PP-Module for File Encryption

Version: 2.0
2025-04-29
National Information Assurance Partnership

Revision History

Version	Date	Comment
1.0	2019-07-25	Initial Release
2.0	2025-01-31	Update to CC:2022

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 2.0

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/KeyedHash 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/Sig (from the [AppPP]) and an approved secure hashing function in accordance with FCS_COP.1/Hash (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. Requirements that allow for the platform to provide functionality have selections that identify where it is acceptable.

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

The following sections list Common Criteria and technology terms used in this document.

1.2.1 Common Criteria Terms

Assurance	Grounds for confidence that a TOE meets the SFRs [CC].
Base Protection	Protection Profile used as a basis to build a PP-Configuration.

Profile (Base-PP)	
Collaborative Protection Profile (cPP)	A Protection Profile developed by international technical communities and approved by multiple schemes.
Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation (International Standard ISO/IEC 15408).
Common Criteria Testing Laboratory	Within the context of the Common Criteria Evaluation and Validation Scheme (CCEVS), an IT security evaluation facility accredited by the National Voluntary Laboratory Accreditation Program (NVLAP) and approved by the NIAP Validation Body to conduct Common Criteria-based evaluations.
Common Evaluation Methodology (CEM)	Common Evaluation Methodology for Information Technology Security Evaluation.
Distributed TOE	A TOE composed of multiple components operating as a logical whole.
Extended Package (EP)	A deprecated document form for collecting SFRs that implement a particular protocol, technology, or functionality. See Functional Packages.
Functional Package (FP)	A document that collects SFRs for a particular protocol, technology, or functionality.
Operational Environment (OE)	Hardware and software that are outside the TOE boundary that support the TOE functionality and security policy.
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-PPs.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in an ST.

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	A value submitted by the user, present on the host, or present on a separate protected

factor (AF)	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 User	A user who has been provided Authorization factors by the administrator to use the TOE.
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 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.
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.
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.
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.
Keying Material	The KEK, FEK, authorization factors and random numbers or any other values from which keys are derived.
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.
Passphrase	A string of words that may be used for authorization to the data on the device.
Password	A short string of characters 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. 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

An ST must claim exact conformance to this PP-Module.

The evaluation methods used for evaluating the TOE are a combination of the workunits defined in [CEM] as well as the Evaluation Activities for ensuring that individual SFRs and SARs have a sufficient level of supporting evidence in the Security Target and guidance documentation and have been sufficiently tested by the laboratory as part of completing ATE_IND.1. Any functional packages this PP claims similarly contain their own Evaluation Activities that are used in this same manner.

CC Conformance Claims

This PP-Module is conformant to Part 2 (extended) and Part 3 (extended) of Common Criteria CC:2022, Revision 1.

PP Claim

This PP-Module does not claim conformance to any Protection Profile.

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 3.0
- PP-Module for File Encryption Enterprise Management, Version 1.0

Package Claim

This PP-Module is not conformant to any Functional or Assurance Packages.

3 Security Problem Definition

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.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.KEYSPACE_EXHAUST
: **Brute Force Attack:** An attacker is able to brute force the keyspace of the algorithms used to force disclosure of sensitive data, allowing access to secure information.

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

T.PLAINTEXT_COMPROMISE
: **Plaintext Compromise:** An attacker is able to uncover plaintext remains with forensic tools, allowing access to secure information.

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

T.UNSAFE_AUTHFACTOR_VERIFICATION
: **Flawed Authentication Factor Verification:** An attacker exploits a flaw in the validation or conditioning of the authorization factor, allowing access to secure information.

3.2 Assumptions

These assumptions are made on the Operational Environment (OE) 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 OE 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.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.

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.

3.3 Organizational Security Policies

An organization deploying the TOE is expected to satisfy the organizational security policy listed below in addition to all organizational security policies defined by the claimed Base-PP.

This document does not define any additional OSPs.

4 Security Objectives

The Security Problem described in 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 Operational 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.

4.2 Security Objectives Rationale

This section describes how the assumptions and organizational security policies map to operational environment security objectives.

Table 1: Security Objectives Rationale

Assumption or OSP	Security Objectives	Rationale
A.AUTH_FACTOR	OE.AUTHORIZATION_FACTOR_STRENGTH	The operational environment objective OE.AUTHORIZATION_FACTOR_STRENGTH is realized through A.AUTH_FACTOR.
A.EXTERNAL_FEK_PROTECTION	OE.STRONG_ENVIRONMENT_CRYPTO	The operational environment objective OE.STRONG_ENVIRONMENT_CRYPTO is realized through A.EXTERNAL_FEK_PROTECTION.
A.FILE_INTEGRITY	OE.STRONG_ENVIRONMENT_CRYPTO	The operational environment objective OE.STRONG_ENVIRONMENT_CRYPTO is realized through A.STRONG_OE_CRYPTO.
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_CRYPTO is realized through A.STRONG_OE_CRYPTO.

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 conventions are used for the completion of operations:

- **Refinement** operation (denoted by **bold text** or ~~strikethrough text~~): Is used to add details to a requirement 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: Is indicated by appending the SFR name with a slash and unique identifier suggesting the purpose of the operation, e.g. "/EXAMPLE1."

5.1 Protection Profile for Application Software 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 . The following section describes 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.6 Timing and event of cryptographic key destruction

FCS_CKM.6.1

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.6.2. 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_CKM.6.2

The TSF shall destroy cryptographic keys and keying material specified by FCS_CKM.6.1 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*

]

]

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

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.6 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).

Evaluation Activities

FCS_CKM.6.1
TSS
The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when they should be expected to be destroyed. The evaluator shall verify the TSS provides a high level description of what it means for keys and key material to be no longer needed and when then should be expected to be destroyed.

KMD

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

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

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

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the KMD to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

The evaluator shall check that the KMD identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

If the selection "destruction of all KEKs protecting target key, where none of the KEKs protecting the target key are derived" is included the evaluator shall examine the TOE's keychain in the KMD and identify each instance when a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM.6.2. The evaluator shall verify that all of the keys capable of decrypting the target key are not able to be derived to reestablish the keychain after their destruction.

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

The evaluator shall verify the KMD includes a key lifecycle, that includes a description where key material reside, how the key material is used, how it is determined that keys and key material are no longer needed, and how the material is destroyed once it is not needed and that the documentation in the KMD follows FCS_CKM.6.2 for the destruction.

Guidance
There are a variety of concerns that may prevent or delay key destruction in some cases.

The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information.

The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.

Some examples of what is expected to be in the documentation are provided here.

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

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.

The drive should be healthy and contain minimal corrupted data and should be at end-of-life before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

Tests
These tests are only for key destruction provided by the application, test 2 does not apply to any keys using the selection "new value of a key":

- *Test FCS_CKM.6.1:1: [Conditional; applies when the application does not perform the zeroization (e.g. garbage collecting) for each key held in volatile memory for FCS_CKM.6.2 (assuming the selection "destruction of the reference followed by a request for garbage collection")]*

  Applied to each key held in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the key destruction method was removal of power, then this test is unnecessary.

  The evaluator shall:

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

  2. Cause the cause the TOE or the underlying platform to dump to perform a normal cryptographic processing with the key from Step #1.

  3. Cause the TOE to clear the key.

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

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

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

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

- *Test FCS_CKM.6.1:2: [Conditional; applies when instructing the underlying platform to destroy the key] If new value of a key is selected this test does not apply.*

  Applied to each key held in non-volatile memory and subject to destruction by the TOE.

  The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.

  1. Identify the purpose of the key and what access should fail when it is deleted. (e.g. the file encryption key being deleted would cause data decryption to fail.)

  2. Cause the TOE to clear the key.

  3. Have the TOE attempt the functionality that the cleared key would be necessary for.

  4. The test succeeds if Step #3 fails.

  Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key, as the TOE has no visibility into the inner workings and completely relies on the underlying platform.

- *Test FCS_CKM.6.1:3: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):*

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

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

  3. Cause the TOE to clear the key.

  4. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.
- *Test FCS_CKM.6.1:4: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:*

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

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

  3. Cause the TOE to clear the key.

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

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

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 [256-bit]*

- *using key generation methods compliant with NIST SP 800-133r2,*
- *derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6*

]

] .

Application Note: For keys generated from a password, even if referencing NIST SP 800-133r2 for password-based key generation, "derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6" must be selected so that FCS_CKM_EXT.6 is included.

