the body of the ST if the ST author chooses to use key derivation in the key chaining approach that is specified in FCS_KYC_EXT.1. This requirement establishes acceptable methods for generating a new random key or an existing submask to create a new key along the key chain.

FCS_SMC_EXT.1 Submask Combining

This is a selection-based component. Its inclusion depends upon selection from FCS_KYC_EXT.1.1.

FCS_SMC_EXT.1.1 The TSF shall [*combine submasks using the following method [**selection**: exclusive OR (XOR), SHA-256, SHA-384, SHA-512, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512] to generate an intermediate key*].

Application Note: This requirement specifies the way that a product may combine the various submasks by using either an XOR or an approved SHA-hash. This requirement is selection dependent on FCS_KYC_EXT.1.1.

FCS_VAL_EXT.2 Validation Remediation

This is a selection-based component. Its inclusion depends upon selection from FIA_AUT_EXT.1.1.

FCS_VAL_EXT.2.1 The TSF shall [**selection**:

- *[perform a key destruction of the FEK(s)] upon a configurable number of consecutive failed validation attempts*,
- *institute a delay such that only [**assignment**: ST author specified number of attempts] can be made within a 24 hour period*,
- *block validation after [**assignment**: ST author specified number of attempts] of consecutive failed validation attempts*,
- *require power cycle/reset the TOE after [**assignment**: ST author specified number of attempts] of consecutive failed validation attempts*

].

Application Note: This SFR must be included if "provide user authorization" is selected in FIA_AUT_EXT.1.1.

Two iterations of this SFR are also defined in PP-Module for File Encryption Enterprise Management. If the TOE also claims this module, the ST author should iterate these SFRs by using "/FE" and "/EM" as unique identifiers for the iterations. This will allow the reader to easily determine which iteration applies to each TOE component.

This requirement is used in the body of the ST if the ST author chooses "provide user authorization" in FIA_AUT_EXT.1.1.

Appendix C - Objective SFRs

This section is reserved for requirements that are not currently prescribed by this PP-Module but are expected to be included in future versions of the PP-Module. Vendors planning on having evaluations performed against future products are encouraged to plan for these objective requirements to be met.

This PP-Module does not define any objective SFRs.

Appendix D - Extended Component Definitions

This appendix contains the definitions for the extended requirements that are used in the PP-Module including those used in Appendices A through C.

D.1 Background and Scope

This Appendix provides a definition for all of the extended components introduced in this PP-Module. These components are identified in the following table:

Functional Class	Functional Components
Cryptographic Support (FCS)	FCS_CKM_EXT Cryptographic Key Management FCS_IV_EXT Initialization Vector Generation FCS_KYC_EXT Key Chaining and Key Storage FCS_VAL_EXT Validation
User Data Protection (FDP)	FDP_PRT_EXT Protection of Selected User Data
Identification and Authentication (FIA)	FIA_AUT_EXT Authorization
Protection of the TSF (FPT)	FPT_KYP_EXT Protection of Key and Key Material
Cryptographic Support (FCS)	FCS_COP_EXT Cryptographic Operation
User Data Protection (FDP)	FDP_AUT_EXT User Data Authentication FDP_PM_EXT Protection of Data in Power Managed States
Identification and Authentication (FIA)	FIA_FCT_EXT Authorization Factors
Cryptographic Support (FCS)	FCS_KDF_EXT Cryptographic Key Derivation Function FCS_SMC_EXT Submask Combining FCS_VAL_EXT Validation Remediation

D.2 Extended Component Definitions

FCS_CKM_EXT Cryptographic Key Management

Components in this family define requirements for key management activities that are beyond the scope of what is defined in the FCS_CKM family in CC Part 2.

Component Leveling

FCS_CKM_EXT.2, File Encryption Key (FEK) Generation, describes the method by which the TSF acquires or generates file encryption keys.

Management: FCS_CKM_EXT.2

There are no specific management functions identified.

Audit: FCS_CKM_EXT.2

There are no auditable events foreseen.

FCS_CKM_EXT.2 File Encryption Key (FEK) Generation

Hierarchical to: No other components.

Dependencies to: FCS_RBG_EXT.1 Random Bit Generation Services

FCS_CKM_EXT.2.1

The TSF shall [**selection**:

- *accept FEK from an enterprise management server*,
- *generate FEK cryptographic keys*
  *[**selection**:*
  - *using a Random Bit Generator as specified in FCS_RBG_EXT.1* ***(from [AppPP])*** *and with entropy corresponding to the security strength of AES key sizes of [**selection**: 128 bit, 256 bit],*
  - *derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6*

  ]

].

FCS_CKM_EXT.2.2

The TSF shall use a unique FEK for each file (or set of files) using the mechanism on the client as specified in FCS_CKM_EXT.2.1.

Component Leveling

FCS_CKM_EXT.4, Cryptographic Key Destruction, describes supported methods for key destruction.

Management: FCS_CKM_EXT.4

The following actions could be considered for the management functions in FMT:

- Manually perform cryptographic erasure.

Audit: FCS_CKM_EXT.4

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Basic: Manual erasure of cryptographic data.

FCS_CKM_EXT.4 Cryptographic Key Destruction

Hierarchical to: No other components.

Dependencies to: No dependencies.

FCS_CKM_EXT.4.1

The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [**selection**:

- *For volatile memory, the destruction shall be executed by a [**selection**:*
    - *single overwrite consisting of [**selection**: a pseudo-random pattern using the TSF's RBG, zeroes, ones, new value of a key, [**assignment**: any value that does not contain any CSP]] ,*
    - *removal of power to the memory,*
    - *destruction of reference to the key directly followed by a request for garbage collection*

  ] ,
- *For non-volatile memory, the destruction shall be executed by [**selection**:*
    - *destruction of all KEKs protecting the target key, where none of the KEKs protecting the target key are derived ,*
    - *the invocation of an interface provided by the underlying platform that [**selection**:*
        - *logically addresses the storage location of the key and performs a [**selection**: single, [**assignment**: ST author defined multi-pass]] overwrite consisting of [**selection**: a pseudo-random pattern using the TSF's RBG, zeroes, ones, new value of a key, [**assignment**: any value that does not contain any CSP]] ,*
        - *instructs the underlying platform to destroy the abstraction that represents the key*

      ]

  ]

].

FCS_CKM_EXT.4.2

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

Component Leveling

FCS_CKM_EXT.5, File Authentication Key (FAK) Support, describes the secure storage of file encryption keys.

Management: FCS_CKM_EXT.5

There are no specific management functions identified.

Audit: FCS_CKM_EXT.5

There are no auditable events foreseen.

FCS_CKM_EXT.5 File Authentication Key (FAK) Support

Hierarchical to: No other components.

Dependencies to: FCS_COP_EXT.1 FAK Encryption/Decryption Support
FCS_RBG_EXT.1 Random Bit Generation Services
FDP_AUT_EXT.2 Data Authentication Using Cryptographic Keyed-Hash Functions

FCS_CKM_EXT.5.1

The TSF shall use a FAK to authenticate sensitive data when a cryptographic, keyed hashing function is used for data authentication and shall be supported in the following manner: [**selection**:

- *A FAK conditioned from a password/passphrase shall never be stored in non-volatile memory*,
- *a FAK will be stored in non-volatile memory encrypted with a KEK as specified in FCS_COP.1(5) using authorization factors as specified in FIA_AUT_EXT.1*

].

FCS_CKM_EXT.5.2

The TSF shall create a unique FAK for each file (or set of files) using the mechanism on the client as specified in FCS_RBG_EXT.1 **(from [AppPP])**.

FCS_CKM_EXT.5.3

The FAKs must be generated by the TOE as specified in FDP_AUT_EXT.2.9.

FCS_CKM_EXT.5.4

The TSF will not write FAKs to non-volatile memory.

FCS_CKM_EXT.5.5

The FAK shall be protected in a manner conformant to FCS_COP_EXT.1.

Component Leveling

FCS_CKM_EXT.3, Key Encrypting Key (KEK) Support, describes the method by which the TSF acquires or generates key encryption keys.

Management: FCS_CKM_EXT.3

There are no specific management functions identified.

Audit: FCS_CKM_EXT.3

There are no auditable events foreseen.

FCS_CKM_EXT.3 Key Encrypting Key (KEK) Support

Hierarchical to: No other components.

Dependencies to: FCS_RBG_EXT.1 Random Bit Generation Services

FCS_CKM_EXT.3.1

The TSF shall [**selection**:

- *accept KEK from an enterprise management server*,
- *generate KEK cryptographic keys*
  *[**selection**:*
    - *using a Random Bit Generator as specified in FCS_RBG_EXT.1* ***(from [AppPP])*** *and with entropy corresponding to the security strength of AES key sizes of [**selection**: 128 bit, 256 bit],*
    - *derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6*

  ]

].

Component Leveling

FCS_CKM_EXT.6, Cryptographic Password/Passphrase Conditioning, requires the TSF to implement password/passphrase conditioning using a specified algorithm and with specific constraints on the password/passphrase composition.

Management: FCS_CKM_EXT.6

There are no specific management functions identified.

Audit: FCS_CKM_EXT.6

There are no auditable events foreseen.

FCS_CKM_EXT.6 Cryptographic Password/Passphrase Conditioning

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation
FCS_RBG_EXT.1 Random Bit Generation Services

FCS_CKM_EXT.6.1

The TSF shall support a password/passphrase of up to [**selection**: *[**assignment**: maximum value supported by the platform]*, *[**assignment**: maximum password size, positive integer of 64 or more]*] characters used to generate a password authorization factor.