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.

Evaluation Activities

FCS_CKM_EXT.2.1
TSS
FCS_CKM_EXT.2.1: The evaluator shall review the TSS to determine that a description covering how and when the FEKs are generated exists. The description must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate the FEKs. The evaluator shall verify that the description of how the FEKs are generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account.

Conditional:
If 'using a Random Bit Generator' was selected, the evaluator shall verify that the TSS describes how the functionality described by FCS_RBG_EXT.1 (from the [AppPP]) is invoked to generate FEK. To the extent possible from the description of the RBG functionality in FCS_RBG_EXT.1 (from [AppPP]), the evaluator shall determine that the key size being requested is identical to the key size and mode to be used for the decryption/encryption of the user data (FCS_COP.1/SKC) (from [AppPP]).

Conditional:
If 'using key generation methods compliant with NIST SP 800-133r2' was selected, the evaluator shall verify that the TSS describes how the functionality described by NIST SP 800-133r2 is implemented to generate the FEK. The evaluator shall verify that the description of the key generation method matches the methods described in SP 800-133r2 and that the FEK is chained to an approved RBG. The evaluator shall verify that the key size and mode being requested is identical to the key size and mode to be used for the decryption/encryption of the user data (FCS_COP.1/SKC) (from [AppPP]).

Conditional:
If 'derived from a password/passphrase' is selected, the examination of the TSS section is performed as part of FCS_CKM_EXT.6 evaluation activities.

FCS_CKM_EXT.2.2: The evaluator shall verify the TSS describes how a FEK is used for a protected resource and associated with that resource. The evaluator confirms that-per this description-the FEK is unique per resource (file or set of files) and that the FEK is established using the mechanisms specified in FCS_CKM_EXT.2.1).

KMD

None.

Guidance
The evaluator shall review the instructions in the AGD guidance to determine that any explicit actions that need to be taken by the user to establish a FEK exist-taking into account any differences that arise from different platforms-and are consistent with the description in the TSS.

Tests
None.

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 establishment as specified in FCS_CKM.2 (from the Base-PP)*
  - *implementation of key derivation as specified in FCS_KDF_EXT.1*
  - *implementation of key wrapping as specified in FCS_COP.1/KW*
  - *implementation of key encryption as specified in FCS_COP.1/KE*
  - *implementation of key combining as specified in FCS_SMC_EXT.1*
  - *implementation of key transport as specified in FCS_COP.1/KT*

  ] while maintaining an effective strength of 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 establishment as specified in FCS_CKM.2 (from the Base-PP)*
  - *implementation of key encryption as specified in FCS_COP.1/KE*
  - *implementation of key transport as specified in FCS_COP.1/KT*
  - *implementation of key wrapping as specified in FCS_COP.1/KW*
  - *implementation of key derivation as specified in FCS_KDF_EXT.1*
  - *implementation of key combining as specified in FCS_SMC_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 threat 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).

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.

Evaluation Activities

FCS_KYC_EXT.1.1
TSS
The evaluator shall verify the TSS contains a high level description of all keychains and authorization methods selected in FIA_AUT_EXT.1 that are used to protect the KEK or FEK.

KMD

The evaluator shall examine the KMD to ensure it describes each key chain in detail, and these descriptions correspond with the selections of the requirement. The description of each key chain shall be reviewed to ensure the options for maintaining the key chain are documented.

The evaluator shall verify the KMD to ensure that it describes how each key chain process functions, such that it does not expose any material that might compromise any key in the chain. A high-level description should include a diagram illustrating the keychain(s) implemented and detail where all keys and keying material is stored or how the keys or key material are derived. The evaluator shall examine the primary key chain to ensure that at no point the chain could be broken without a cryptographic exhaust or knowledge of the KEK or FEK and the effective strength of the FEK is maintained throughout the Key Chain as specified in the requirement.

Guidance
None.

Tests
None.

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/KW*
  - *hash the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1/Hash (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/SKC (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.

Evaluation Activities

FCS_VAL_EXT.1.1
TSS
Conditional:
If 'validating' is selected in FCS_VAL_EXT.1.1, the evaluator shall examine the TSS to determine which authorization factors support validation.

The evaluator shall examine the TSS to ensure that it contains a high-level description of how the submasks are validated. If multiple submasks are used within the TOE, the evaluator shall verify that the TSS describes how each is validated (e.g., each submask validated before combining, once combined validation takes place).

Conditional:

If 'receiving assertion' is selected in FCS_VAL_EXT.1.1, the evaluator shall examine the TSS to verify that it describes the environments that can be leveraged with the TOE and how each claims to perform validation. The evaluator shall ensure that none of the stated platform validation mechanisms weaken the key chain of the product.

KMD

None.

Guidance
If the validation functionality is configurable, the evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established.

Tests
There are no test activities for this requirement.

FCS_VAL_EXT.2 Validation Remediation

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.

Evaluation Activities

FCS_VAL_EXT.2.1
TSS
The evaluator shall examine the TSS to determine which remediation options are supported for which authentication options.

KMD

None.

Guidance
If the remediation functionality is configurable, the evaluator shall examine the operational guidance to ensure it describes how to configure the TOE to ensure the limits regarding validation attempts can be established.

Tests
The evaluator shall perform the following tests:

- *Test FCS_VAL_EXT.2.1:1: The evaluator shall determine the limit on the average rate of the number of consecutive failed authorization attempts. For each authentication factor supported, the evaluator will test the TOE by entering that number of incorrect authorization factors in consecutive attempts to access the protected data. If the limit mechanism includes any "lockout" period, the time period tested should include at least one such period. Then the evaluator will verify that the TOE behaves as described in the TSS.*

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/SKC (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.6.

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.6.

Evaluation Activities

FDP_PRT_EXT.1.1
TSS

The evaluator shall examine the TSS to ensure there is a high-level description of how the FEK is protected.

The evaluator shall examine the TSS to ensure there is a description of how the FEK is protected.

The evaluator shall examine the TSS to ensure that it describes all temporary files/resources created or memory used during the decryption/encryption process and when those files/resources or memory is no longer needed.

The TSS shall describe how the TSF or TOE platform deletes the non-volatile memory (for example, files) and volatile memory locations after the TSF is done with its decryption/encryption operation.

Guidance

There are no additional guidance evaluation activities for FDP_PRT_EXT.1.2.

Tests

- *Test FDP_PRT_EXT.1.1:1: This test only applies for application provided functionality.*

1. Using a file editor, create and save a text file that is encrypted per the evaluation configured encryption policy. The contents of the file will be limited to a known text pattern to ensure that the text pattern will be present in all encryption/decryption operations performed by the TOE.

2. Exit the file editor so that the file (including its known text pattern) has "completed the decryption/encryption operation" and process memory containing the known text pattern is released.

3. The evaluator will take a dump of volatile memory and search the retrieved dump for the known pattern. The test fails if the known plaintext pattern is found in the memory dump.

4. The evaluator will search the underlying non-volatile storage for the known pattern. The test fails if the known plaintext pattern is found in the search.

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.6 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.6.

Evaluation Activities

FDP_PRT_EXT.2.1
TSS
The evaluator shall examine the TSS to ensure that it describes all temporary file (or set of files) that are created in the filesystem of the host during the decryption/encryption process, and that the TSS describes how these files are deleted after the TSF is done with its decryption/encryption operation. Note that if other objects/resources are created on the host that are 1) persistent and 2) visible to other processes (users) on that host that are not filesystem objects, those objects shall be identified and described in the TSS as well.

KMD

None.

Guidance
None.