FCS_CKM_EXT.6.2

The TSF shall allow passwords to be composed of any combination of upper case characters, lower case characters, numbers, and the following special characters: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")", and [**selection**: *[**assignment**: other supported special characters]*, *no other characters*].

FCS_CKM_EXT.6.3

The TSF shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm HMAC-[**selection**: *SHA-256*, *SHA-384*, *SHA-512*], with [**selection**: *[**assignment**: positive integer of 4096 or more] iterations*, *value supported by the platform, greater than 1000*], and output cryptographic key sizes [**selection**: *128*, *256*] that meet the following: [*NIST SP 800-132*].

FCS_CKM_EXT.6.4

The TSF shall not accept passwords less than [**selection**: *a value settable by the administrator*, *[**assignment**: minimum password length accepted by the TOE, must be >= 1]*] and greater than the maximum password length defined in FCS_CKM_EXT.6.1.

FCS_CKM_EXT.6.5

The TSF shall generate all salts using an RBG that meets FCS_RBG_EXT.1 **(from [AppPP])** and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.6.3.

FCS_IV_EXT Initialization Vector Generation

Components in this family define requirements for initialization vector generation.

Component Leveling

FCS_IV_EXT.1, Initialization Vector Generation, specifies the required initialization vector generation methods used by the TSF for various cryptographic algorithms.

Management: FCS_IV_EXT.1

There are no specific management functions identified.

Audit: FCS_IV_EXT.1

There are no auditable events foreseen.

FCS_IV_EXT.1 Initialization Vector Generation

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_IV_EXT.1.1

The TSF shall [**selection**:

- *invoke platform-provided functionality to generate IVs*,
- *generate IVs with the following properties [**selection**:*
  - *CBC: IVs shall be non-repeating and unpredictable*,
  - *CCM: Nonce shall be non-repeating and unpredictable*,
  - *XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer,*
  - *GCM: IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^32 for a given secret key*

  ]

].

FCS_KYC_EXT Key Chaining and Key Storage

Components in this family define requirements for the secure storage of keys through the use of a logical key chain.

Component Leveling

FCS_KYC_EXT.1, Key Chaining and Key Storage, requires the TSF to specify how it implements key chaining.

Management: FCS_KYC_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of the cryptographic functionality.

Audit: FCS_KYC_EXT.1

There are no auditable events foreseen.

FCS_KYC_EXT.1 Key Chaining and Key Storage

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation
FCS_KDF_EXT.1 Cryptographic Key Derivation Function
FCS_SMC_EXT.1 Submask Combining

FCS_KYC_EXT.1.1

The TSF shall maintain a key chain of: [**selection**:

- *a conditioned password as the [FEK]*,
- *[KEKs] originating from **[selection: one or more authorization factors(s), a file encryption enterprise management server]** to **[selection: the FEK(s), a file encryption enterprise management server]** using the following method(s): [**selection**:*
  - *utilization of the platform key storage*,
  - *utilization of platform key storage that performs key wrap with a TSF provided key,*
  - *implementation of key wrapping as specified in FCS_COP.1**(5)**,*
  - *implementation of key combining as specified in FCS_SMC_EXT.1*,
  - *implementation of key encryption as specified in FCS_COP.1**(7)**,*
  - *implementation of key transport as specified in FCS_COP.1**(6)**,*
  - *implementation of key derivation as specified in FCS_KDF_EXT.1*

  *] while maintaining an effective strength of [**selection**:*
  - *[**selection**: 128 bits, 256 bits] for symmetric keys ,*
  - *[**selection**: 128 bits, 192 bits, 256 bits] for asymmetric keys*

  ] commensurate with the strength of the FEK

] and [**selection**:

- *no supplemental key chains*,
- *other supplemental key chains that protect a key or keys in the primary key chain using the following method(s): [**selection**:*

    - *utilization of the platform key storage*,
    - *utilization of platform key storage that performs key wrap with a TSF provided key*,
    - *implementation of key wrapping as specified in FCS_COP.1**(5)**,*
    - *implementation of key combining as specified in FCS_SMC_EXT.1*,
    - *implementation of key encryption as specified in FCS_COP.1**(7)**,*
    - *implementation of key transport as specified in FCS_COP.1**(6)**,*
    - *implementation of key derivation as specified in FCS_KDF_EXT.1*

  ]

].

FCS_VAL_EXT Validation

Components in this family define requirements for validation of data supplied to the TOE and any consequences resulting from failed validation attempts.

Component Leveling

FCS_VAL_EXT.1, Validation, requires the TSF to specify what data is being validated and how the validation is performed.

Management: FCS_VAL_EXT.1

There are no specific management functions identified.

Audit: FCS_VAL_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Minimal: Change to configuration of validation function behavior.

FCS_VAL_EXT.1 Validation

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_VAL_EXT.1.1

The TSF shall perform validation of the [*user*] by [**selection**:

- *receiving assertion of the subject's validity from [**assignment**: Operational Environment component responsible for authentication]*,
- *validating the [**selection**: submask, intermediate key] using the following methods: [**selection**:*
    - *key wrap as specified in FCS_COP.1**(5)**,*
    - *hash the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1**(2) (from [AppPP])** and compare it to a stored hash,*
    - *decrypt a known value using the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1**(1) (from [AppPP])** and compare it against a stored known value*

  ]

].

FCS_VAL_EXT.1.2

The TSF shall require validation of the [*user*] prior to [*decrypting any FEK*].

Component Leveling

FCS_VAL_EXT.2, Validation Remediation, requires the TSF to specify what the TOE's response is in the event of a data validation failure.

Management: FCS_VAL_EXT.2

The following actions could be considered for the management functions in FMT:

- Configuration of the number of failed validation attempts required to trigger corrective behavior.
- Configuration of the corrective behavior to issue in the event of an excessive number of failed validation attempts.

Audit: FCS_VAL_EXT.2

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Minimal: Triggering of excessive validation failure response behavior.

FCS_VAL_EXT.2 Validation Remediation

Hierarchical to: No other components.

Dependencies to: FCS_VAL_EXT.1 Validation

FCS_VAL_EXT.2.1

The TSF shall [**selection**:

- *[perform a key destruction of the FEK(s)] upon a configurable number of consecutive failed validation attempts*,
- *institute a delay such that only [**assignment**: ST author specified number of attempts] can be made within a 24 hour period*,
- *block validation after [**assignment**: ST author specified number of attempts] of consecutive failed validation attempts*,
- *require power cycle/reset the TOE after [**assignment**: ST author specified number of attempts] of consecutive failed validation attempts*

].

FDP_PRT_EXT Protection of Selected User Data

Components in this family define requirements for the TOE's ability to protect sensitive data at rest.

Component Leveling

FDP_PRT_EXT.1, Protection of Selected User Data, requires the TOE to encrypt and decrypt sensitive data using a specified cryptographic algorithm.

Management: FDP_PRT_EXT.1

There are no specific management functions identified.

Audit: FDP_PRT_EXT.1

There are no auditable events foreseen.

FDP_PRT_EXT.1 Protection of Selected User Data

Hierarchical to: No other components.

Dependencies to: FCS_CKM_EXT.4 Cryptographic Key Destruction
FCS_COP.1 Cryptographic Operation

FDP_PRT_EXT.1.1

The TSF shall perform encryption and decryption of the user-selected file (or set of files) in accordance with FCS_COP.1(1) **(from [AppPP])**.

FDP_PRT_EXT.1.2

The TSF shall [**selection**: *invoke platform-provided functionality*, *implement functionality*] to ensure that all sensitive data created by the TOE when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory when the data is no longer needed according to FCS_CKM_EXT.4.

Component Leveling

FDP_PRT_EXT.2, Destruction of Plaintext Data, requires the TOE to destroy any plaintext data that is created as a result of the encryption/decryption process for sensitive data.

Management: FDP_PRT_EXT.2

There are no specific management functions identified.

Audit: FDP_PRT_EXT.2

There are no auditable events foreseen.

FDP_PRT_EXT.2 Destruction of Plaintext Data

Hierarchical to: No other components.

Dependencies to: FCS_CKM_EXT.4 Cryptographic Key Destruction
FDP_PRT_EXT.1 Protection of Selected User Data

FDP_PRT_EXT.2.1

The TSF shall [**selection**: *invoke platform-provided functionality*, *implement functionality*] to ensure that all original plaintext data created when decrypting/encrypting the user-selected file (or set of files) are destroyed in volatile and non-volatile memory according to FCS_CKM_EXT.4 upon completion of the decryption/encryption operation.

FIA_AUT_EXT Authorization

Components in this family define requirements for how subject authorization is performed. Where FIA_UAU in CC Part 2 defines circumstances where authentication is required, this family describes the specific computational methods used to determine whether a subject's presented authentication data is valid.

Component Leveling

FIA_AUT_EXT.1, Subject Authorization, specifies the manner in which the TSF performs user authorization.

Management: FIA_AUT_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of authentication factors.

Audit: FIA_AUT_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Minimal: Failure of authorization function.
- Basic: All use of authorization function.

FIA_AUT_EXT.1 Subject Authorization

Hierarchical to: No other components.