Tests

- *Test FDP_PRT_EXT.2.1:1: If the TSS creates temporary files/resources during file decryption/encryption, the evaluator shall perform the following tests to verify that the temporary files/resources are destroyed. If the product supported shared files per FIA_FCT_EXT.2, this test must be repeated with a shared file. The evaluator shall use a tool (e.g., procmon for a Windows system) that is capable of monitoring the creation and deletion of files during the decryption/encryption process is performed. A tool that can search the contents of the hard drive (e.g., WinHex) will also be needed. The tools used to perform the monitoring shall be identified in the test report.*

  (Creating an encrypted document)
  - *Open an editing application.*
  - *Create a special string inside the document. The string could be 5-10 words. It is recommended to remove the spaces. This will create a one page document.*
  - *Start the file monitoring tool.*
  - *Save and close the file.*

- *Encrypt the file using the TOE (if the TOE does not encrypt automatically for the user).*

Analysis Steps

- *If needed, exit/close the TOE.*
- *Stop the file monitoring tool. View the results. Identify any temporary files that were created during the encryption process. Examine to see if the temporary files were destroyed when the TOE closed.*
- *If temporary files remain, these temporary files should be examined to ensure that no plaintext data remains. If plaintext data is found in these files, the test fails.*
- *Search the contents of the hard drive (using the second tool) for the plaintext string used above. (The search should be performed using both ASCII and Unicode formats.)*
- *If the string is found, this means that plaintext from the test fails.*

(Creating, encrypting a blank document and then adding text):

- *Encrypt a blank document using the tool.*
- *Create a special string inside the document. The string could be 5-10 words. It is recommended to remove the spaces. This will create a one page document.*
- *Start the file monitoring tool.*
- *Save and close the file.*
- *Perform the "Analysis Steps" listed above.*

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 asymmetric keys as defined in FCS_CKM.1.1/AK (from [AppPP]) 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.

Evaluation Activities

FIA_AUT_EXT.1.1
The evaluation activities for this component will be driven by the selections made by the ST author. This section describes evaluation activities for all possible selections in an ST; it should be understood that if a capability is not selected in the ST, the noted evaluation activity does not

need to be performed.

TSS

The evaluator shall examine the TSS to ensure that it describes how user authentication is performed. The evaluator shall verify that the authorization methods listed in the TSS are specified and included in the requirements in the ST.

Requirement met by the TOE

The evaluator shall first examine the TSS to ensure that the authorization factors specified in the ST are described. For password-based factors the examination of the TSS section is performed as part of FCS_CKM_EXT.6 Evaluation Activities. Additionally in this case, the evaluator shall verify that the operational guidance discusses the characteristics of external authorization factors (e.g., how the authorization factor must be generated; format(s) or standards that the authorization factor must meet) that are able to be used by the TOE. If other authorization factors are specified, then for each factor, the TSS specifies how the factors are input into the TOE.

Requirement met by the platform

The evaluator shall examine the TSS to ensure a description is included for how the TOE is invoking the platform functionality and how it is getting an authorization value that has appropriate entropy.

KMD

None.

Guidance

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

Tests

The evaluator shall ensure that authorization using each selected method is tested during the course of the evaluation, setting up the method as described in the operational guidance and ensuring that authorization is successful and that failure to provide an authorization factor results in denial to access to plaintext data.

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

5.2.4 Security Management (FMT)

FMT_SMF.1/FE Specification of File Encryption Management Functions

FMT_SMF.1.1/FE

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.6.2***
- ***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***
- ***disable cryptographic functionality***
- ***[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.6.2" 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.

Evaluation Activities

FMT_SMF.1.1/FE
The evaluation activities for this component will be driven by the selections made by the ST author. This section describes evaluation activities for all possible selections in an ST; it should be understood that if a capability is not selected in the ST, the noted evaluation activity does not need to be performed. The following sections are divided up into "Required Activities" and "Conditional Activities" for ease of reference. If password or passphrase authorization factors are implemented by the TOE, then the appropriate "change" selection must be included. 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 implemented 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 "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.
TSS
Conditional Activities: For all selected events,the evaluator shall examine the TSS to ensure that it describes the sequence of activities that take place from an implementation perspective when this activity is performed (for example, how it determines which resources are associated with the KEK, the decryption and re-encryption process), and ensure that the KEK and FEK are not exposed during this change.

Cryptographic Configuration: None for this requirement.

KMD

None.

Guidance
Conditional Activities: The evaluator shall examine the Operational Guidance for each included selection to ensure that it either requires no configuration or describes how the functionality is configured.

Cryptographic Configuration: The evaluator shall determine from the TSS for other requirements (FCS_*, FDP_PRT_EXT, FIA_AUT_EXT) what portions of the cryptographic functionality are configurable. The evaluator shall then review the AGD documentation to determine that there are instructions for manipulating all of the claimed mechanisms.

Tests
Cryptographic Erase: If the TOE uses stored FEKS or KEKs, the evaluator shall examine the key chain to determine that the keys destroyed by a cryptographic erase will result in the data becoming unrecoverable. Testing for this activity is performed for other components in this PP-Module.

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/KW*
  - *encrypted, as specified in FCS_COP.1/KE*
  - *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/KW*
  - *encrypted as specified in FCS_COP.1/KE*
  - *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/KE or wrap a key as specified in FCS_COP.1/KW that is already encrypted as specified in FCS_COP.1/KE or wrapped as specified in FCS_COP.1/KW*

]

] .

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.

Evaluation Activities

FPT_KYP_EXT.1.1
TSS
The evaluator shall verify the TSS for a high level description of the method(s) used to protect keys stored in non-volatile memory.

KMD

The evaluator shall verify the KMD to ensure it describes the storage location of all keys and the protection of all keys stored in non-volatile memory. The description of the key chain shall be reviewed to ensure FCS_COP.1/KW is followed for the storage of wrapped or encrypted keys in non-volatile memory and plaintext keys in non-volatile memory meet one of the criteria for storage.

Guidance
None.

Tests
None.

5.3 TOE Security Functional Requirements Rationale

The following rationale provides justification for each SFR for the TOE, showing that the SFRs are suitable to address the specified threats:

Table 2: SFR Rationale

Threat	Addressed by	Rationale
T.KEYING_MATERIAL_COMPROMISE	FCS_CKM.6	Mitigates the threat by destroying keys and key material when not needed.
	FPT_KYP_EXT.1	Mitigates the threat by protecting keys stored on non-volatile memory.
T.KEYSPACE_EXHAUST	FCS_CKM_EXT.2	Mitigates the threat by generating FEKs in a secure manner.
	FCS_CKM_EXT.3 (selection-based)	Mitigates the threat by using or generating secure KEKs.
	FCS_CKM_EXT.6	Mitigates the threat by conditioning a password or passphrase

	(selection-based)	before using it for key-related functions.
	FCS_COP.1/KE (selection-based)	Mitigates the threat by encrypting keys using a specified secure algorithm.
	FCS_COP.1/KT (selection-based)	Mitigates the threat by transporting keys using a specified secure algorithm.
	FCS_COP.1/KW (selection-based)	Mitigates the threat by wrapping keys using a specified secure algorithm.
	FCS_COP.1/SKC (from Base-PP)	Mitigates the threat by encrypting/decrypting according to a specified approved algorithm.
	FCS_KYC_EXT.1	Mitigates the threat by maintaining and storing secure key chains.
	FCS_SMC_EXT.1 (selection-based)	Mitigates the threat by combining submasks using a secure method.
	FCS_VAL_EXT.1	Mitigates the threat by using validation elements that are generated or protected in a secure manner.
T.MANAGEMENT_ ACCESS	FMT_MEC_EXT.1 (from Base-PP)	Mitigates the threat by storing configuration in a protected manner.
	FMT_SMF.1/FE	Mitigates the threat by enumerating and protecting management functions.
T.PLAINTEXT_ COMPROMISE	FDP_PRT_EXT.1	Mitigates the threat by encrypting a file or files.
	FDP_PRT_EXT.2	Mitigates the threat by destroying original plaintext data is once the original data is decrypted or encrypted.
	FDP_PRT_EXT.3 (optional)	Mitigates the threat by destroying or encrypting temporary files when decrypting or encryption of files are completed.
T.UNAUTHORIZED_ DATA_ACCESS	FCS_CKM_EXT.5 (optional)	Mitigates the threat by authenticating data using a FAK.
	FCS_COP_EXT.1 (optional)	Mitigates the threat by protecting a FAK with the same protections as a FEK.
	FCS_COP.1/SKC (from Base-PP)	Mitigates the threat by encrypting/decrypting according to a specified approved algorithm.
	FDP_AUT_EXT.1 (optional)	Mitigates the threat by verifying the authenticity of files and providing notice if modification is detected.
	FDP_AUT_EXT.2 (optional)	Mitigates the threat by authenticating the encrypted data using a keyed-hash function.
	FDP_AUT_EXT.3 (optional)	Mitigates the threat by authenticating the encrypted data using an asymmetric signing and verification function.
	FDP_PM_EXT.1 (optional)	Mitigates the threat by encrypting data during a power-managed state.
	FDP_PRT_EXT.1	Mitigates the threat by encrypting a file or files in accordance with a specified algorithm.
	FDP_PRT_EXT.2	Mitigates the threat by ensuring original plaintext data is destroyed once the data is decrypted or encrypted.
	FDP_PRT_EXT.3 (optional)	Mitigates the threat by removing or encrypting temporary files created during encryption/decryption upon completion of the encryption/decryption.
	FIA_FCT_EXT.1 (optional)	Mitigates the threat by ensuring that files can be encrypted with unique KEKs per user.
	FIA_FCT_EXT.2 (optional)	Mitigates the threat by providing a key sharing mechanism to safely share files between users.
T.UNSAFE_ AUTHFACTOR_ VERIFICATION	FCS_VAL_EXT.1	Mitigates the threat by requiring the user be validated using an appropriate validation factor before any decryption occurs.
	FIA_AUT_EXT.1	Mitigates the threat by authorizing the user using an appropriate

authorization factor.