Dependencies to: FCS_CKM_EXT.6 Cryptographic Password/Passphrase Conditioning
FCS_RBG_EXT.1 Random Bit Generation Services

FIA_AUT_EXT.1.1

The TSF shall [**selection**: *implement platform-provided functionality to provide user authorization*, *provide user authorization*] based on [**selection**:

- *a password authorization factor conditioned as defined in FCS_CKM_EXT.6*,
- *an external smart card factor that is at least the same bit-length as the FEK(s), and is protecting a submask that is [**selection**: generated by the TOE (using the RBG as specified in FCS_RBG_EXT.1* ***(from [AppPP])****), generated by the platform] protected using RSA with key size [**selection**: 3072 bits, 4096 bits] with user presence proved by presentation of the smart card and [**selection**: no PIN, an OE defined PIN, a configurable PIN]* ,
- *an external USB token factor that is at least the same security strength as the FEK(s), and is providing a submask generated by the [**selection**: TOE (using the RBG as specified in FCS_RBG_EXT.1* ***(from [AppPP])****), platform]*

].

FPT_KYP_EXT Protection of Key and Key Material

Components in this family define requirements for secure storage of keys.

Component Leveling

FPT_KYP_EXT.1, Protection of Keys and Key Material, requires the TSF to protect stored key data in a specified manner.

Management: FPT_KYP_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of the cryptographic functionality.

Audit: FPT_KYP_EXT.1

There are no auditable events foreseen.

FPT_KYP_EXT.1 Protection of Keys and Key Material

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation
FCS_KDF_EXT.1 Cryptographic Key Derivation Function
FCS_KYC_EXT.1 Key Chaining and Key Storage
FCS_SMC_EXT.1 Submask Combining
FCS_STO_EXT.1 Storage of Credentials

FPT_KYP_EXT.1.1

The TSF shall [**selection**:

- *not store keys in non-volatile memory*,
- *store keys in non-volatile memory only when [**selection**:*
  - *wrapped, as specified in FCS_COP.1(5)*,
  - *encrypted, as specified in FCS_COP.1(7)*,
  - *the plaintext key is stored in the underlying platform's keystore as specified by FCS_STO_EXT.1.1* ***(from [AppPP])***,
  - *the plaintext key is not part of the key chain as specified in FCS_KYC_EXT.1,*
  - *the plaintext key will no longer provide access to the encrypted data after initial provisioning*
  - *the plaintext key is a key split that is combined as specified in FCS_SMC_EXT.1 and another contribution to the split is [**selection**:*
    - *wrapped as specified in FCS_COP.1(5)*,
    - *encrypted as specified in FCS_COP.1(7)*,
    - *derived as specified in FCS_KDF_EXT.1.1 and not stored in non-volatile memory*,
    - *supplied by the enterprise management server*

    ] ,
  - *the plaintext key is stored on an external storage device for use as an authorization factor,*
  - *the plaintext key is used to encrypt a key as specified in FCS_COP.1(7) or wrap a key as specified in FCS_COP.1(5) that is already encrypted as specified in FCS_COP.1(7) or wrapped as specified in FCS_COP.1(5)*

  ]

].

FCS_COP_EXT Cryptographic Operation

Components in this family define requirements for cryptographic operations specific to file encryption.

Component Leveling

FCS_COP_EXT.1, FAK Encryption/Decryption Support, defines requirements for how to protect a file encryption key.

Management: FCS_COP_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of the cryptographic functionality.

Audit: FCS_COP_EXT.1

There are no auditable events foreseen.

FCS_COP_EXT.1 FAK Encryption/Decryption Support

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_COP_EXT.1.1

The FAK shall be protected in the same manner as the FEK, in accordance with FCS_COP.1(5).

FDP_AUT_EXT User Data Authentication

Components in this family define requirements for authentication of protected user data.

Component Leveling

FDP_AUT_EXT.1, Authentication of Selected User Data, requires the TSF to support data authentication and to specify the particular data authentication method that is supported.

Management: FDP_AUT_EXT.1

There are no specific management functions identified.

Audit: FDP_AUT_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Minimal: Failed authentication attempts.
- Basic: All authentication attempts.

FDP_AUT_EXT.1 Authentication of Selected User Data

Hierarchical to: No other components.

Dependencies to: FDP_AUT_EXT.2 Data Authentication Using Cryptographic Keyed-Hash Functions
FDP_AUT_EXT.3 Data Authentication Using Asymmetric Signing and Verification

FDP_AUT_EXT.1.1

The TSF shall perform authentication of the user-selected file (or set of files) and provide notification to the user if modification had been detected.

FDP_AUT_EXT.1.2

The TSF shall implement a data authentication method based on [**selection**: *cryptographic keyed hashing service and verification in accordance with FDP_AUT_EXT.2*, *asymmetric signing and verification in accordance with FDP_AUT_EXT.3*].

Component Leveling

FDP_AUT_EXT.2, Data Authentication Using cryptographic Keyed-Hash Functions, requires the TOE to implement data authentication using a keyed hash function with a FAK as its key.

Management: FDP_AUT_EXT.2

There are no specific management functions identified.

Audit: FDP_AUT_EXT.2

There are no auditable events foreseen.

FDP_AUT_EXT.2 Data Authentication Using cryptographic Keyed-Hash Functions

Hierarchical to: No other components.

Dependencies to: FCS_CKM_EXT.5 File Authentication Key (FAK) Support
FCS_COP.1 Cryptographic Operation
FCS_COP_EXT.1 FAK Encryption/Decryption Support
FCS_RBG_EXT.1 Random Bit Generation Services

FDP_AUT_EXT.2.1

The TSF shall use a cryptographic, keyed hash function in accordance with FCS_COP.1(4) **(from [AppPP])**.

FDP_AUT_EXT.2.2

The TSF shall use a File Authentication Key (FAK) in accordance with FCS_COP_EXT.1 and FCS_CKM_EXT.5 as the secret key to the keyed hash function.

FDP_AUT_EXT.2.3

The TSF shall use the entirety of the ciphertext file as the message input to the keyed hash function.

FDP_AUT_EXT.2.4

The TSF shall concatenate the output of the keyed hash function, the Message Authentication Code (MAC).

FDP_AUT_EXT.2.5

The TSF shall authenticate the encrypted file prior to decryption.

FDP_AUT_EXT.2.6

The TSF shall authenticate the data by comparing the keyed hash output of the ciphertext against the stored MAC.

FDP_AUT_EXT.2.7

The TSF shall notify the user of an unsuccessful authentication and prevent decryption of the ciphertext.

FDP_AUT_EXT.2.8

During verification, the TSF shall verify the MAC is at the end of the ciphertext file.

FDP_AUT_EXT.2.9

The FAK will be generated using a RBG that meets FCS_RBG_EXT.1 **(from [AppPP])**.

Component Leveling

FDP_AUT_EXT.3, Data Authentication Using Asymmetric Signing and Verification, requires the TOE to implement data authentication using a cryptographic signature and hash.

Management: FDP_AUT_EXT.3

There are no specific management functions identified.

Audit: FDP_AUT_EXT.3

There are no auditable events foreseen.

FDP_AUT_EXT.3 Data Authentication Using Asymmetric Signing and Verification

Hierarchical to: No other components.

Dependencies to: FCS_CKM.1 Cryptographic Key Generation
FCS_COP.1 Cryptographic Operation

FDP_AUT_EXT.3.1

The TSF shall use a secure hash function in accordance with FCS_COP.1(2) **(from [AppPP])** with the entire ciphertext file as input to create a hash.

FDP_AUT_EXT.3.2

The TSF shall use a cryptographic signing function in accordance with FCS_COP.1(3) **(from [AppPP])** and must use the hash generated in accordance with FDP_AUT_EXT.3.1 as input to the signing process. Additionally, use of ephemeral key for signing purposes is prohibited.

FDP_AUT_EXT.3.3

The TSF shall use a public and private key pair generated in accordance with FCS_CKM.1(1) **(from [AppPP])** and must use this key pair as part of the cryptographic signing process in accordance with FDP_AUT_EXT.3.2.

FDP_AUT_EXT.3.4

The TSF shall authenticate the ciphertext data prior to decryption.

FDP_AUT_EXT.3.5

The TSF shall notify the user of an unsuccessful authentication and prevent decryption of the ciphertext if such an event were to occur.

FDP_AUT_EXT.3.6

The TSF shall append the signature to the end of the ciphertext file.

FDP_AUT_EXT.3.7

During verification, the TSF shall verify the signature is at the end of the ciphertext file.

FDP_PM_EXT Protection of Data in Power Managed States

Components in this family define requirements for the protection of data in cases where the host platform becomes locked or unpowered.

Component Leveling

FDP_PM_EXT.1, Protection of Data in Power Managed States, requires the TOE to ensure that TSF-protected data does not lose its protections if the host platform is placed in a locked or unpowered state.

Management: FDP_PM_EXT.1

There are no specific management functions identified.

Audit: FDP_PM_EXT.1

There are no auditable events foreseen.

FDP_PM_EXT.1 Protection of Data in Power Managed States

Hierarchical to: No other components.

Dependencies to: FDP_PRT_EXT.1 Protection of Selected User Data

FIA_AUT_EXT.1 User Authorization

FDP_PM_EXT.1.1

The TSF shall protect all data selected for encryption during the transition to the [**assignment**: *powered-down state(s) or locked system states for which this capability is provided*] state as per FDP_PRT_EXT.1.1.

FDP_PM_EXT.1.2

On the return to a powered-on state from the state(s) indicated in FDP_PM_EXT.1.1, the TSF shall authorize the user in the manner specified in FIA_AUT_EXT.1.1 once before any protected data are decrypted.

FDP_PM_EXT.1.3

The TSF shall destroy all key material and authentication factors stored in plaintext when transitioning to a protected state as defined by FDP_PM_EXT.1.1.

FIA_FCT_EXT Authorization Factors

Components in this family define requirements for the use of alternative authorization factors for users to access protected data.

Component Leveling

FIA_FCT_EXT.1, Multi-User Authorization, requires the TSF to maintain differing authorization factors for multiple users.