6 Consistency Rationale

6.1 Protection Profile for Application Software

6.1.1 Consistency of TOE Type

When this PP-Module is used to extend the AppPP, the TOE type for the overall TOE is still an application. The TOE boundary is simply extended to include File Encryption functionality that is provided by the application.

6.1.2 Consistency of Security Problem Definition

Table 3: Consistency of Security Problem Definition (App PP base)

PP-Module Threat, Assumption, OSP	Consistency Rationale
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.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.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.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.
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.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.
A.AUTH_FACTOR	
A.EXTERNAL_FEK_PROTECTION	
A.FILE_INTEGRITY	
A.SHUTDOWN	
A.STRONG_OE_CRYPTO	

6.1.3 Consistency of OE Objectives

Table 4: Consistency of OE Objectives (App PP base)

PP-Module OE 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 PP is being used for its intended purpose. The rationale for why this does not conflict with the claims defined by the App PP are as follows:

Table 5: Consistency of Requirements (App PP base)

PP-Module Requirement	Consistency Rationale
Modified SFRs	
This PP-Module does not modify any requirements when the App PP is the base.	
Additional SFRs	
This PP-Module does not add any requirements when the App PP is the base.	
Mandatory SFRs	
FCS_CKM.6	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_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_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.
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.
FDP_PRT_EXT.1	This SFR is consistent with the Base-PP because it is a specific application of the FCS_COP.1/SKC 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/FE	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

[illegible]	[illegible] 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.

Objective SFRs

This PP-Module does not define any Objective requirements.

Implementation-dependent SFRs

This PP-Module does not define any Implementation-dependent requirements.

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/KE	This SFR defines key encryption functionality that is outside the scope of the original cryptographic operations defined in the Base-PP.
FCS_COP.1/KT	This SFR defines key transport functionality that is outside the scope of the original cryptographic operations defined in the Base-PP.
FCS_COP.1/KW	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_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.

Appendix A - Optional SFRs

A.1 Strictly Optional Requirements

A.1.1 Cryptographic Support (FCS)

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**/KW** 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.

Evaluation Activities

FCS_CKM_EXT.5.1
TSS
FCS_CKM_EXT.5.1: The evaluator shall examine the TSS to determine how the FAK is stored (or not stored) in memory.

FCS_CKM_EXT.5.2: The evaluator shall examine the TSS to determine that it describes how a FAK is created for a protected resource and associated with that resource; protection of the FAK itself is covered by FCS_COP_EXT.1. The evaluator confirms that-per this description-the FAK is unique per resource (file or set of files) and that the FAK is created using a DRBG.

FCS_CKM_EXT.5.3: The TSS must detail that the FAKs are generated on the client machine and are not generated on an external server.

FCS_CKM_EXT.5.4: FCS_CKM.6 contains the requirements necessary to ensure that plaintext keys and key material do not remain in plaintext form in the TSF's non-volatile memory space. In TOEs where the FAK is protected with a KEK, the FAK will need to be encrypted and stored in non-volatile memory when not being used to decrypt/encrypt a file. (Typically, the encrypted FAK is stored in the meta-data of the encrypted file(s).) The evaluator shall examine the TSS to ensure that it describes how the FAK is encrypted, both after its initial creation and after it has been decrypted for use (note that in the entirely likely possibility that the FAK is not re-encrypted, then this case must be indicated in the TSS and the description for FCS_CKM.6 will cover disposal of the plaintext FEK and FAK). The evaluator shall further check to ensure that the TSS describes how the FAK and any other associated meta-data necessary to decrypt the file or set of files are associated with the resource. This description can be combined with the description required for FCS_COP_EXT.1.

KMD

None.

Guidance
None.

Tests
An example ciphertext file generated via the TOE shall be provided to the evaluator with the accompanying FAK and prerequisite authorization information used for encryption. The evaluator will use the TOE in conjunction with a debugging or forensics utility to attempt an authentication of the ciphertext file using the provided authorization information. The evaluator will then terminate processing of the TOE and perform a search through non-volatile memory using the provided FAK string. The evaluator must document each command, program or action taken during this process, and must confirm that the FAK was never written to non-volatile memory. This test must be performed three times to ensure repeatability. If during the course of this testing the evaluator finds that the FAK was written to non-volatile memory, they should be able to identify the cause (i.e. the TOE wrote the FAK to disk, the TOE platform dumped volatile memory as a page file, etc.), and document the reason for failure to comply with the requirement.

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 **/KW** .

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.

Evaluation Activities

FCS_COP_EXT.1.1
TSS
The evaluator shall follow the evaluation activities as laid out in FCS_COP.1/KW to assert proper FAK protection.

KMD

None.

Guidance
None.

Tests
None.

A.1.2 User Data Protection (FDP)

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.

Evaluation Activities

FDP_AUT_EXT.1.1
TSS
The evaluator shall examine the TSS to determine that it lists each type of resource that can be authenticated (e.g., file, directory) and what "authenticated" means in terms of the resource (e.g., "authenticating a directory" means that all of the files contained in the directory are authenticated, but the data in the directory itself (which are filenames and pointers to the files) are not authenticated).

The evaluator shall also confirm that the TSS describes how each type of resource listed is authenticated by the TOE and how authentication measures are added to each resource (e.g. taking all the encrypted files through a MAC function and appending the MAC to the set of files). The evaluator shall ensure that this description includes the case where an existing file or set of files has authentication measures added for the first time; a new file or set of files is created and adds authentication measure; an existing file or set of files updates or replaces its existing authentication measures (that is, it had a MAC appended to the data; it was authenticated and decrypted (by the TOE) for use by the user, and is then subsequently re-encrypted with an updated MAC); and corresponding decryption scenarios. If other scenarios exist due to product implementation/features, the evaluator shall ensure that those scenarios are covered in the TSS as well.

KMD

None.

Guidance
If the TOE creates temporary objects and these objects can be protected through administrative measures (e.g., the TOE creates temporary files in a designated directory that can be protected through configuration of its access control permissions), then the evaluator shall check the Operational Guidance to ensure that these measures are described.

If there are special measures necessary to configure the method by which the file or set of files are authenticated (e.g., choice of function used, additional keys, etc.), then those instructions shall be included in the Operational Guidance and verified by the evaluator. This includes, for instance, lists of allowed platforms, libraries, and devices, and instructions for using them. In these cases, the evaluator checks to ensure that all non-TOE products used to satisfy the requirements of the ST that are described in the Operational Guidance are consistent with those listed in the ST, and those tested by the evaluation activities of this PP-Module.

Tests
The evaluator shall also perform the following tests. These tests must be performed for each data authentication feature and platform claimed in the ST; all instructions for configuring the TOE and each of the environments must be included in the Operational Guidance and used to establish the test configuration.

For each resource and data authentication scenario listed in the TSS, the evaluator shall ensure that the TSF is able to successfully add authentication measures and authenticate the resource using the following methodology.

Monitor the temporary resources being created (if any) and deleted by the TSF-the tools used to perform the monitoring (e.g., procmon for a Windows system) shall be identified in the test report. The evaluator shall ensure that these resources are consistent with those identified in the TSS, and that they are protected as specified in the Operational Guidance and are deleted when the decryption/encryption and authentication operations are completed.

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 **/KeyedHash** (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 [**selection**: *a RBG that meets FCS_RBG_EXT.1 (from [AppPP]), key generation methods compliant with NIST SP 800-133r2*] .

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.6.

Evaluation Activities

FDP_AUT_EXT.2.1
TSS
The evaluator shall check the TSS section to confirm that it describes how a request for each type of supported resource (file (or set of files)) will result in data authentication using a keyed hash function. The evaluator will confirm that the TOE will respond appropriately to a failed authentication, to include notifying the user of an invalid authentication and preventing decryption. The evaluator will confirm that any file encryption utility will be able to identify where the MAC is placed.

The evaluator will confirm that a FAK is used as part of the authentication process and will identify the keyed hash function utilized.

Conditional:
If 'using a Random Bit Generator' was selected, the evaluator shall verify that the TSS describes how the functionality described by FCS_RBG_EXT.1 (from the [AppPP]) is used to generate the FAK.

Conditional:
If 'key generation methods compliant with NIST SP 800-133r2' was selected, the evaluator shall verify that the TSS describes how the functionality described by NIST SP 800-33r1 is implemented to generate the FAK. The evaluator shall verify that the description of the key generation method matches the methods described in SP 800-133r2 and that the FAK is chained to an approved RBG.

KMD

None.

Guidance
It is encouraged for every implementation to use a FAK that is wholly different and independently generated from the FEK.

Tests
The evaluator shall perform the following test:

- *Test FDP_AUT_EXT.2.1:1: Create an encrypted file and confirm that authentication of this file using the correct FAK will result in a successful decryption.*
- *Test FDP_AUT_EXT.2.1:2: Modify an arbitrary number of bits of ciphertext and attempt to*

run the authentication and decryption operations on the file. Assert that the TOE successfully identified the forged ciphertext file and notified the user.

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 **/Hash** (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 **/SigGen** (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 **/AK** (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.

Evaluation Activities

FDP_AUT_EXT.3.1
TSS
The evaluator shall check the TSS section to confirm that it describes how a request for each type of supported resource (file (or set of files)) will result in data authentication using a secure hash and cryptographic signing process. The evaluator will confirm that the supplied public and private key pair were generated in accordance with FCS_CKM.1(1). The evaluator will confirm that the entire ciphertext file was used to create the hash and that the hash was used as input to the cryptographic signing function. The evaluator will confirm that the TSF notifies the user of an unsuccessful authentication and prevents decryption. The evaluator shall confirm that the signature is appended to the end of the ciphertext file.

KMD

None.

Guidance
None.

Tests
The evaluator shall perform the following test:

- *Test FDP_AUT_EXT.3.1:1: Create an encrypted file and demonstrate that authentication of this file using the correct keying material will be successful.*
- *Test FDP_AUT_EXT.3.1:2: Modify an arbitrary number of bits of ciphertext and attempt to run the authentication and decryption operations on the file. Assert that the TOE successfully identified the forged ciphertext file and notified the user.*

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.

Evaluation Activities

FDP_PM_EXT.1.1
TSS
The evaluator shall examine the TSS to ensure that it describes the state(s) that are supported by this capability. For each state, the evaluator ensures that the TSS contains a description of how the state is entered, and the actions of the TSF on entering the state, specifically addressing how multiple open resources (of each type) are protected, and how keying material associated with these resources is protected (if different from that described elsewhere). The TSF shall also describe how the state is exited, and how the requirements are met during this transition to an operational state.

The evaluator shall verify the TSS provides a description of what keys and key material are destroyed when entering any protected state.

KMD

The evaluator shall verify the KMD includes a description of the areas where keys and key material reside. The evaluator shall verify the KMD includes a key lifecycle that includes a description where key material reside, how the key material is used, and how the material is destroyed once a claimed power state is entered and that the documentation in the KMD follows FCS_CKM.6.2 for the destruction.

Guidance
The evaluator shall check the Operational Guidance to determine that it describes the states that are supported by the TOE, and provides information related to the correct configuration of these modes and the TOE.

The evaluator shall validate that guidance documentation contains clear warnings and information on conditions in which the TOE may end up in a non-protected state. In that case it must contain mitigation instructions on what to do in such scenarios.

Tests
The following tests must be performed by the evaluator for each supported State, type of resource, platform, and authorization factor:

- *Test FDP_PM_EXT.1.1:1: Following the Operational guidance, configure the Operational Environment and the TOE so that the lower power state of the platform is enabled and protected by the TOE. Open several resources (documented in the test report) that are protected. Invoke the lower power state. On resumption of normal power attempt to access a previously-opened protected resource, observe that an incorrect entry of the authorization factor(s) does not result in access to the system, and that correct entry of the authorization factor(s) does result in access to the resources.*

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.

Evaluation Activities

FDP_PRT_EXT.3.1
TSS
The evaluator shall examine the TSS to ensure that it describes how the TOE detects and encrypts temporary files (or set of files) that are created in the filesystem of the host by third party products.

KMD

None.

Guidance
[conditional] If any configuration is required for this process the evaluator shall verify it is described in the guidance documentation.

Tests
The evaluator shall utilize any third party application that would be protected under this protection to generate files, then verify those files are being encrypted.

A.1.3 Identification and Authentication (FIA)

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.

Evaluation Activities

FIA_FCT_EXT.1.1
TSS
The evaluator shall examine the TSS to determine that it identifies each of the resource protected in encrypted form and the key chain that protects that resource.

The evaluator shall examine the TSS to verify key chains are separate ensuring resources are protected from other users, with the exception of files permitted to be shared under the mechanism described in FIA_FCT_EXT.2.

KMD

None.

Guidance
The evaluator shall examine the operation guidance to determine that it contains instructions on how to establish multiple accounts and protect resources from other users. If different for different underlying platforms, the evaluator determines that all platforms listed in the ST are addressed.

Tests
The evaluator shall ensure that different users using different authorization factors are unable to decrypt each others protected resources for each type of protected resource identified in the TSS. The test succeeds if the users are unable to decrypt resources not chained to them, with the exception of any resources linked in FIA_FCT_EXT.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**/KT**, Authorized User Key sharing via key wrapping as specified in FCS_COP.1**/KW**, Distribution of a shared key from [an enterprise management server] as specified in FCS_CKM.2* ***(from Base-PP)***] .

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.

Evaluation Activities

FIA_FCT_EXT.2.1
TSS
The evaluator shall examine the TSS to determine that it identifies each of the resources that is shareable in encrypted form (for instance, encrypted files may be shareable among users, but encrypted directories may not), and the method by which the resource can be shared among users with different authorization factors.

The evaluator shall examine the operation guidance to determine that it contains instructions on how to set up and share resources with other users, if additional actions are necessary due to use of the encryption product. If different for different underlying platforms, the evaluator determines that all platforms listed in the ST are addressed.

KMD

None.

Guidance
The evaluator shall examine the operation guidance to determine that it contains instructions on how to set up and share resources with other users, if additional actions are necessary due to use of the encryption product. If different for different underlying platforms, the evaluator determines that all platforms listed in the ST are addressed.

Tests
- *Test FIA_FCT_EXT.2.1:1: For each type of resource that is identified in the TSS as sharable in its encrypted form, the evaluator shall ensure that different users using different authorization factors are able to successfully access the resource using different authorization factors. This should include making changes to the resource to ensure that the same resource is being shared, and that a per-user copy of the resource is not being made.*
- *Test FIA_FCT_EXT.2.1:2: For each type of resource that is identified in the TSS as sharable in its encrypted form, the evaluator shall ensure that a different unauthorized user is unable to access the encrypted file shared. This test succeeds if the unauthorized user is unable to access a file shared between other authorized users.*

A.2 Objective Requirements

This PP-Module does not define any Objective SFRs.

A.3 Implementation-dependent Requirements

This PP-Module does not define any Implementation-dependent SFRs.

Appendix B - Selection-based Requirements

B.1 Cryptographic Support (FCS)

FCS_CKM_EXT.3 Key Encrypting Key (KEK) Support

The inclusion of this selection-based component depends upon selection in 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 and with entropy corresponding to the security strength of AES key sizes of 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.

Evaluation Activities

FCS_CKM_EXT.3.1
TSS
The evaluator shall review the TSS to determine that a description covering how and when KEK(s) are generated exists. The description must cover all environments on which the TOE is claiming conformance, and include any preconditions that must exist in order to successfully generate the KEKs. The evaluator shall verify that the description of how the KEK(s) are generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account.

Conditional:
If using a RBG was selected the evaluator shall examine the TSS and verify that it describes how the functionality described by FCS_RBG_EXT.1 (from the [AppPP]) is invoked to generate KEK(s). To the extent possible from the description of the RBG functionality in FCS_RBG_EXT.1 (from [AppPP]), the evaluator shall determine that the key size being requested is identical to the key size selected.

Conditional:
If derived from a password/passphrase is selected the examination of the TSS section is performed as part of FCS_CKM_EXT.6 evaluation activities.

KMD

None.

Guidance
The evaluator shall review the instructions in the AGD guidance to determine that any explicit actions that need to be taken by the user to establish a KEK exist-taking into account any differences that arise from different platforms-and are consistent with the description in the TSS.

Tests
None.

FCS_CKM_EXT.6 Cryptographic Password/Passphrase Conditioning

The inclusion of this selection-based component depends upon selection in FCS_CKM_EXT.2.1, FIA_AUT_EXT.1.1, 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 [**selection**: *HMAC-SHA-384*, *HMAC-SHA-512*] , with [**selection**: *[**assignment**: positive integer of 10,000 or more] iterations*, *value supported by the platform, greater than 1000*] , and output cryptographic key sizes *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 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_KYC_EXT.1.

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 10,000 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.

Evaluation Activities

FCS_CKM_EXT.6.1
TSS
FCS_CKM_EXT.6.1: There are two aspects of this component that require evaluation: passwords/passphrases of the length specified in the requirement (at least 64 characters or a length defined by the platform) are supported, and that the characters that are input are subject to the selected conditioning function. These activities are separately addressed in the text below.

Support for minimum length: The evaluators shall check to ensure that the TSS describes the allowable ranges for password/passphrase lengths, and that at least 64 characters or a length defined by the platform may be specified by the user.

Support for character set: The evaluator shall check to ensure that the TSS describes the allowable character set and that it contains the characters listed in the SFR.

Support for PBKDF: The evaluator shall examine the TSS to ensure that the formation of all KEKs or FEKs (as decided in the FCS_CKM_EXT.3 selection) is described and that the key sizes match that described by the ST author.

The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_KYC_EXT.1.

For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1(4)). If any manipulation of the key is performed in forming the submask that will be used to form the FEK or KEK, that process shall be described in the TSS.

No explicit testing of the formation of the submask from the input password is required.

FCS_CKM_EXT.6.2: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1 (from the [AppPP]).

KMD

None.

Guidance
Support for minimum length: The evaluators shall check the Operational Guidance to determine that there are instructions on how to generate large passwords/passphrases, and instructions on how to configure the password/passphrase length to provide entropy commensurate with the keys that the authorization factor is protecting.

Tests
Support for Password/Passphrase characteristics: In addition to the analysis above, the evaluator shall also perform the following tests on a TOE configured according to the Operational Guidance:

- *Test FCS_CKM_EXT.6.1:1: Ensure that the TOE supports password/passphrase lengths as defined in the SFR assignments.*
- *Test FCS_CKM_EXT.6.1:2: Ensure that the TOE does not accept more than the maximum number of characters specified in FCS_CKM_EXT.6.1.*
- *Test FCS_CKM_EXT.6.1:3: Ensure that the TOE does not accept less than the minimum number of characters specified in FCS_CKM_EXT.6.4. If the minimum length is settable by the administrator, the evaluator determines the minimum length or lengths to test.*
- *Test FCS_CKM_EXT.6.1:4: Ensure that the TOE supports passwords consisting of all characters listed in FCS_CKM_EXT.6.2.*

Conditioning: *No explicit testing of the formation of the authorization factor from the input password/passphrase is required.*

FCS_COP.1/KE Cryptographic operation (Key Encryption)

The inclusion of this selection-based component depends upon selection in FCS_KYC_EXT.1.1.

FCS_COP.1.1/KE

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 [*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.

Evaluation Activities

FCS_COP.1.1/KE
TSS

Requirement met by the platform

If the platform provides the FEK encryption/decryption, then the evaluator shall examine the TSS to verify that it describes how the FEK encryption/decryption is invoked.

Requirement met by the TOE

The evaluator shall verify the TSS includes a description of the key size used for encryption and the mode used for the key encryption

KMD

None.

Guidance
None.

Tests
The evaluation activity tests specified for AES in CBC mode in FCS_COP.1.1(1) in the underlying [AppPP] shall be performed.

FCS_COP.1/KT Cryptographic operation (Key Transport)

The inclusion of this selection-based component depends upon selection in FCS_KYC_EXT.1.1.

FCS_COP.1.1/KT

The TSF shall perform [*key transport*] in accordance with a specified cryptographic algorithm [**selection**:

- *CNSA 2.0 Compliant Algorithms:*
  - *[Hybrid Key-Transport Method] that meets the following: [Section 9.3 of NIST SP 800-56B, Revision 2]* ***using a key establishment algorithm [ML-KEM] as specified in FCS_CKM.2 (from the Base-PP) and a key wrapping algorithm as specified in FCS_COP.1/KW***
- *CNSA 1.0 Compliant Algorithms:*
  - *[KTS-OAEP] using cryptographic algorithm [RSA] with cryptographic key sizes [**selection**: 3072, 4096] bits that meet the following: [NIST SP 800-56B, Revision 2]*

] .

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.
When describing the hybrid key-transport methods, Section 9.3 of NIST SP 800-56B only mentions using the key establishment schemes contained in SP800-56B to establish a shared symmetric key encryption key (KEK) for the transport; however any approved symmetric key establishment method could easily be substituted. This PP-Module allows the use of ML-KEM key establishment schemes with the hybrid key-transport methods described in SP800-56B to provide a CNSA2-compliant key-transport method.

Future versions of this PP-Module may support the use of a key-encapsulation mechanism (KEM) for key transport as defined in NIST SP 800-227 (in draft as of the publication of this PP-Module), sections 5.1 and 6.2.1.

Evaluation Activities

FCS_COP.1.1/KT

TSS

The evaluator shall verify the TSS provides a high level description of the RSA scheme and the cryptographic key size that is being used, and that the asymmetric algorithm being used for key transport is RSA. If more than one scheme/key size are allowed, then the evaluator shall make sure and test all combinations of scheme and key size. There may be more than one key size to specify - an RSA modulus size (and/or encryption exponent size), an AES key size, hash sizes, MAC key/MAC tag size.

The evaluator shall verify that the TSS identifies the hash function, the mask generating function, the random bit generator, the encryption primitive and decryption primitive.

KMD

None.

Guidance

None.

Tests

For each supported key transport schema, the evaluator shall initiate at least 25 sessions that require key transport with an independently developed remote instance of a key transport entity, using known RSA key-pairs. The evaluator shall observe traffic passed from the sender-side and to the receiver-side of the TOE, and shall perform the following tests, specific to which key transport scheme was employed.

- *Test FCS_COP.1.1/KT:1: The evaluator shall inspect each cipher text, C, produced by the RSA-OAEP encryption operation of the TOE and make sure it is the correct length, either 256 or 384 bytes depending on RSA key size. The evaluator shall also feed into the TOE's RSA-OEAP decryption operation some cipher texts that are the wrong length and verify that the erroneous input is detected and that the decryption operation exits with an error code.*
- *Test FCS_COP.1.1/KT:2: The evaluator shall convert each cipher text, C, produced by the RSA-OAEP encryption operation of the TOE to the correct cipher text integer, c, and use the decryption primitive to compute em = RSADP((n,d),c) and convert em to the encoded message EM. The evaluator shall then check that the first byte of EM is 0x00. The evaluator shall also feed into the TOE's RSA-OEAP decryption operation some cipher texts where the first byte of EM was set to a value other than 0x00, and verify that the erroneous input is detected and that the decryption operation exits with an error code.*
- *Test FCS_COP.1.1/KT:3: The evaluator shall decrypt each cipher text, C, produced by the RSA-OAEP encryption operation of the TOE using RSADP, and perform the OAEP decoding operation (described in NIST SP 800-56B section 7.2.2.4) to recover HA' \|\| X. For each HA', the evaluator shall take the corresponding A and the specified hash algorithm and verify that HA' = Hash(A). The evaluator shall also force the TOE to perform some RSA-OAEP decryption where the A value is passed incorrectly, and the evaluator shall verify that an error is detected.*
- *Test FCS_COP.1.1/KT:4: The evaluator shall check the format of the 'X' string recovered in OAEP.Test.3 to ensure that the format is of the form PS \|\| 01 \|\| K, where PS consists of zero or more consecutive 0x00 bytes and K is the transported keying material. The evaluator shall also feed into the TOE's RSA-OEAP decryption operation some cipher texts for which the resulting 'X' strings do not have the correct format (i.e., the leftmost non-zero byte is not 0x01). These incorrectly formatted 'X' variables shall be detected by the RSA-OEAP decrypt function.*
- *Test FCS_COP.1.1/KT:5: The evaluator shall trigger all detectable decryption errors and validate that the returned error codes are the same and that no information is given back to the sender on which type of error occurred. The evaluator shall also validate that no intermediate results from the TOE's receiver-side operations are revealed to the sender.*

FCS_COP.1/KW Cryptographic operation (Key Wrapping)

The inclusion of this selection-based component depends upon selection in FCS_KYC_EXT.1.1.

FCS_COP.1.1/KW

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 [*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.

Evaluation Activities

FCS_COP.1.1/KW
TSS
Conditional: If use platform provided functionality was selected, then the evaluator shall examine the TSS to verify that it describes how the FEK encryption/decryption is invoked.

Conditional: If implement functionality was selected, The evaluator shall check that the TSS includes a description of encryption function(s) used for key wrapping. The evaluator should check that this description of the selected encryption function includes the key sizes and modes of operations as specified in the selection above. The evaluator shall check that the TSS describes the means by which the TOE satisfies constraints on algorithm parameters included in the selections made for 'cryptographic algorithm' and 'list of standards'.

The evaluator shall verify the TSS includes a description of the key wrap function(s) and shall verify the key wrap uses an approved key wrap algorithm according to the appropriate specification.

KMD

The evaluator shall review the KMD to ensure that all keys are wrapped using the approved method and a description of when the key wrapping occurs.

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

Tests
The evaluation activity tests specified for AES in GCM mode in the underlying [AppPP] shall be performed in the case that "GCM" is selected in the requirement.

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator will test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- *128 and 256 bit key encryption keys (KEKs)*
- *Three plaintext lengths. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).*

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator will use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator will test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

The evaluator will test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext

lengths:

- *One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).*
- *One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).*

The evaluator will test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

AES-CCM Tests

It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths: Keys: All supported and selected key sizes (e.g., 128, 256 bits). Associated Data: Two or three values for associated data length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported. Payload: Two values for payload length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported payload lengths. Nonces: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13) in bytes. Tag: All supported tag lengths (4, 6, 8, 10, 12, 14, 16) in bytes.

The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

Variable Associated Data Test
For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Payload Text
For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Nonce Test
For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Tag Test
For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Decryption-Verification Process Test
To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

FCS_KDF_EXT.1 Cryptographic Key Derivation Function

The inclusion of this selection-based component depends upon selection in 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 **/KeyedHash** (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.

Evaluation Activities

FCS_KDF_EXT.1.1
TSS
The evaluator shall verify the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 and SP 800-132.

KMD

None.

Guidance
None.

Tests
None.

FCS_SMC_EXT.1 Submask Combining

The inclusion of this selection-based component depends upon selection in FCS_KYC_EXT.1.1.

FCS_SMC_EXT.1.1

The TSF shall combine submasks using the following method [***[selection**: exclusive OR (XOR), SHA-384, SHA-512, 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.

Evaluation Activities

FCS_SMC_EXT.1.1
TSS
If keys are XORed together to form an intermediate key, the TSS section shall identify how this is performed (e.g., if there are ordering requirements, checks performed, etc.). The evaluator shall also confirm that the TSS describes how the length of the output produced is at least the same as that of the FEK.

KMD

None.

Guidance
None.

Tests
None.

Appendix C - Extended Component Definitions

This appendix contains the definitions for all extended requirements specified in the PP-Module.

C.1 Extended Components Table

All extended components specified in the PP-Module are listed in this table:

Table 6: Extended Component Definitions

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

C.2 Extended Component Definitions

C.2.1 Cryptographic Support (FCS)

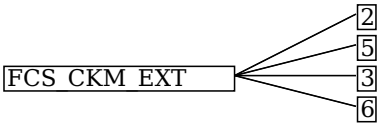
This PP-Module defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.1 FCS_CKM_EXT Cryptographic Key Management

Family Behavior

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.

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

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

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.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 and with entropy corresponding to the security strength of AES key sizes [**assignment**: supported key size]*
  - *using key generation methods compliant with NIST SP 800-133r2,*
  - *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.

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 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.

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.

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 and with entropy corresponding to the security strength of AES key sizes of 256 bits*
  - *derived from a password/passphrase that is conditioned as defined in FCS_CKM_EXT.6*

  ]

] .

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-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 [**assignment**: *output key size*] that meet the following: [**assignment**: *applicable standard*] .

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

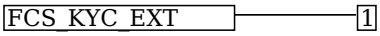
The TSF shall generate all salts using an RBG that meets FCS_RBG_EXT.1 and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.6.3.

C.2.1.2 FCS_KYC_EXT Key Chaining and Key Storage

Family Behavior

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 [**assignment**: key type]*
- *[**assignment**: key type] originating from [**assignment**: origin of key] to [**assignment**: end point of key chain] using the following method(s): [**assignment**: list of supported key protection methods] while maintaining an effective strength of commensurate with the strength of the FEK*

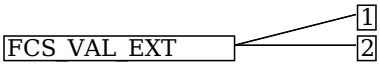
] and [**selection**: *no supplemental key chains, other supplemental key chains that protect a key or keys in the primary key chain using the following method(s): [**assignment**: list of supported key protection methods]*] .

C.2.1.3 FCS_VAL_EXT Validation

Family Behavior

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.

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.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 [**assignment**: *subject requiring validation*] 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*
  - *hash the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1 and compare it to a stored hash*
  - *decrypt a known value using the [**selection**: submask, intermediate key, FEK] as specified in FCS_COP.1 and compare it against a stored known value*

  ]

] .

FCS_VAL_EXT.1.2

The TSF shall require validation of the [**assignment**: *subject requiring validation*] prior to [**assignment**: *action requiring validation*].

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 [**assignment**: type of key(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*

] .

C.2.1.4 FCS_COP_EXT Cryptographic Operation

Family Behavior

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

Component Leveling

FCS_COP_EXT ——— 1

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

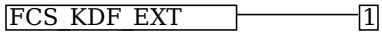
The FAK shall be protected in the same manner as the FEK, in accordance with FCS_COP.1.

C.2.1.5 FCS_KDF_EXT Cryptographic Key Derivation Function

Family Behavior

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_SMC_EXT.1 Submask Combining

FCS_KDF_EXT.1.1

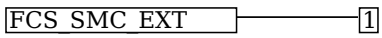
The TSF shall [**assignment**: *intermediate key derivation function*] to derive an intermediate key, as defined in [**assignment**: *applicable key standard*] using the keyed-hash functions specified in FCS_COP.1, such that the output is at least of equivalent security strength (in number of bits) to the [**assignment**: *derived key*] .

C.2.1.6 FCS_SMC_EXT Submask Combining

Family Behavior

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: No dependencies.

FCS_SMC_EXT.1.1

The TSF shall combine submasks using the following method [**assignment**: *combination method*] to

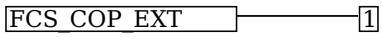
generate an intermediate key.

C.2.1.7 FCS_COP_EXT Cryptographic Operation

Family Behavior

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

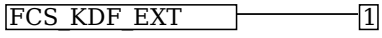
The FAK shall be protected in the same manner as the FEK, in accordance with FCS_COP.1.

C.2.1.8 FCS_KDF_EXT Cryptographic Key Derivation Function

Family Behavior

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_SMC_EXT.1 Submask Combining

FCS_KDF_EXT.1.1

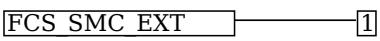
The TSF shall [**assignment**: *intermediate key derivation function*] to derive an intermediate key, as defined in [**assignment**: *applicable key standard*] using the keyed-hash functions specified in FCS_COP.1, such that the output is at least of equivalent security strength (in number of bits) to the [**assignment**: *derived key*] .

C.2.1.9 FCS_SMC_EXT Submask Combining

Family Behavior

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: No dependencies.

FCS_SMC_EXT.1.1

The TSF shall combine submasks using the following method [**assignment**: *combination method*] to generate an intermediate key.

C.2.2 Identification and Authentication (FIA)

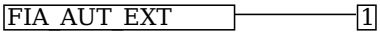
This PP-Module defines the following extended components as part of the FIA class originally defined by CC Part 2:

C.2.2.1 FIA_AUT_EXT Authorization

Family Behavior

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), generated by the platform] protected using asymmetric keys as defined in FCS_CKM.1.1/AK (from*

*[AppPP]) 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), platform]*

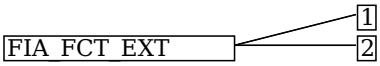
] .

C.2.2.2 FIA_FCT_EXT Authorization Factors

Family Behavior

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.

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.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.

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_CKM.2 Cryptographic Key Distribution FCS_COP.1 Cryptographic Operation

FIA_FCT_EXT.2.1

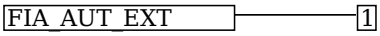
The TSF shall support [**selection**: *Authorized User Key sharing via key transport as specified in FCS_COP.1, Authorized User Key sharing via key wrapping as specified in FCS_COP.1, Distribution of a shared key from [**assignment**: external IT entity] as specified in FCS_CKM.2*] .

C.2.2.3 FIA_AUT_EXT Authorization

Family Behavior

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), generated by the platform] protected using asymmetric keys as defined in FCS_CKM.1.1/AK (from [AppPP]) 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), platform]*

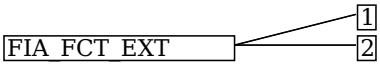
] .

C.2.2.4 FIA_FCT_EXT Authorization Factors

Family Behavior

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.

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.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.

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_CKM.2 Cryptographic Key Distribution 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, Authorized User Key sharing via key wrapping as specified in FCS_COP.1, Distribution of a shared key from [**assignment**: external IT entity] as specified in FCS_CKM.2*] .

C.2.3 Protection of the TSF (FPT)

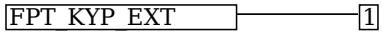
This PP-Module defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.1 FPT_KYP_EXT Protection of Key and Key Material

Family Behavior

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*
  - *encrypted, as specified in FCS_COP.1*
  - *the plaintext key is stored in the underlying platform's keystore as specified by FCS_STO_EXT.1.1*
  - *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*
- *encrypted as specified in FCS_COP.1*
- *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 or wrap a key as specified in FCS_COP.1 that is already encrypted as specified in FCS_COP.1 or wrapped as specified in FCS_COP.1*

]

] .

C.2.4 User Data Protection (FDP)

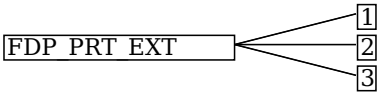
This PP-Module defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.4.1 FDP_PRT_EXT Protection of Selected User Data

Family Behavior

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.

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.

FDP_PRT_EXT.3, Protection of Third-Party Data, requires the TOE to destroy temporary files that may be created during the encryption or decryption process to prevent the inadvertent disclosure of sensitive data.

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.6 Timing and event of 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.

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.6.

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.6 Timing and event of 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.6 upon completion of the decryption/encryption operation.

Management: FDP_PRT_EXT.3

There are no specific management functions identified.

Audit: FDP_PRT_EXT.3

There are no auditable events foreseen.

FDP_PRT_EXT.3 Protection of Third-Party Data

Hierarchical to: No other components.

Dependencies to: FDP_PRT_EXT.1 Protection of Selected User Data

FDP_PRT_EXT.3.1

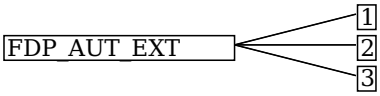
The TSF shall ensure that all temporary files created by [**selection**: *all applications*, ***[assignment**: subset of applications that can integrate with the FE]*] when decrypting/encrypting the user-selected file (or set of files) are removed or encrypted upon completion of the decryption/encryption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

C.2.4.2 FDP_AUT_EXT User Data Authentication

Family Behavior

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.

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.

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.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*].

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.

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 [**selection**: *a RBG that meets FCS_RBG_EXT.1*, *key generation methods compliant with NIST SP 800-133r2*] .

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 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 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 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

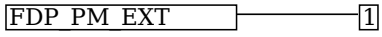
During verification, the TSF shall verify the signature is at the end of the ciphertext file.

C.2.4.3 FDP_PM_EXT Protection of Data in Power Managed States

Family Behavior

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

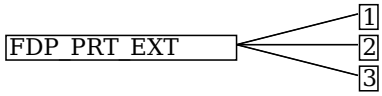
a protected state as defined by FDP_PM_EXT.1.1.

C.2.4.4 FDP_PRT_EXT Protection of Selected User Data

Family Behavior

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.

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.

FDP_PRT_EXT.3, Protection of Third-Party Data, requires the TOE to destroy temporary files that may be created during the encryption or decryption process to prevent the inadvertent disclosure of sensitive data.

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.6 Timing and event of 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.

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.6.

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.6 Timing and event of 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.6 upon completion of the decryption/encryption operation.

Management: FDP_PRT_EXT.3

There are no specific management functions identified.

Audit: FDP_PRT_EXT.3

There are no auditable events foreseen.

FDP_PRT_EXT.3 Protection of Third-Party Data

Hierarchical to: No other components.

Dependencies to: FDP_PRT_EXT.1 Protection of Selected User Data

FDP_PRT_EXT.3.1

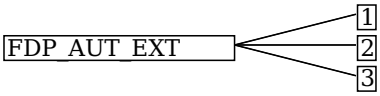
The TSF shall ensure that all temporary files created by [**selection**: *all applications*, *[**assignment**: subset of applications that can integrate with the FE]*] when decrypting/encrypting the user-selected file (or set of files) are removed or encrypted upon completion of the decryption/encryption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

C.2.4.5 FDP_AUT_EXT User Data Authentication

Family Behavior

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.

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.

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.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*].

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.

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 [**selection**: *a RBG that meets FCS_RBG_EXT.1, key generation methods compliant with NIST SP 800-133r2*] .

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 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 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 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

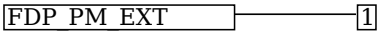
During verification, the TSF shall verify the signature is at the end of the ciphertext file.

C.2.4.6 FDP_PM_EXT Protection of Data in Power Managed States

Family Behavior

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.

Appendix D - Appendix - 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 E - Acronyms

Table 7: Acronyms

Acronym	Meaning
AF	Authorization factor
Base-PP	Base Protection Profile
CC	Common Criteria
CEM	Common Evaluation Methodology
cPP	Collaborative Protection Profile
DRBG	Deterministic Random Bit Generator
EP	Extended Package
FAK	File Authentication Key
FEK	File Encryption Key
FP	Functional Package
KEK	Key Encryption Key
OE	Operational Environment
PP	Protection Profile
PP-Configuration	Protection Profile Configuration
PP-Module	Protection Profile Module
RBG	Random Bit Generator
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification

Appendix F - Bibliography

Table 8: Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and general model, CCMB-2022-11-001, CC:2022, Revision 1, November 2022. • Part 2: Security functional requirements, CCMB-2022-11-002, CC:2022, Revision 1, November 2022. • Part 3: Security assurance requirements, CCMB-2022-11-003, CC:2022, Revision 1, November 2022. • Part 4: Framework for the specification of evaluation methods and activities, CCMB-2022-11-004, CC:2022, Revision 1, November 2022. • Part 5: Pre-defined packages of security requirements, CCMB-2022-11-005, CC:2022, Revision 1, November 2022.
[CEM]	Common Methodology for Information Technology Security Evaluation - • Evaluation methodology, CCMB-2022-11-006, CC:2022, Revision 1, November 2022.
[AppPP]	Protection Profile for Application Software, Version 2.0, June 16, 2025
[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
[FIPS186-5]	Federal Information Processing Standard Publication (FIPS-PUB) 186-5, Digital Signature Standard (DSS), National Institute of Standards and Technology, February 2023
[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
[NIST800-132]	NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation, December 2010
[NIST800-38A]	NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques, 2001 Edition
[NIST800-38F]	NIST Special Publication 800-38F,Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012
[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