Management: FIA_FCT_EXT.1

There are no specific management functions identified.

Audit: FIA_FCT_EXT.1

There are no auditable events foreseen.

FIA_FCT_EXT.1 Multi-User Authorization

Hierarchical to: No other components.

Dependencies to: FIA_AUT_EXT.1 User Authorization

FIA_FCT_EXT.1.1

The TSF shall support the use of authorization factors from multiple users that result in unique KEKs.

FIA_FCT_EXT.1.2

The TSF shall support the ability of each user to have files protected by a key chain tied only to that user's credentials.

Component Leveling

FIA_FCT_EXT.2, Authorized Key Sharing, requires the TSF to support some mechanism to share a valid authorization factor between different users.

Management: FIA_FCT_EXT.2

There are no specific management functions identified.

Audit: FIA_FCT_EXT.2

There are no auditable events foreseen.

FIA_FCT_EXT.2 Authorized Key Sharing

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FIA_FCT_EXT.2.1

The TSF shall support [**selection**: *Authorized User Key sharing via key transport as specified in FCS_COP.1(6) Distribution of a shared key from an enterprise management Server*].

FCS_KDF_EXT Cryptographic Key Derivation Function

Components in this family define requirements for the implementation of cryptographic key derivation functions

Component Leveling

FCS_KDF_EXT.1, Cryptographic Key Derivation Function, requires the TSF to specify how it performs key derivation.

Management: FCS_KDF_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of the cryptographic functionality.

Audit: FCS_KDF_EXT.1

There are no auditable events foreseen.

FCS_KDF_EXT.1 Cryptographic Key Derivation Function

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation
FCS_RBG_EXT.1 Random Bit Generation Services

FCS_KDF_EXT.1.1

The TSF shall [*accept [**selection**: a submask generated by an RBG as specified in FCS_RBG_EXT.1**(from [AppPP])**, a conditioned password, an imported submask] to derive an intermediate key, as defined in [**selection**:*

- *NIST SP 800-108 [**selection**: KDF in Counter Mode, KDF in Feedback Mode, KDF in Double-Pipeline Iteration Mode] ,*
- *NIST SP 800-132*

*] using the keyed-hash functions specified in FCS_COP.1**(4) (from [AppPP])**, such that the output is at least of equivalent security strength (in number of bits) to the [FEK]*].

FCS_SMC_EXT Submask Combining

Components in this family define requirements for generation of intermediate keys via submask combining.

Component Leveling

FCS_SMC_EXT.1, Submask Combining, requires the TSF to implement submask combining in a specific manner to support the generation of intermediate keys.

Management: FCS_SMC_EXT.1

The following actions could be considered for the management functions in FMT:

- Configuration of the cryptographic functionality.

Audit: FCS_SMC_EXT.1

There are no auditable events foreseen.

FCS_SMC_EXT.1 Submask Combining

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_SMC_EXT.1.1

The TSF shall [*combine submasks using the following method [**selection**: exclusive OR (XOR), SHA-256, SHA-384, SHA-512, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512] to generate an intermediate key*].

FCS_VAL_EXT Validation Remediation

Components in this family define requirements for validation of data supplied to the TOE and any consequences resulting from failed validation attempts.

Component Leveling

FCS_VAL_EXT.1, Validation, requires the TSF to specify what data is being validated and how the validation is performed.

Management: FCS_VAL_EXT.1

There are no specific management functions identified.

Audit: FCS_VAL_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Minimal: Change to configuration of validation function behavior.

FCS_VAL_EXT.1 Validation

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_VAL_EXT.1.1

The TSF shall perform validation of the [*user*] by [**selection**:

- *receiving assertion of the subject's validity from [**assignment**: Operational Environment component responsible for authentication]*,
- *validating the [**selection**: submask, intermediate key] using the following methods: [**selection**:*
  - *key wrap as specified in FCS_COP.1**(5)**,*
  - *hash the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1**(2) (from [AppPP])** and compare it to a stored hash,*
  - *decrypt a known value using the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1**(1) (from [AppPP])** and compare it against a stored known value*

  ]

].

FCS_VAL_EXT.1.2

The TSF shall require validation of the [*user*] prior to [*decrypting any FEK*].

Component Leveling

FCS_VAL_EXT.2, Validation Remediation, requires the TSF to specify what the TOE's response is in the event of a data validation failure.

Management: FCS_VAL_EXT.2

The following actions could be considered for the management functions in FMT:

- Configuration of the number of failed validation attempts required to trigger corrective behavior.
- Configuration of the corrective behavior to issue in the event of an excessive number of failed validation attempts.

Audit: FCS_VAL_EXT.2

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

- Minimal: Triggering of excessive validation failure response behavior.

FCS_VAL_EXT.2 Validation Remediation

Hierarchical to: No other components.

Dependencies to: FCS_VAL_EXT.1 Validation

FCS_VAL_EXT.2.1

The TSF shall [**selection**:

- *[perform a key destruction of the FEK(s)] upon a configurable number of consecutive failed validation attempts*,
- *institute a delay such that only [**assignment**: ST author specified number of attempts] can be made within a 24 hour period*,
- *block validation after [**assignment**: ST author specified number of attempts] of consecutive failed validation attempts*,
- *require power cycle/reset the TOE after [**assignment**: ST author specified number of attempts] of consecutive failed validation attempts*

].

Appendix E - Key Management Description

The documentation of the product's encryption key management should be detailed enough that, after reading, the evaluator will thoroughly understand the product's key management and how it meets the requirements to ensure the keys are adequately protected. This documentation should include an essay and diagram(s). This documentation is not required to be part of the TSS - it can be submitted as a separate document and marked as developer proprietary.

Essay:

The essay will provide the following information for all keys in the key chain:

- The purpose of the key
- If the key is stored in non-volatile memory
- How and when the key is protected
- How and when the key is derived
- The strength of the key
- When or if the key would be no longer needed, along with a justification
- How and when the key may be shared

The essay will also describe the following topics:

- A description of all authorization factors that are supported by the product and how each factor is handled, including any conditioning and combining performed.
- If validation is implemented, the process for validation shall be described, noting what value is used for validation and the process used to perform the validation. It shall describe how this process ensures no keys in the key chain are weakened or exposed by this process.
- The authorization process that leads to the decryption of the FEK(s). This section shall detail the key chain used by the product. It shall describe which keys are used in the protection of the FEK(s) and how they meet the encryption or derivation requirements including the direct chain from the initial authorization to the FEK(s). It shall also include any values that add into that key chain or interact with the key chain and the protections that ensure those values do not weaken or expose the overall strength of the key chain.
- The diagram and essay will clearly illustrate the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or all of the initial authorization values and the effective strength of the FEK(s) is maintained throughout the key chain.
- A description of the data encryption engine, its components, and details about its implementation (e.g. initialization of the product, drivers, libraries (if applicable), logical interfaces for encryption/decryption, and how resources to be encrypted are identified. The description should also include the data flow from the device's host interface to the device's persistent media storing the data, information on those conditions in which the data bypasses the data encryption engine. The description should be detailed enough to verify all platforms ensure that when the user enables encryption, the product encrypts all selected resources.
- The process for destroying keys when they are no longer needed by describing the storage location of all keys and the protection of all keys stored in non-volatile memory.

Diagram:

- The diagram will include all keys from the initial authorization factor(s) to the FEK(s) and any keys or values that contribute into the chain. It must list the cryptographic strength of each key and indicate how each key along the chain is protected with either options from key chaining requirement. The diagram should indicate the input used to derive or decrypt each key in the chain.
- A functional (block) diagram showing the main components (such as memories and processors) the initial steps needed for the activities the TOE performs to ensure it encrypts the targeted resources when a user or administrator first provisions the product.

Appendix F - Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017. • Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017. • Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.
[AppPP]	Protection Profile for Application Software, Version 1.3
[FIPS140-2]	Federal Information Processing Standard Publication (FIPS-PUB) 140-2, Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, March 19, 2007
[FIPS180-4]	Federal Information Processing Standards Publication (FIPS-PUB) 180-4, Secure Hash Standard, March, 2012
[FIPS186-4]	Federal Information Processing Standard Publication (FIPS-PUB) 186-4, Digital Signature Standard (DSS), National Institute of Standards and Technology, July 2013
[FIPS197]	Federal Information Processing Standards Publication (FIPS-PUB) 197, Specification for the Advanced Encryption Standard (AES), November 26, 2001
[FIPS198-1]	Federal Information Processing Standards Publication (FIPS-PUB) 198-1, The Keyed-Hash Message Authentication Code (HMAC), July 2008
[NIST800-38A]	NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques, 2001 Edition
[NIST800-56A]	NIST Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised), March 2007
[NIST800-56B]	NIST Special Publication 800-56B, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, August 2009
[NIST800-90]	NIST Special Publication 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised), March 2007
[NIST800-132]	NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation, December 2010
[NIST800-38F]	NIST Special Publication 800-38F,Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012

Appendix G - Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
CC	Common Criteria
FAK	File Authentication Key
FEK	File Encryption Key
DRBG	Deterministic Random Bit Generator
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman (see NIST SP 800-56A rev 2, section 6.2.2.2)
FIPS	Federal Information Processing Standards
ISSE	Information System Security Engineers
IT	Information Technology
KDF	Key Derivation Function
KEK	Key Encryption Key
PBKDF	Password-Based Key Derivation Function
PIN	Personnel Identification Number
PKI	Public Key Infrastructure
PP	Protection Profile
PUB	Publication
RBG	Random Bit Generator
SAR	Security Assurance Requirement
SF	Security Function
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification