Application Software Protection Profile (ASPP) Extended Package: File Encryption: Mitigating the Risk of Disclosure of Sensitive Data on a System

10 November 2014

Version 1.0

Table of Contents

Revision History

Version	Date	Description
0.1	25 July 2014	Initial release
1.0	10 November 2014	Public release, incorporating comments received

1 Introduction

This Extended Package (EP) describes security requirements for an encryption product that is configurable for the data it encrypts and is intended to provide a minimal, baseline set of requirements that are targeted at mitigating well defined and described threats. However, this EP is not complete in itself, but rather extends the Protection Profile for Application Software (AS PP). This introduction will describe the features of a compliant Target of Evaluation, and will also discuss how this EP is to be used in conjunction with the AS PP.

1.1 Conformance Claims

1 The *Application Software Protection Profile (AS PP)* defines the baseline Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) for application software products. This EP serves to extend the AS PP baseline with additional SFRs and associated 'Assurance Activities' specific to File Encryption products. Assurance Activities are the actions that the evaluator performs in order to determine a TOE's compliance to the SFRs.

2 This EP conforms to *Common Criteria for Information Technology Security Evaluation*, Version 3.1, Revision 4. It is CC Part 2 extended and CC Part 3 conformant.

3 In order to be conformant to this EP, a TOE must demonstrate Exact Compliance. Exact Compliance, a subset of Strict Compliance as defined by the CC, is defined as the ST containing all of the requirements in section 4 of the AS PP, and potentially requirements from Appendix C of the AS PP. While iteration is allowed, no additional requirements (from the CC parts 2 or 3) are allowed to be included in the ST. Further, no requirements in section 4 of the AS PP are allowed to be omitted.

1.2 How to Use This Extended Package

4 As an EP of the AS PP, it is expected that the content of both this EP and the AS PP be appropriately combined in the context of each product-specific Security Target. This EP has been specifically defined such that there should be no difficulty or ambiguity in so doing. An ST must identify the applicable versions of the AS PP (see http://www.niap-ccevs.org/pp/ for the current version) and this EP in its conformance claims. When requirements are referenced from the AS PP, a short notation is included.

1.3 Compliant Targets of Evaluation

5 This EP specifically addresses encryption of a set of data. This EP 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 EP 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.).

6 There are two use cases for this EP. First, the traditional ability to encrypt files and power down the machine and know the data is securely protected. Second, the ability to encrypt a file on a machine and then send the encrypted file securely using a non-encrypted data in transit method.

1.3.1 Usage and Major Security Features of the Target of Evaluation (TOE)

7 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 EP 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. For the purposes of this EP, "set of files" describes implementations that use one encryption key to encrypt more than one file.

8 The foremost security objective of file encryption is to force an adversary to perform a cryptographic exhaust against a prohibitively large key space. Note that this can be achieved only if the authorized user of the file encryption product follows good security practices and does not store an authorization factor in the clear.

9 Technology is changing at a rapid rate and the definition of mobile devices and traditional laptop/PC devices is quickly merging. Requirements will diverge slightly for Mobile vs Laptop/PC and the Assurance Activities will describe any differences. For this EP, the following table will be used to explain the general principles for several key concepts, including power state and memory management.

Topic	Mobile	Laptop/PC
Memory Management - when the TOE is running	The TOE must initiate the request to clear the cryptographic keys and plaintext data, but the TOE Platform will handle the actual instruction through memory management queue. (AS PP or MDF PP) (The assurance is dependent on the TOE Platform to perform the action of clearing the plaintext data and cryptographic keys.)	The TOE is responsible for handling the clearing of cryptographic keys and plaintext in volatile memory by overwrite or zeroization. (Risk: Non-volatile memory (page files) may still contain the original plaintext data and keys. Reboot is required to ensure that this memory space has been wiped. This risk can be minimized by following good operational practices.)
Memory Management - when the TOE application cleanly closes	The TOE must initiate the request to clear the cryptographic keys and plaintext data, but the TOE Platform will handle the actual instruction through memory management queue. (AS PP or MDF PP) (The assurance is dependent on the TOE Platform to perform the action of clearing the plaintext data and cryptographic keys.)	The TOE's application memory space is gone and all volatile memory associated with the application no longer exists. All plaintext data and plaintext keys have been destroyed. (Risk: Non-volatile memory (page files) may still contain the original plaintext data and keys. Reboot is required to ensure that this memory space has been wiped. This risk can be minimized by following good operational practices.)
Memory Management - Lockscreen	If the TOE is running and a plaintext document is displayed on the screen the user's data is not protected.	Lockscreen/Standby/Hibernate happen invisibly to the TOE, therefore if a plaintext document is displayed on the desktop during one of these events, the user's data is assumed to be not protected.

Memory Management - unintentional shutdown	If the TOE is running and a plaintext document is displayed on the screen and an unintentional shutdown occurs, there is a chance that temporary files may still exist but all volatile memory will be destroyed over time (pending cold boot attack).	If the TOE is running and a plaintext document is displayed on the desktop and an unintentional shutdown occurs, there is a chance that temporary files may still exist but all volatile memory will be destroyed.

When the user is finished working with sensitive data from an encrypted file, the file encryption product must re-encrypt this data and is responsible for removing all keying materials and any plaintext data from the encryption product's volatile memory or any temporary files (non-volatile memory) it creates during the decryption/encryption process. This functionality can be met by a combination of the TOE and the Operational Environment.

10 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 (e.g. smartcard with a RSA or ECC key pair) to protect the randomly generated FEK. The external token will later be referred to as an "external entity" in this EP, and contains "external authorization factors."

From a terminology standpoint, a KEK is either a symmetric key (as in case 2) or an asymmetric key pair (as in case 3), 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 assurance activities below.

11 Secure design and use of a file encryption product must be addressed on multiple levels. From a software-design standpoint, the product must employ strong cryptography, robust error handling, and ensure complete deletion of all keying materials and plaintext data stored in its volatile memory and/or non-volatile memory (platform dependent). From a system standpoint, the product may need to be configured to interact with other hardware or software (smart cards, cryptographic libraries, etc.) that are required on the machine. Finally, from a user standpoint, the product must be simple enough to operate to prevent the user from simply not encrypting their files, and must include instructions to promote secure operational usage. If any of these perspectives are ignored, then secure use of the file encryption product is compromised. Therefore, this EP addresses both the cryptography and implementation requirements necessary to design a secure product, as well as the user and configuration guidance necessary to securely operate the file encryption software (for example, how to disable hibernation).

12 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.

13 The vendor is required to provide configuration guidance (AGD_PRE, AGD_OPE) to correctly install and administer the TOE for every operational environment supported (for example, for every OS supported by the product).

14 Some products support the use of a recovery key that can be used to recover the encrypted data if the FEK is lost. This functionality must be configurable so it can be turned off and cannot diminish the overall strength of the FEK.

1.3.1.1 Authorization

15 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.

16 All compliant TOEs must provide (or support, in the case of an external authorization factor) at least one of the following authorization factor options and be able to support the configuration of:

- A password/passphrase that supports at least a 64 character space,
- An external token (e.g. smartcard) or software capability (on the host, for instance) containing a software certificate for the user with RSA or ECC key pairs may be used.
  The implementation of this capability is largely outside the TOE boundary (depending on the particulars of the implementation); however, the TOE must interface with the external entity; be able to specify the use of RSA or ECC CDH to protect the FEK (even if this specification is implicit rather than explicit); and be able to provide information to the external entity in order to unlock the private key (if required by the external entity). The computations involving the private key are performed by the cryptographic capability of the external entity.

17 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.

18 A password/passphrase authentication factor with low entropy reduces the overall algorithm strength. While this EP 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. Once the password/passphrase is entered by the user, it is conditioned by the TOE prior to being provided as an encryption key. 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. The requirements in Appendix C for the selected authorization factors should be included in the ST.

1.3.1.2 Encryption

19 One or more authorization factors must be established or entered before data can be encrypted or decrypted, respectively. Entry of an incorrect authorization factor should not result in the user seeing an improperly decrypted file. Entry of a correct or incorrect authorization factor should not

aid an attacker in guessing the KEK or FEK.

20 If the cryptography used to generate, handle, and protect keys or authorization factors is sufficiently robust and if the implementation has no critical mistakes, the only option for an adversary who obtains the encrypted information without the authorization factors or KEK must be to exhaust the encryption key space of the KEK or FEK for data decryption. Note that if passwords are used, a password might offer less strength than exhausting over the potential number of keys for the data encryption algorithm (AES). Furthermore, if the password is the only authorization factor unknown to the adversary, then the key space is the minimum of the work needed to exhaust the KEK or FEK or to exhaust the number of possible passwords. As a consequence, the next generation of this EP may require support for more robust authorization factors.

21 If external authorization factors are used, the external device generally requires some factor (such as a PIN or password) to unlock the private key. In these cases, the PIN or password may have a similar smaller exhaust space than the KEK or FEK, and this should be taken into account when choosing a product that is conformant to this EP.

22 The data being secured by the file encryption product must be encrypted using a FEK. If the FEK is protected by the KEK, the FEK will be generated using a Deterministic Random Bit Generator (DRBG) that meets the requirements of FCS_RBG_EXT.1 (from the AS PP). The DRBG comprises an entropy source and the DRBG algorithm. A properly seeded DRBG provides enough entropy to be of equal or greater value than the exhaust space of the KEK.

1.3.1.3 Administration

23 The base requirements of the TOE do not require the TOE to maintain an administrative role (the notion of an administrator of the TOE is that there exists a subset of the users of the TOE that have greater "trust" than the general user population and who have specific responsibilities). Typically, administrators possess privilege to invoke functionality on the TOE that is not available to general users. For file encryption products, however, once the product is installed there should be little need for administrative involvement.

1.3.1.4 Authorized Users

24 Authorized users are expected to adhere to the user guidance to minimize the risk of compromised data. Authorization is determined by possessing and providing the TOE the correct authorization factor(s) to enable the file encryption product's functionality. It is the responsibility of the authorized users of the host machine to secure and protect the host machine and authorization factors for the TOE while it is officially in their possession. Authorized users will not leave/store unprotected authorization factors (e.g., passwords, passphrases) in written or digital form on or around the host machine. The user will be provided appropriate guidance to maintain a secure TOE.

1.3.1.5 Data Authentication (optional)

25 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 EP 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.

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

27 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(2) (from the AS PP) and an approved secure hashing function in accordance with FCS_COP.1(3) (from the AS PP). The primary requirement addressing data authentication via asymmetric signing is FDP_AUT_EXT.3.

1.3.2 The TOE and Its Supporting Environment

28 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 an Operational Environment with the necessary features and to provide instructions for how to configure it correctly and securely.

29 The EP contains requirements (Section 4) that must be met by either the TOE or the platform on which it operates. A “platform” is defined as a separate entity whose functions may be used by the TOE, but is not part of the TOE. A third-party library used by the TOE is not considered part of the TOE’s “platform”, but (for instance) cryptographic functionality that is built into an Operating System on which the TOE executes can be considered part of the platform. Likewise, an external entity (such as a smartcard) that performs cryptographic operations with respect to the FEK would also be considered a part of the “TOE Platform”.

30 Requirements that can be satisfied by either the TOE or the platform are identified in Section 4.3. 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 assurance activities.

31 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:

32 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

33 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.
- Instructions on how to configure the operational environment such that it is compliant with the Mobile Device Fundamentals Protection Profile.

34 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 B.

35 Authorized users of the TOE are those users possessing valid authorization factors for the TOE. While some of these functions specified in the EP 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.

2 Security Problem Definition

36 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.

2.1 Threats

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

38 For this EP, 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 EP. For example, there are no requirements in this EP 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.

39 Note that this EP does not repeat the threats identified in the AS PP, though they all apply given the conformance and hence dependence of this EP on the AS PP. Note also that while the AS PP contains only threats to the ability of the TOE to provide its security functions, this EP focuses on threats to resources in the operational environment. Together the threats of the AS PP and those defined in this EP define the comprehensive set of security threats addressed by a file encryption TOE.

40 Compromise of Keying MaterialAttacks against the encryption product could take several forms; for example, if there is a weakness in the random number generation mixing algorithm or the data sources used in random number generation are guessable, then the output may be guessable as well. If an attacker can guess the output of the pseudorandom number generator (PRNG) at the time an encryption key is made, then the output may be used to recreate the keying material and decrypt the protected files. As the encryption program runs, it will store a variety of information in memory. Some of this information, such as random bit generation (RBG) inputs, RBG output, copies of the plaintext file, and other keying material, could be very valuable to an attacker who wishes to decrypt an encrypted file. If the encryption product does not wipe these memory spaces appropriately, an attacker may be able to recreate the encryption key and access encrypted files.

(T.KEYING_MATERIAL_COMPROMISE**)**

41 Brute Force AttackThe protection of the data involves encrypting said data assuming an attacker may have significant computing resources at their disposal. Several ciphers have already been broken through brute-force attacks because the length of the keys used in those ciphers was too short to provide protection against a concerted computing effort to discover those keys. Because protection of the data may rely on a chaining of keys and encryption mechanisms, there are many opportunities for brute force attacks against each potential key in the chain, such that the weakest link in the chain of factors/keys will determine the overall strength against a brute force attack.

42 (T.KEYSPACE_EXHAUST)

43 Plaintext CompromiseUnlike full disk encryption, selectable encryption products also need to protect against data leaks to other applications on the machine. Many file creators and editors store temporary files as the user is working on a file, and restore files if the machine experiences an interrupt while a file is open. Any of these files, if not properly protected or deleted, could leak information about a protected file to an attacker. Other applications might also access volatile or non-volatile memory released by the file encryption product, and the software used to create files prior to encryption may retain information about the file even after it has been encrypted. As the user creates and saves a new document, the plaintext will be stored on the machine's hard drive. An attacker could then search for the plaintext of the sensitive, encrypted information. An attacker may not even have to access the encrypted file for the protected information to be compromised. When the user wishes to encrypt the document, this plaintext file should be replaced with the new encrypted version. For non-mobile devices, it is expected that if the volatile and/or non-volatile memory space where the plaintext file was stored is merely released back to the machine without being first wiped clean of the data that was stored there, then the information the user wishes to protect will still be accessible. While protection of the encryption algorithm itself is vital, memory must also be properly managed by the file encryption product or the TOE platform in order for security to remain intact. For mobile devices, it is assumed that the File Encryption product will not be responsible for providing memory management cleanup and the environment's platform has met the Mobile Device Fundamentals Protection Profile.

44 Additionally, some encryption products offer to create backup files. These files are meant to be used in the event an encrypted file becomes corrupted and incapable of being decrypted. Each backup file is a valuable resource to protect information that the user cannot afford to lose; however, it also may provide another route for an attacker to access the encrypted information. If the backup file is insufficiently protected, then the attacker may choose to attempt to break into it, rather than the copy of the encrypted file that the user would typically access.

(T.PLAINTEXT_COMPROMISE)

45 TSF FailureSecurity mechanisms of the TOE generally build up from a primitive set of mechanisms (e.g., memory management, privileged modes of process execution) to more complex sets of mechanisms. Failure of the primitive mechanisms could lead to a compromise in more complex mechanisms, resulting in a compromise of the TSF.

46 (T.TSF_FAILURE)

47 Unauthorized Data AccessThe central functionality of the TOE is the protection of resources under its control through encryption. In a shared resource environment, users on a system may have access to administrative-level tools that are capable of over-riding a system’s access control protections. Further, if the system were to be lost or the system’s storage device stolen, the attacker could then look directly at the storage device using low-level forensic tools in an attempt to access data for which they are not authorized. However, the need to protect the data in these

scenarios should not interfere with the data-owner's (or another user that has been granted access to those data) ability to read or manipulate the data.

(T.UNAUTHORIZED_DATA_ACCESS)

48 Flawed Authentication Factor VerificationWhen a user enters an authorization factor, the TOE is required to ensure that the authorization factor is valid prior to providing any data to the user; the purpose of verification is to ensure the FEK is correctly derived. If the data is decrypted with an incorrectly derived FEK (the FEK is conditioned from the password/passphrase or is decrypted by the KEK), then unpredictable data will be provided to the user. If verification is not performed in a secure manner, keying material or user data may be exposed or weakened.

(T.UNSAFE_AUTHFACTOR_VERIFICATION)

49 Data Spoofing (optional)For certain modes of encryption, it is possible for a malicious person to modify ciphertext data to force unintended modification to the underlying plaintext data, without the user being notified. There are various failures that may occur on the part of the TOE, to include: failure to verify the integrity of the data prior to decryption, failure to provide integrity on the sensitive data, failure to use a cryptographic or secure hashing code and failure to differentiate the File Authentication Key (FAK) from the FEK; the FAK is any secret value used as input to a keyed hashing function or as part of an asymmetric authentication process.

(T.PLAINTEXT_DATA_SPOOFING)

2.2 Assumptions

50 The assumptions for the File Encryption are defined in Appendix A.1.2.

2.3 Organizational Security Policy

51 There are no additional OSPs for the File Encryption product.

3 Security Objectives

3.1 Security Objectives for the TOE

52 The Security Problem described in Section 2 will be addressed by a combination of cryptographic capabilities. Compliant TOEs will provide security functionality that addresses threats to the TOE and enforces policies that are imposed by law and regulation. The following subsections provide a description of the security objectives required to meet the threats/policies previously discussed. The description of these security objectives are in addition to that described in the AS PP.

53 Note: in each subsection below particular security objectives are identified (highlighted by O.) and they are matched with the associated security functional requirements (SFRs) that provide the mechanisms to satisfy the objectives.

54 The Security Objectives are the requirements for the Target of Evaluation (TOE) and for the Operational Environment derived from the threats in Section 2.

3.1.1 Protection of Key Material (O.KEY_MATERIAL_PROTECTION)

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

56 [FCS_CKM_EXT.4, FDP_PRT_EXT.1 (optional: FDP_PM_EXT.1)]

3.1.2 Encryption Using a Strong FEK and KEK (O.FEK_SECURITY)

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

58 [FCS_CKM_EXT.2, FMT_SMF.1, FCS_COP.1(1), FCS_COP.1(5), FCS_IV_EXT.1, FPT_FEK_EXT.1 (optional: FCS_COP.1(6)) (selectable: FCS_CKM.1, FCS_CKM_EXT.1, FCS_COP.1(4)]

3.1.3 Removal of Plaintext Data (O.WIPE_MEMORY)

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

60 [FDP_PRT_EXT.1 (optional: FDP_PRT_EXT.2)]

3.1.4 Protection of Data (O.AUTHORIZATION, O.PROTECT_DATA)

61 The TOE will encrypt data to protect the data from unauthorized access. Encrypting the file or set

of files will protect the user data even when low-level tools that bypass operating system protections such as discretionary and mandatory access controls are available to an attacker. Users that are authorized to access the data must provide *authorization factors* to the TOE in order for the data to be decrypted and provided to the user.

62

[FCS_CKM_EXT.1, FDP_PRT_EXT.1, FMT_SMF.1, FCS_COP.1(1) (optional: FDP_AUT_EXT.2, FDP_AUT_EXT.3) (selectable: FCS_KM_EXT.1, FCS_COP.1(4), FCS_CKM.1(A))]

3.1.5 Safe Authentication Factor Verification (O.SAFE_AUTHFACTOR_VERIFICATION)

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

64

65 [FIA_AUT_EXT.1 (selectable: FIA_FCT_EXT.1(1), FIA_FCT_EXT.1(2))]

66

3.1.6 Data Authentication (O.DATA_AUTHENTICATION)

67 For certain encryption modes, it is feasible to maliciously modify the ciphertext data to cause unintended modifications to plaintext data, without user notification. The TOE may provide a method for authenticating the sensitive data and using an approved data authentication method.

68

69 [FCS_CKM_EXT.4 (optional: FDP_AUT_EXT.1, FDP_AUT_EXT.2, FDP_AUT_EXT.3, , FCS_CKM_EXT.5)]

3.2 Security Objectives for the TOE’s Operational Environment

70 The objectives that are required to be met by the TOE’s operational environment are defined in Appendix A.

4 Security Functional Requirements

71 The Security Functional Requirements included in this section are derived from Part 2 of the *Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4* (the CC), with additional extended functional components. The Security Assurance Requirements included in this section are derived from Part 3 of the CC. Supplemental Guidance is provided in the form of Assurance Activities associated with the functional requirements in Sections 4.2 and 4.3, as well as with the Security Assurance Requirements themselves in Section 4.4.

4.1 Conventions

The CC defines operations on Security Functional Requirements: assignments, selections, assignments within selections and refinements. This document uses the following font conventions to identify the operations defined by the CC:

- Refinement operation (denoted by **bold** text): used to add details to a requirement, and thus further restricts a requirement.
- Selection (denoted by underlined text): used to select one or more options provided by the [CC] in stating a requirement.
- Assignment operation (denoted by *italicized* text): used to assign a specific value to an unspecified parameter, such as the length of a password.
- Assignment within a selection (denoted by *italicized*, underlined text): used to make an assignment within the context of a selection
- Iteration operation: are identified with a number inside parentheses (e.g. "(1)")

4.2 Security Functional Requirements for the File Encryption Application (TOE)

72 As indicated in Section 1.3.2, security functional requirements in the main body of the EP are divided into those that must be satisfied by the file encryption application (the TOE), and those that must be satisfied by either the TOE or the platform on which it runs. This section contains the requirements that must be met by the TOE.

4.2.1 Class: Cryptographic Support (FCS)

Cryptographic Key Management (FCS_CKM)

73 Conformant implementations will use a File Encryption Key (FEK) conditioned from a password/passphrase, or randomly generated and protected by a Key Encryption Key (KEK). A KEK is either produced by the TOE, or composed of a public/private key pair in hardware (e.g., a smartcard device) or software (service on the host) external to the TOE (the latter are referred to as “external entities” in this EP, and contain “external authorization factors”). If the FEK are randomly generated, then they must be generated by the TOE as specified in FCS_CKM_EXT.2, and support the use of a KEK as specified in this section. However, depending on the KEK(s) supported, either the TOE or the TOE platform (or some combination of the two) will implement the lower-level functionality, so those capabilities are specified in Section 4.2. If a FEK and KEK are used, authentication factors (especially the Password Authentication Factor) can be changed without having to re-encrypt all of the user data on the device.

FCS_CKM_EXT.2	**Cryptographic key generation (FEK)**
FCS_CKM_EXT.2.1	The TSF shall generate FEK cryptographic keys [selection: <u>using a Random Bit Generator as specified in FCS_RBG_EXT.1 *(from the AS PP)* and with entropy corresponding to the security strength of AES key sizes of [selection: 128 bit, 256 bit];</u> <u>conditioned from a password/passphrase as defined in FCS_CKM.1(A)</u> **]**
FCS_CKM_EXT.2.2	The TSF shall create a unique FEK for each file (or set of files) using the mechanism on the client as specified in FCS_CKM_EXT.2.1.
FCS_CKM_EXT.2.3	The FEKs must be generated by the TOE.

Application Note:

74 *For the first selection, the key generation capability of the TOE uses a RBG implemented on the TOE device (FCS_RBG_EXT.1 from the AS PP). Either 128-bit or 256-bit (or both) are allowed for the FEK; the ST author makes the selection appropriate for the device. For the second selection, the key is generated by the conditioning of a password/passphrase.*

75

76 *FCS_CKM_EXT.2.2 requires that each resource to be encrypted has a unique FEK, and that this FEK is generated by the TSF. If the encrypted resource is a set of files encrypted under one FEK, additional requirements on the initialization vectors and cipher modes must be adhered to in Section 4.2.*

Assurance Activities:

Activity	**Assurance Activity**
TSS	*FCS_CKM_EXT.2.1: The evaluator shall review the TSS to determine that a description covering how and when a FEK 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 FEK. The TSS is checked to ensure that the description of how the FEK are generated is consistent with the instructions in the AGD guidance, and any differences that arise from different platforms are taken into account. This assurance activity may be combined with activities for FCS_COP.1(5) and FCS_CKM_EXT.2.1.* *For the first selection, the evaluator shall review the TSS to determine that it*

	describes how the functionality described by FCS_RBG_EXT.1 (from the AS PP) is invoked to generate FEK. To the extent possible from the description of the RBG functionality in FCS_RBG_EXT.1 (from the AS PP), 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(1)). *For the second selection, password/passphrase-based factors, the examination of the TSS section is performed as part of FCS_CKM.1(A) assurance activities.* *FCS_CKM_EXT.2.2: The evaluator shall examine the TSS to determine that it describes how a FEK is created for a protected resource and associated with that resource; protection of the FEK itself is covered by FIA_AUT_EXT.1 and FCS_COP.1(5). The evaluator confirms that—per this description—the FEK is unique per resource (file or set of files) and that the FEK is created using the mechanisms specified in).* *FCS_CKM_EXT.2.3: The evaluator shall review the TSS to verify it details that the FEKs are generated on the client machine and are not generated on an external server.*
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 create a FEK exist—taking into account any differences that arise from different platforms—and are consistent with the description in the TSS.*
Tests	*None*

4.2.2 Class: User Data Protection (FDP)

77 This stipulates encryption, decryption and authentication of user-selected files or sets of files. There are several more requirements in Section 4.2 and Appendix B that also address plaintext data being successfully removed and sharing resources between users. There are requirements in Appendix C discussing specific methods for authenticating the data, as this is dependent on the choice of encryption mode.

Extended: Protection of Selected User Data (FDP_PRT_EXT)

FDP_PRT_EXT.1	**Extended: Protection of Selected User Data**
FDP_PRT_EXT.1.1	The TSF shall perform encryption and decryption of the user-selected file (or set of files) in accordance with FCS_COP.1(1).

Application Note:

78 *This is the primary requirement for encrypting and decrypting the protected resources (files and sets of files). Apart from the actual encryption and decryption of the resources, there are two other functions specified by this requirement.*

Assurance Activities:

Activity	**Assurance Activity**
TSS	*FDP_PRT_EXT.1.1: The evaluator shall examine the TSS to determine that it lists each type of resource that can be encrypted (e.g., file, directory) and what "encrypted" means in terms of the resource (e.g., "encrypting a directory" means that all of the files contained in the directory are encrypted, but the data in the directory itself (which are filenames and pointers to the files) are not encrypted). The evaluator shall also confirm that the TSS describes how each type of resource listed is encrypted and decrypted by the TOE. The evaluator shall ensure that this description includes the case where an existing file or set of files is encrypted for the first time; a new file or set of files is created and encrypted; an existing file or set of files is re-encrypted (that is, it had been initially encrypted; it was decrypted (by the TOE) for use by the user, and is then subsequently re-encrypted); 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.*
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 encrypted (e.g., choice of algorithm used, key size, etc.), then those instructions shall be included in the Operational Guidance and verified by the evaluator. 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 assurance activities of this EP.*
Tests	*The evaluator shall also perform the following tests. 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 decryption/encryption scenario listed in the TSS, the evaluator shall ensure that the TSF is able to successfully encrypt and decrypt 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 operation is completed.

4.2.3 Class: Security Management (FMT)

82 The primary intent in this section is to call out critical activities that must be performed by the user (or administrator) in order to use the TOE in a safe manner. The critical activities are defined as those that reference the Cryptographic Support items in Section 4.2.1.

83

Specification of Management Functions (FMT_SMF)

FMT_SMF.1 **Specification of Management Functions**

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions: [selection:
change password/passphrase authentication credential;
disable key recovery functionality;
[assignment: *no other function; configure password/passphrase complexity setting; configure cryptographic functionality; other management functions provided by the TSF*]
].

Application Note:

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

- *If password or passphrase authorization factors are implemented by the TOE, then the appropriate "change" selection must be included, along with FIA_FCT_EXT.1(2) from Appendix C.*
- *If the TOE provides for a password/passphrase complexity setting, then "configure password/passphrase complexity setting" will be included, and the specifics of the functionality offered can either be written from the requirement as bullets points, or included in the TSS.*
- *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 does include a key recovery function, the TOE must provide the capability for the user to turn this functionality off so that no recovery key is generated and no keys are permitted to be exported.*
- *If "other management functions" are assigned, a validation authority must be consulted to*

ensure the assurance activities and other functionality requirements that may be needed are appropriately specified so that the ST can claim conformance to this EP.

Assurance Activities:

85 *The assurance activities for this component will be driven by the selections made by the ST author. This section describes assurance activities for all possible selections in an ST; it should be understood that if a capability is not selected in the ST, the noted assurance activity does not need to be performed. The following sections are divided up into "Required Activities" and "Conditional Activities" for ease of reference.*

Activity	**Assurance Activity**
TSS	***Conditional Activities:*** *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.* ***Disable Key Recovery****: If the TOE supports key recovery, this must be stated in the TSS. The TSS shall also describe how to disable this functionality. This includes a description of how the recovery material is provided to the recovery holder. The guidance for disabling this capability shall be described in the AGD documentation.*
Guidance	***Conditional Activities:*** *The evaluator shall examine the Operational Guidance to ensure that it describes how the password/passphrase-based authorization factor is to be changed.* ***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	***Conditional Activities:*** *The evaluator shall set all length and complexity settings offered by the TOE. The evaluator shall then attempt to enter values that violate those settings and ensure they are not accepted.* *Disable Key Recovery*: *If the TOE provides key recovery capability, then the evaluator shall devise a test that ensures that the key recovery capability has been or can be disabled following the guidance provided by the vendor.* ***Cryptographic Configuration:*** *Testing for this activity is performed for other components in this EP.*

4.2.4 Class: Protection of the TSF (FPT)

Extended: Protection of FEK (FPT_FEK_EXT)

FPT_FEK_EXT.1 **File Encryption Key (FEK) Support**

FPT_FEK_EXT.1.1 The TSF shall [selection:

- Never store a FEK conditioned from a Password/Passphrase in non-volatile memory;
- Store a FEK in Non-Volatile memory conformant with FPT_KYP_EXT.1

].

Application Note:

86 *FPT_FEK_EXT.1.details how a FEK may be directly conditioned from a password/passphrase or may be a randomly generated from an approved randomizer.*

Activity	**Assurance Activity**
TSS	*In TOEs where the FEK is protected with a KEK, the FEK will need to be encrypted and stored in non-volatile memory when not being used to decrypt/encrypt a file. (Typically, the encrypted FEK is stored in the meta-data of the encrypted file(s).) The evaluator shall examine the TSS to ensure that it describes how the FEK is encrypted, both after its initial creation and after it has been decrypted for use (note that in the entirely likely possibility that the FEK is not re-encrypted, then this case must be indicated in the TSS and the description for FCS_CKM_EXT.4 will cover disposal of the plaintext FEK). The evaluator shall further check to ensure that the TSS describes how the FEK 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.1(5).*
Guidance	*None*
Tests	*Test 1: An example ciphertext file generated via the TOE shall be provided to the evaluator with the accompanying FEK and prerequisite authorization information used for encryption. The evaluator will use the TOE in conjunction with a debugging or forensics utility to attempt a decrypt 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 FEK string. The evaluator must document each command, program or action taken during this process, and must confirm that the FEK 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 FEK was written to non-volatile memory, they should*

	be able to identify the cause (i.e. the TOE wrote the FEK to disk, the TOE platform dumped volatile memory as a page file, etc), and document the reason for failure to comply with the requirement.

4.2.5 Class: Protection of the TSF (FPT)

FPT_KYP_EXT.1 **Extended: Protection of Key and Key Material (FPT_KYP_EXT)**

FPT_KYP_EXT.1.1 The TSF shall [selection: not store keys in non-volatile memory, only store keys in non-volatile memory when wrapped, as specified in FCS_COP.1(5 unless the key meets any one of following criteria [selection:

- 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 the other half of the key split is either [selection: wrapped as specified in FCS_COP.1(5) or derived and not stored in non-volatile memory.]
- The plaintext key is stored on an external storage device for use as an authorization factor.
- The plaintext key is used to wrap a key as specified in FCS_COP.1(5) that is already wrapped as specified in FCS_COP.1(5).]

].

Application Note:

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

Activity	Assurance Activity
TSS	*The evaluator shall verify the TSS for a high level description of method used to protect keys stored in non-volatile memory.* *The evaluator shall verify the TSS 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(5) 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*

4.3 Security Functional Requirements for the Software File Encryption Application or Client Platform

87 As indicated in Section 1.3.2, security functional requirements in the main body of the EP are divided into those that must be satisfied by the file encryption application (the TOE), and those that must be satisfied by either the TOE or the platform on which it runs. This section contains requirements that must be met, but they can either be met by the TOE or the platform on which the TOE operates. Assurance activities are therefore constructed such that those that apply when the requirements are met by the TOE are identified, and those that are performed when the platform on which the TOE operates implements the required functionality are likewise identified. If a test or documentation assurance activity is specified that is not specifically associated with either the TOE or the TOE platform, then it applies regardless of where the requirement is implemented.

4.3.1 Class: Cryptographic Support (FCS)

FCS_CKM_EXT.4 **Extended: Cryptographic Key Destruction**

FCS_CKM_EXT.4.1 The application shall [selection:

<u>invoke platform-provided key destruction;</u>

<u>implement key destruction using [selection:</u>

- <u>For volatile memory, the erasure shall be executed by a single direct overwrite [selection: consisting of a pseudo-random pattern using the TSF’s RBG, consisting of a pseudo-random pattern using the host platform’s RBG, consisting of zeroes] following by a read-verify.</u>
- <u>For non-volatile storage, the erasure shall be executed by:</u>

  <u>A [selection: single, three or more times] overwrite of key data storage location consisting of [selection: a pseudo random pattern using the TSF’s RBG (as specified in FCS_RBG_EXT.1, a pseudo-random pattern using the host platform’s RBG, a static pattern], followed by a [selection: read-verify, none]. If read-verification of the overwritten data fails, the process shall be repeated again;</u>

<u>] that meets the following: [selection: NIST SP800-88, no standard]</u>

] for destroying all plaintext keying material and cryptographic security parameters when no longer needed.

Application Note:

For the purposes of this requirement, plaintext keying material refers to authentication data, passwords, symmetric keys, data used to derive keys, etc.

For Mobile Devices, it is assumed that the TOE will call the platform for the memory management calls (and the Platform meets the MDF PP) to destroy the plaintext keying material when it is no longer necessary, including when the TOE is powered down and when the wipe function is performed. In the future, "no longer needed", will include keys generated for protecting sensitive data received while in a locked state.

The destruction indicated above applies to each intermediate storage area for plaintext key/cryptographic critical security parameter (i.e., any storage, such as memory buffers, that is included in the path of such data) upon the transfer of the key/cryptographic critical security parameter to another memory location.

Assurance Activities:

Activity	**Assurance Activity**
TSS	*If the platform provides the key destruction, then the evaluator shall examine the TSS to verify that it describes how the key destruction functionality is invoked.* *If the application invokes key destruction, the evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption and/or data authentication), private keys, and CSPs used to generate key; when they are zeroized (for example, immediately after use, on system shutdown, etc.); and the type of zeroization procedure that is performed (overwrite with zeros, overwrite three times with random pattern, etc.). If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the zeroization procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are zeroized by overwriting once with zeros, while secret keys stored on the internal hard drive are zeroized by overwriting three times with a random pattern that is changed before each write").*
Guidance	*None*
Tests	*These tests are only for key destruction provided by the application:* *Test 1: For each type of authorization service, encryption mode and encryption operation, a known authorization factor, FEK and KEK must be provided to the*

	evaluator with an associated ciphertext data set (e.g. if a passphrase is used to create a KEK, then the ciphertext containing the FEK as well as the KEK itself must be provided to the evaluator. If a passphrase is used to generate a FEK, then the ciphertext file encrypted with the FEK as well as the FEK must be provided to the evaluator.) The evaluator will use the TOE in conjunction with a debugging or forensics utility to attempt to authorize themselves, resulting in the generation of a FEK or decryption of a FEK. The evaluator will ascertain from the TSS what the vendor defines as "no longer needed" and execute the sequence of actions via the TOE to invoke this state. At this point, the evaluator should take a dump of volatile memory and search the retrieved dump for the provided authorization credentials, KEKs or FEKs (e.g. if the password was "PaSSw0rd", perform a string search of the forensics dump for "PaSSw0rd"). The evaluator must document each command, program or action taken during this process, and must confirm that no plaintext keying material resides in volatile memory. The evaluator must perform this test three times to ensure repeatability. If during the course of this testing the evaluator finds that keying material remains in volatile memory, they should be able to identify the cause (i.e. execution of the grep command for "PaSSw0rd" caused a false positive) and document the reason for failure to comply with this requirement. The evaluator will repeat this same test, but looking for keying material in non-volatile memory -- in some cases, the non-volatile memory testing may be satisfied by other assurance activities (see FCS_CKM_EXT.1 and FPT_FEK_EXT.1). *Test 2: For each data authentication mechanism supported by the TOE, the evaluator must be provided known keying material with the associated ciphertext file(s). The evaluator will attempt to authenticate the ciphertext data using the known key. The evaluator will ascertain from the TSS what the vendor defines as "no longer needed" and execute the sequence of actions via the TOE to invoke this state. Once this state is attained, the evaluator shall take a forensics dump of volatile memory and perform a search for the authentication keying material (i.e. if a FAK is used as input to an HMAC, then the evaluator will look for the FAK string in the forensics dump). The evaluator must document each command, program or action taken during this process, and must confirm that no plaintext keying material resides in volatile memory. The evaluator must perform this test three times to ensure repeatability. If during the course of this testing the evaluator finds that keying material remains in volatile memory, they should be able to identify the cause and document the reason for failure to comply with this requirement. The evaluator will repeat this same test, but looking for keying material in non-volatile memory -- in some cases, the non-volatile memory testing may be satisfied by other assurance activities (see FCS_CKM_EXT.4).*

Cryptographic Operation (FCS_COP)

This requirement is used to specify the symmetric decryption/encryption algorithm that is used to encrypt and decrypt the data.

FCS_COP.1(1) **Cryptographic operation (Data Encryption)**

FCS_COP.1.1(1) **Refinement:** The application shall [selection: implement platform-provided AES encryption, implement AES encryption] shall perform **data encryption and decryption** in accordance with a specified cryptographic algorithm **AES used in** [selection:
CBC (as defined in NIST SP 800-38A);
XTS (as defined in NIST SP 800-38E)
] mode and cryptographic key sizes
[selection:
128 bits;
256 bits
].

Application Notes:

88 *The intent of this requirement is to specify the approved AES modes that the ST author may select for AES encryption of the appropriate information on the file encryption software. The first selection indicates whether the TOE or the platform performs the actual cryptographic operations. For the second selection, the ST author should indicate the mode or modes supported by the TOE/platform implementation. The third selection indicates the key size to be used, which is identical to that specified for FCS_CKM_EXT.1. The fourth selection must agree with the mode or modes chosen in the first selection. If multiple modes are supported, it may be clearer in the ST if this component was iterated.*

89 *The CBC encryption mode may also be used to encrypt sets of files and must follow NIST SP 800-38 A to use unique IVs for each file that is encrypted.*

90 *Future versions of this EP may include new cryptographic modes as they are reviewed and approved by NIST.*

Assurance Activities:

Activity	**Assurance Activity**
TSS	**Requirement met by the platform** *If the platform provides the AES symmetric encryption/decryption, then the evaluator shall examine the TSS to verify that it describes how the key destruction encryption/decryption is invoked.* **Requirement met by the TOE** *If multiple modes are supported, the evaluator examines the TSS and guidance*

	documentation to determine how a specific mode/key-size is chosen by the end user. The evaluator then tests each mode/key size combination in the manner found in the following sections, as appropriate.
Guidance	*None*
Tests	*These tests are only for data encryption provided by the application:* **AES-CBC Tests** ***AES-CBC Known Answer Tests*** *There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.* ***KAT-1.*** *To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.* *To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.* ***KAT-2.*** *To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.* *To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.* ***KAT-3.*** *To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].* *To test the decrypt functionality of AES-CBC, the evaluator shall supply the two*

sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. *To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].*

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test
The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests
The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
if i == 1:
```

	CT[1] = AES-CBC-Encrypt(Key, IV, PT) PT = IV else: CT[i] = AES-CBC-Encrypt(Key, PT) PT = CT[i-1] *The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.* *The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.* ***<u>XTS-AES Monte Carlo Test</u>*** *The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:* ***256 bit (for AES-128) and 512 bit (for AES-256) keys*** ***Three data unit (i.e., plaintext) lengths**. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 2^{16} bits, whichever is smaller.* *using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.* *The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.* *The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.*

FEK decryption and encryption (Key Wrapping)

92 *This requirement specifies the operations to be used if the FEK are encrypted and decrypted using a KEK. If intermediate keys are used, the ST author iterates this requirement to specify the operations used in those cases.*

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

FCS_COP.1.1(5) **Refinement:** The application shall [selection: <u>use platform-provided</u>

functionality to perform Key Wrapping, implement functionality to perform Key Wrapping] in accordance with a specified cryptographic algorithm
[selection:

AES Key Wrap;
AES Key Wrap with Padding;
RSA using the KTS-OAEP-basic scheme;
RSA using the KTS-OAEP-receiver-confirmation scheme;
ECC CDH

]
and the cryptographic key size
[selection:

128 bits (AES), 256 bits (AES), 2048 (RSA), 4096 (RSA), 256-bit prime modulus (ECC CDH), 384-bit prime modulus (ECC CDH)

]
that meet the following:
[selection:

"NIST SP 800-38F" for Key Wrap (section 6.2) and Key Wrap with Padding (section 6.3);
"NIST SP 800-56B" for RSA using the KTS-OAEP-basic (section 9.2.3) and KTS-OAEP-receiver-confirmation (section9.2.4) scheme, "NIST SP 800-56A rev 2" for ECC CDH (sections 5.6.1.2 and 6.2.2.2)

].

Application Note:

93 *This requirement specifies the protection of the FEK (that is, protecting it using the KEK specified in FCS_CKM_EXT.1) and unwrapping/decryption of the FEK with the KEK so that it may be used to encrypt or decrypt files or set of files.*

94 *This requirement allows the TSF to control how the FEK is encrypted and decrypted. When encrypting the FEK, the TSF may pass the FEK to the operational environment with various amounts of information. For instance, if 128-bit AES Key Wrap is being used, the TSF may invoke an interface specifying these parameters. If RSA is being used, the FEK may invoke a crypto-library and pass the private key and the FEK to the crypto-library; or it may invoke crypto-functionality on a smart card that contains the private key, so the TSF only passes the FEK.*

95 *In the first selection, the ST author chooses the entity that performs the decryption/encryption of the FEK. If one operation is done by the TOE platform (e.g., decryption of the FEK) and one operation is done by the TSF (e.g., encryption of the FEK), the ST author should iterate and refine the requirement to reflect this functionality. Iteration can also be used if the TOE supports either option; in this case the assurance activities will be performed for all claimed modes.*

96 *In the second selection, the ST author chooses the method by which the KEK is used to encrypt the FEK:*

- *Using one of the two AES-based Key Wrap methods specified in NIST SP 800-38F;*
- *Using one of the two the KTS-OAEP schemes for RSA as described in NIST SP 800-56B (KTS-OAEP-basic described in section 9.2.3*

- *Using ECC CDH as described in NIST SP 800-56A section 6.2.2.2. In this case, the ST author also incorporates FCS_CKM.1(1) in Appendix C to ensure the ephemeral keys to be used in the exchange with the external entity are generated. Any key wrap mode included in NIST SP 800-38 F is allowed.*

97 *Based on the method(s) selected, the last selection should be used to select the appropriate reference(s). The fourth selection should be made to reflect the size of the KEK; 2048/4096 is used for the RSA-based schemes, while the size of the prime modulus is used for ECC-based schemes.*

Assurance Activities:

Activity	**Assurance Activity**
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 examine the TSS to ensure there is a high-level description of how the FEK is protected.*
Guidance	*None*
Tests	*These tests are only for data encryption provided by the application:* • *Test 1: The evaluator shall use platform tools (such as a debugger) to generate a FEK to be generated and capture the value of the FEK. The evaluator shall then continue with the TOE operation which will result in an encrypted resource, as well as an encrypted FEK being associated with the resource as described in the TSS. The evaluator shall then examine the encrypted FEK to determine that it is different than the value of the unencrypted FEK. The evaluator shall then use the information provided in the ST and TSS to determine that the unencrypted FEK—when wrapped according to the algorithm and parameters used by the TOE as described—produces the value observed for the encrypted FEK.* ***AES Key Wrap (with or without padding)*** *If AES Key Wrap is used to decrypt/encrypt the key, the evaluator shall examine the TSS to determine that it specifies that the implementation conforms to SP 800-38F with the appropriate (with or without padding) Key Wrap section using AES.*

	The evaluator shall also perform the verification procedures outlined in the testing methodology, "The Key Wrap Validation System". (http://csrc.nist.gov/groups/STM/cavp/documents/mac/KWVS.pdf) ***RSA*** *The evaluator shall check the TSS to ensure it describes the various values used for the RSA-OAEP encryption and decryption scheme described in NIST SP 800-56B, section 7.2.2 and other referenced sections.* *The evaluator shall also perform the validation procedures outlined in http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-rsa-cryptography-standard.htm.* ***ECC CDH*** *The evaluator shall verify a TOE's implementation of the ECC DH key agreement scheme using the following Function and Validity tests. These validation tests verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MAC data and the calculation of MAC tag.* ***Function Test*** *The Function test verifies the ability of the TOE to implement the key agreement scheme correctly. To conduct this test, the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one NIST approved curve per 10 sets of ephemeral public keys. The evaluator shall obtain the DKM, the corresponding TOE's public keys, the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields. The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values. If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.*

	Validity Test *The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 30 test vectors consisting of data sets including the selected NIST approved curves, the evaluator's public keys, the TOE's ephemeral public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.* *The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial public key validation, the evaluator will also individually inject errors in the static public keys, the ephemeral public keys and the TOE's ephemeral private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function. At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).* *The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.*

FCS_IV_EXT.1 **Extended: Initialization Vector Generation**

FCS_IV_EXT.1.1 The application shall [selection: implement platform-provided functionality to generate IVs, generate IVs] in accordance with Appendix H: Initialization Vector Requirements for NIST-Approved Cipher Modes.

Application Note:

98 *Appendix G lists the requirements for composition of IVs according to the NIST Special Publications for each cipher mode. The composition of IVs generated for encryption according to a cryptographic protocol is addressed by the protocol. Thus, this requirement addresses only IVs generated for key storage and data storage encryption.*

Assurance Activities:

Activity	Assurance Activity
TSS	**Requirement met by the platform** *If the platform provides the IV generation, then the evaluator shall examine the TSS to verify that it describes how the IV generation is invoked.* **Requirement met by the TOE** *The evaluator shall examine the key hierarchy section of the TSS to ensure that the encryption of all keys is described and the formation of the IVs for any data encrypted by the same key meets FCS_IV_EXT.1.*
Guidance	*None*
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:
One;
a conditioned password as the FEK;
KEKs originating from one or more authorization factors(s) to the FEK(s) 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;
implement key wrapping as specified in FCS_COP.1(5);
implement key combining as specified in FCS_SMC_EXT.1
]
while maintaining an effective strength of [selection: 128 bits, 256 bits]
].

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; including those in areas of protected.

This requirement also describes how keys are stored.

Assurance Activities:

Activity	Assurance Activity
TSS	**Requirement met by the TOE** *The evaluator shall verify the TSS* describes a high level description of the key hierarchy for all authorizations methods selected in FIA_AUT_EXT that are used to protect the KEK or FEK. The evaluator shall examine the TSS to ensure it describes the key chain in detail. The description of the key chain shall be reviewed to ensure it maintains a chain of keys using key wrap that meet FCS_COP.1(5).* *The evaluator shall verify the TSS* to ensure that it describes how the key chain process functions, such that it does not expose any material that might compromise any key in the chain. A high-level description should include a diagram illustrating the key hierarchy implemented and detail where all keys and keying material is stored or what it is derived from. The evaluator shall examine the key hierarchy to ensure that at no point the chain could be broken without a cryptographic exhaust or knowledge of the KEK or FEK and the effective strength of the FEK is maintained throughout the Key Chain.* **if necessary, this information could be contained in a proprietary document and not appear in the TSS.* **Requirement met by the platform** *If the platform provides the IV generation, then the evaluator shall examine the TSS to verify that it describes how the IV generation is invoked.*
Guidance	*None*
Tests	*None*

4.3.2 Class: Identification and Authentication (FIA)

FIA_AUT_EXT.1 User Authorization

FIA_AUT_EXT.1.1 The application shall [selection: implement platform-provided functionality to provide user authorization, provide user authorization] based on [selection: external entity authorization factors, password/passphrase authorization factors].

Application Note:

99 *Requirements that pertain to the selection are contained in Appendix C. The ST author will include FIA_FCT_EXT.1(1) in the ST if the TOE supports RSA/ECC CDH authorization factors, and will include FIA_FCT_EXT.1(2) in the ST if the TOE supports password/passphrase authorization factors.*

100 *It is possible that the platform is providing the actual authorization functionality.*

Assurance Activities:

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

Activity	**Assurance Activity**
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** *Nothing additional.* **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.*
Guidance	*The evaluator shall verify that the operational guidance includes instructions for configuring the selected authorization method.*
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.*

4.3.3 Class: User Data Protection (FDP)

Extended: Protection of Selected User Data (FDP_PRT_EXT)

FDP_PRT_EXT.1 **Extended: Protection of Selected User Data**

FDP_PRT_EXT.1.2 The application shall [selection: <u>invoke platform-provided functionality, implement functionality</u>] 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 upon completion of the decryption/encryption operation.

Application Note:

102 *The intent is that the TSF controls the use and clearing of any data that it manipulates. However, it is possible that the TSF shall only be invoking the The TSF is not responsible for temporary files that an editor application creates. An optional requirement on cleaning up the temporary files created by an editor application is in Appendix B.*

103 *The TSF has "completed the decryption/encryption operation" after it has decrypted the file (or set of files) and any edited data has been stored encrypted and the plaintext editor has been closed.*

Assurance Activities:

Activity	**Assurance Activity**
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 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 that it describes all temporary files/resources created or memory used during the decryption/encryption process. 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	*None*
Tests	*These tests are only for application provided functionality:* *For each type of encryption mode and encryption operation, a known plaintext file, ciphertext file and the associated keying material must be supplied to the evaluator. The evaluator will use the TOE in conjunction with a debugging or forensics utility to attempt to encrypt the plaintext and decrypt the ciphertext. The evaluator will ascertain from the TSS what the vendor defines as "no longer needed" for plaintext information and execute the sequence of actions via the TOE to invoke this state. At this point, the evaluator should take a dump of volatile memory and search the retrieved dump for any plaintext*

	information. The evaluator must document each command, program or action taken during this process, and must confirm that no sensitive data resides in volatile memory. The evaluator must perform this test three times to ensure repeatability. If during the course of this testing the evaluator finds that plaintext material remains in volatile memory, they should be able to identify the cause and document the reason for failure to comply with this requirement. The evaluator will repeat this same test, but looking for sensitive data in non-volatile memory.

4.4 Security Assurance Requirements for the File Encryption Application (TOE)

104 The Security Objectives for the TOE in Section 3.1 were constructed to address threats identified in Section 2. The Security Functional Requirements (SFRs) in Section 4.2 and 4.3 are a formal instantiation of the Security Objectives. The EP draws from EAL1 the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

105 As indicated in the introduction to Section 4, while this section contains the complete set of SARs from the CC, the Assurance Activities to be performed by an evaluator are detailed both in Sections 4.2 and 4.3 as well as in this section.

106 The general model for evaluation of TOEs against STs written to conform to this EP is as follows. After the ST has been approved for evaluation, the CCTL will obtain the TOE, supporting environmental IT, and the administrative guides for the TOE. The Assurance Activities listed in the ST (which will be refined by the CCTL to be TOE-specific, either within the ST or in a separate document) will then be performed by the CCTL. The results for the assurance activities will be documented and presented (along with the Operational Guidance used) for validation by CCEVS.

107 For each family, "Developer Notes" are provided on the developer action elements to clarify what, if any, additional documentation/activity needs to be provided by the developer. For the content/presentation and evaluator activity elements, additional assurance activities (to those already contained in Section 4) are described as a whole for the family, rather than for each element. Additionally, the assurance activities described in this section are complementary to those specified in Sections 4.2 and 4.3. The totality of the assurance activities specified in Sections 4.2 and 4.3 and this section are sufficient to provide EAL1 assurance.

108 The TOE security assurance requirements, summarized in Table 5, identify the management and evaluative activities required to address the threats identified in Section 2 of this EP. Section 4.5 provides a succinct justification for choosing EAL1 as the assurance level for this EP.

Assurance Class	Assurance Components	Assurance Components Description
Development	ADV_FSP.1	Basic Functional Specification
Guidance Documents	AGD_OPE.1	Operational user guidance

	AGD_PRE.1	Preparative User guidance
Tests	ATE_IND.1	Independent testing - conformance
Vulnerability Assessment	AVA_VAN.1	Vulnerability analysis
Life Cycle Support	ALC_CMC.1	Labeling of the TOE
	ALC_CMS.1	TOE CM coverage

Table 5: TOE Security Assurance Requirements

4.4.1 Class ADV: Development

109 At EAL1, the information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (TSS) portion of the ST. While it is not required that the TOE developer write the TSS, the TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The Assurance Activities contained in Sections 4.2 and 4.3 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

4.4.1.1 ADV_FSP.1 Basic functional specification

110 The functional specification describes the TSFIs. At EAL1, it is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this EP will necessarily have interfaces to the Operational Environment that cannot be directly invoked by TOE users, at EAL1 there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. The activities for this family for this EP should focus on understanding the interfaces presented in the TSS in response to the functional requirement and the interfaces presented in the AGD documentation. No additional "functional specification" document should be necessary to satisfy the assurance activities specified.

111 In understanding the interfaces to the TOE, it is important to consider that the primary threat to be countered is that where an attacker gains access to a host machine with sensitive data and attempts to gain access to the TOE functionality in order to decrypt the data on the host drive. Once an attacker has access to the TOE authentication interface the attacker may attempt to guess the authorization factors to access the TOE's decryption functionality. The operational interface (how the TOE is configured) also needs to be described.

112 The interfaces that need to be evaluated are characterized through the information needed to perform the assurance activities listed, rather than as an independent, abstract list.

Developer action elements:

ADV_FSP.1.1D	The developer shall provide a functional specification.
ADV_FSP.1.2D	The developer shall provide a tracing from the functional

	specification to the SFRs.
Developer Note:	As indicated in the introduction to this section, the functional specification is comprised of the information contained in the AGD_OPR and AGD_PRE documentation, coupled with the information provided in the TSS of the ST. The assurance activities in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.
	Content and presentation elements:
ADV_FSP.1.1C	The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.2C	The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.3C	The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.
ADV_FSP.1.4C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
	Evaluator action elements:
ADV_ FSP.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
ADV_ FSP.1.2E	The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

Assurance Activities:

113 *There are no specific assurance activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Sections 4.2 and 4.3, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided. For example, if the TOE provides the capability to configure the key length for the AES encryption algorithm but fails to specify an interface to perform this function, then the assurance activity associated with FMT_SMF would fail.*

4.4.2 Class AGD: Guidance Documents

114 The guidance documents will be provided with the developer's security target. As indicated in the introduction, the duties of actual "administrators" are fairly restricted, so the guidance documents will contain information that is required by and used by all users of the TOE. To this end "authorized user" is used in most cases in the text below; when "administrator" is used (except in the verbatim requirements from the CC) it is referring to the subset of users with responsibility for creating strong password/passphrase authorization factors.
Guidance must be provided for every Operational Environment that the product supports as claimed in the ST. This guidance includes

- instructions to successfully install the TOE in that environment; and
- instructions to manage the security of the TOE as a product and as a component of the larger Operational environment.

Guidance pertaining to particular security functionality is also provided; requirements on such guidance are contained in the assurance activities specified in Sections 4.2 and 4.3.

In addition to the areas already mentioned, the guidance specifies which power management modes (e.g., hibernate, sleep) conform to OE.POWER_SAVE and provides instructions how to disable those that do not conform to be disabled.

4.4.2.1 AGD_OPE.1 Operational User Guidance

Developer action elements:

AGD_OPE.1.1D	The developer shall provide operational user guidance.
Developer Note:	Rather than repeat information here, the developer should review the assurance activities for this component to ascertain the specifics of the guidance that the evaluators will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.1C	The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.
AGD_OPE.1.2C	The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.
AGD_OPE.1.3C	The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.
AGD_OPE.1.4C	The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that

need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5C The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AGD_OPE.1.6C The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7C The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.1E The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

Assurance Activities:

115 *Some of the contents of the Operational Guidance will be verified by the assurance activities in Sections 4.1 and 4.2. This guidance will cover all platforms that the TOE claims conformance to. However, two additional warnings shall be provided in the guidance to users. The guidance shall warn authorized users that they must not let the host machine leave their physical control while the host is powered on and encrypted files are opened in plain text. Additionally, it shall state that authorized users shall not leave/store the password/passphrases, and/or external hardware stored authorization factors unprotected with the host machine or if multiple factors are used, with each other.*

116 *The following additional information is also required:*

117 *Non-mobile systems (and laptops in particular) generally support a number of modes that are targeted at states of user inactivity: power management modes (e.g., hibernation, sleep/standby, auto-shutdown). There are two areas that need to be covered in the guidance.*

118 *The first addresses the steps that must be performed to configure the platform so that the system powers down completely after a period of user inactivity; the point being that on power-down, the residual keying/plaintext material in volatile memory will be erased and all user resources in non-volatile are encrypted if so designated. While it is allowable for a function such as a PC-screen lock to become active due to user inactivity prior to the power-down process being initiated, it is not a substitute for power-down and does not satisfy this requirement.*

119 *The second addresses instructions to disable the power saving modes that do not completely power down the system and shut down the operating system; instead, the system has some state stored (either in volatile memory or on disk) allowing the user to start working from where they left off prior to the mode that was entered. Conformant TOEs are not allowed to enter any modes that leave the computer in a compromised state.*

120 *If the TOE claims to provide protection to data while in a power saving mode, the requirements in Appendix B.2 will apply.*

121 *For Mobile devices, control over power and lifecycle management is performed differently because the TOE is dependent on the TOE Platform to provide the mechanisms and implementation for these state changes. Background execution is performed frequently and occurs automatically when the TOE is not in focus. The TOE is expected to substantiate any claim of providing data protection when the TOE platform signals that a state change as described above has occurred.*

4.4.2.2 AGD_PRE.1 Preparative procedures

Developer action elements:

AGD_PRE.1.1D	The developer shall provide the TOE including its preparative procedures.
Developer Note:	As with the Operational Guidance, the developer should look to the assurance activities to determine the required content with respect to preparative procedures.

Content and presentation elements:

AGD_ PRE.1.1C	The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.
AGD_ PRE.1.2C	The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_ PRE.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
AGD_ PRE.1.2E	The evaluator *shall apply* the preparative procedures to confirm that the TOE can be prepared securely for operation.

Assurance Activities:

122 *As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms (that is, combination of hardware and operating system) claimed for the TOE in the ST.*

123 *The evaluator shall check to ensure that the following guidance is provided:*

- *Instructions and information are provided to the authorized user detailing how to configure the product so that selected user data on the host machine is encrypted when setting up the product, and that this is the only allowed configuration for conformant TOEs.*
- *If there are requirements on the operational environment with respect to the cryptographic functionality listed in Appendix C, Section C.1, then the evaluator shall ensure that acceptable implementations for the TOE are identified, and that testing is conducted in an allowed configuration identified in the guidance.*

4.4.3 Class ATE: Tests

124 Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this EP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

4.4.3.1 ATE_IND.1 Independent testing - Conformance

125 Testing is performed to confirm the functionality described in the TSS as well as the operational documentation provided. The focus of the testing is to confirm that the requirements specified in Sections 4.2 and 4.3 are being met, although some additional testing is specified for SARs in Section 4.3. The Assurance Activities identify the minimum testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this EP.

	Developer action elements:
ATE_IND.1.1D	The developer shall provide the TOE for testing.
	Content and presentation elements:
ATE_IND.1.1C	The TOE shall be suitable for testing.
	Evaluator action elements:
ATE_IND.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.1.2E	The evaluator *shall test* a subset of the TSF to confirm that the TSF operates as specified.

Assurance Activities:

126 *The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the body of this EP's Assurance Activities.*

While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

127 *The Test Plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platform and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. Evaluators shall especially consider OS-based mechanisms that deal with power management modes such as power-saving and hibernation functions when writing this justification. If all platforms claimed in the ST are tested, then no rationale is necessary.*

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

129 *The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results. The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.*

4.4.4 Class AVA: Vulnerability assessment

130 For the first generation of this protection profile, the evaluation lab is expected to survey open sources to learn what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, evaluators will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

4.4.4.1 AVA_VAN.1 Vulnerability survey

	Developer action elements:
AVA_VAN.1.1D	The developer shall provide the TOE for testing.
	Content and presentation elements:
AVA_VAN.1.1C	The TOE shall be suitable for testing.

Evaluator action elements:

AVA_VAN.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.1.2E	The evaluator *shall perform* a search of public domain sources to identify potential vulnerabilities in the TOE.
AVA_VAN.1.3E	The evaluator *shall conduct* penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

Assurance Activities:

131 *As with ATE_IND the evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to determine the vulnerabilities that have been found in file encryption products in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. For example, if the vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this EP.*

4.4.5 Class ALC: Life-cycle support

132 At the assurance level provided for TOEs conformant to this EP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation at this assurance level.

4.4.5.1 ALC_CMC.1 Labeling of the TOE

133 This component is targeted at identifying the TOE such that it can be distinguished from other products or version from the same vendor and can be easily specified when being procured by an end user.

Developer action elements:

ALC_CMC.1.1D	The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.1.1C	The TOE shall be labeled with its unique reference.

Evaluator action elements:

ALC_CMC.2.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

Assurance Activities:

134 The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

4.4.5.2 ALC_CMS.1 TOE CM coverage

135 Given the scope of the TOE and its associated evaluation evidence requirements, this component's assurance activities are covered by the assurance activities listed for ALC_CMC.1.

Developer action elements:

ALC_CMS.1.1D	The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.1.1C	The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.
ALC_CMS.1.2C	The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

Assurance Activities:

136 The "evaluation evidence required by the SARs" in this EP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component.

4.5 Rationale for Security Functional Requirements

137 The rationale for choosing these security functional requirements is that this is the first U.S. Government Protection Profile for this technology. If vulnerabilities are found in these types of products, then more stringent security assurance requirements will be mandated based on actual

vendor practices.

This EP does not claim conformance to another PP.

Appendix A: Rationale

In this EP, the focus in the initial sections of the document is to use a narrative presentation in an attempt to increase the overall comprehensibility of the threats addressed by a File Encryption product; the methods used to mitigate those threats; and the extent of the mitigation achieved by compliant TOEs. This presentation style does not readily lend itself to a formalized evaluation activity, so this section contains the tabular artifacts that can be used for the evaluation activities associated with this document. Note that the rationale for threats, objectives, and requirements is contained in the narrative in the body of the document.

A.1 Security Problem Definition

A.1.1 Threats

Threat	Description of Threat
T.KEYING_MATERIAL_COMPROMISE	An attacker can obtain unencrypted key material (the KEK, the FEK, authorization factors, and random numbers, or any other values from which a key is derived) that the TOE has written to volatile memory, and use these values to gain unauthorized access to sensitive encrypted user data.
T.KEYSPACE_EXHAUST	An unauthorized user may attempt a brute force attack to determine cryptographic keys or authorization factors to gain unauthorized access to user or TSF data.
T.PLAINTEXT_COMPROMISE	An attacker may obtain unauthorized read access to sensitive plaintext material (the input to the file encryption) that the TOE has written to volatile memory as a result of the creation of a temporary file or improper memory clean-up.
T.TSF_FAILURE	Security mechanisms of the TOE may fail, leading to a compromise of the TSF.
T.UNAUTHORIZED_DATA_ACCESS	An unauthorized user that has access to filesystem on which a protected resource resides may gain access to data for which they are not authorized according to the TOE security policy.
T.UNSAFE_AUTHFACTOR_VERIFICATION	An attacker can take advantage of an unsafe method for performing verification of an authorization factor, resulting in exposure of the KEK, FEK, or user data.
T.PLAINTEXT_DATA_SPOOFING	An attacker can take advantage of certain encryption modes to modify the underlying plaintext without user awareness.

A.1.2 Assumptions

Assumption	Description of Assumption
A.AUTHORIZED_USER	Authorized users of the host machine are well-trained, not actively working against the protection of the data, and will follow all provided guidance.
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_PROTECTIO N	External entities that implement ECC CDH or RSA that are used to encrypt and decrypt a FEK have the following characteristics: meet National requirements for the cryptographic mechanisms implemented; require authentication via a pin or other mechanisms prior to allowing access to protected information (the decrypted FEK, or the private key); implement anti-hammer provisions where appropriate (for example, when a pin is the authentication factor).
A.SHUTDOWN	An authorized user will not leave the machine in a mode where sensitive information persists in non-volatile storage (e.g., power it down or enter a power managed state, such as a “hibernation mode”).
A.STRONG_OE_CRYPTO	All cryptography implemented in the Operational Environment and used by the TOE will meet the requirements listed in Appendix C of this EP. This includes generation of external token authorization factors by a RBG.
A.PLATFORM_STATE	The platform on which the TOE resides is free of malware that could interfere with the correct operation of the product.
A.AUTHORIZED_CONFIGURAT ION	Access and ability to modify the cryptographic configuration files may be done only by authorized users.
A.KEK_SECURITY	The KEK will be derived from a strong entropy source, attaining equal or greater bit strength to that of the block cipher it is used in.
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.2 Security Objectives

A.2.1 Security Objectives for the TOE

Objective	Objective Description
O.AUTHORIZATION	The TOE must enforce the entry of authorization factor(s) by authorized users to be able to encrypt and decrypt user data.
O.CORRECT_TSF_OPERATION	The TOE will provide the capability to test the TSF to ensure the correct operation of the TSF in its operational environment.
O.PROTECT_DATA	The TOE will decrypt/encrypt all user data that is provided to the file encryption program in order to protect it while it is not being activity accessed by the user.
O.FEK_SECURITY	The TOE will encrypt the FEK using a KEK created from one or more authorization factors so that a threat agent who does not have the authorization factor(s) will be unable to gain access to the user data by obtaining the FEK. The size of the FEK will be large enough to make a brute force attack infeasible.
O.KEY_MATERIAL_PROTECTION	The TOE shall ensure that unencrypted keys or keying material are properly removed from memory after use.
O.MANAGE	The TOE will provide all the functions and facilities necessary to support the authorized administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.
O. SAFE_AUTHFACTOR_VERIFICATION	The TOE shall perform verification of the authorization factors in such a way that the KEK, FEK, or user data are not inadvertently exposed.
O.WIPE_MEMORY	The TOE shall ensure that non-volatile memory space corresponding to sensitive plaintext material (encryption input) is wiped from the TOE’s memory. This includes temporary files that may have been created.
O.DATA_AUTHENTICATION *(optional)*	The TOE shall verify the integrity of the plaintext data using an approved data authentication method.

A.2.2 Security Objectives for the Operational Environment

138 The Operational Environment of the TOE implements technical and procedural measures to assist the TOE in correctly providing its security functionality (which is defined by the security objectives for the TOE). The security objectives for the Operational Environment consist of a set of statements describing the goals that the Operational Environment should achieve.

139 This section defines the security objectives that are to be addressed by the IT domain or by non-technical or procedural means. The assumptions identified in Section A.1.2 are incorporated as security objectives for the environment.

Objective	Objective Description
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 power down after a period of time in the same fashion as the user electing to shutdown the system (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.
OE.TRAINED_USERS	Authorized users of the host machine will be trained to follow all provided guidance.

Appendix B: Optional Requirements

140 As indicated in the introduction to this EP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this EP. Additionally, there are three other types of requirements specified in Appendices B, C, and D.

141 The first type (in this Appendix) are optional requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this EP. The second type (in Appendix C) are requirements based on selections in the body of the EP: if certain selections are made, then additional requirements in that appendix will need to be included. The third type (in Appendix D) are components that are not required in order to conform to this PP, but will be included in the baseline requirements in future versions of this EP, so adoption by File Encryption Product vendors is encouraged. Note that the ST author is responsible for ensuring that requirements that may be associated with those in Appendix B, Appendix C, and/or Appendix D but are not listed (e.g., FMT-type requirements) are also included in the ST.

B.1 Sharing Encrypted Resources

142 *An optional feature that may be claimed by a TOE is the ability to share encrypted resources. 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 smartcards, two different passwords, one using a password and another using a smartcard). If this capability is supported, then the ST author adds FDP_PRT_EXT.2 (and the associated application notes and assurance activities) to the ST.*

Extended: Protection of Selected User Data (FDP_PRT_EXT)

FDP_PRT_EXT.2	**Extended: Protection of Selected User Data**
FDP_PRT_EXT.2.1	The application 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 upon completion of the decryption/encryption operation.
FDP_PRT_EXT.2.2	The TSF shall support more than one user being able to access the same encrypted resource, each using a different authorization factor.
FDP_PRT_EXT.2.3	The TSF shall ensure that all temporary files created (including those by a third-party application, for example an editor) when decrypting/encrypting the user-selected file (or set of files) are removed or encrypted upon completion of the decryption/encryption operation.

Application Note:

143 *This is the primary requirement for encrypting and decrypting the protected resources (file or set of files). Apart from the actual encryption and decryption of the resources, there are three other*

functions specified by this requirement.

144 *FDP_PRT_EXT.2.2 requires that a single resource (file or set of files) be able to be encrypted, but sharable among more than one user. The TSF must support each user having a different authorization factor, which could be the same type of authorization factor but just a different value (like two different passwords), or different types of authorization factors altogether.*

145 *For FDP_PRT_EXT.2.2-2.3, 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, or any plaintext key material that could be used to recover that information from the encrypted resource, remains after the TSF has finished operating on that resource. In the context of FDP_PRT_EXT.2.2, 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 filesystem.*

Assurance Activities:

Activity	**Assurance Activity**
TSS	*FDP_PRT_EXT.2.2 - The evaluator shall examine the TSS to determine that it identifies each of the resources that is sharable in encrypted form (for instance, encrypted files may be sharable among users, but encrypted directories may not), and the method by which the resource can be shared among users with different authorization factors. This description shall also cover the TSF actions when adding or removing users to the set allowed to access the file.* *FDP_PRT_EXT.2.3 - 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.*
Guidance	*FDP_PRT_EXT.2.2 - 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 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 2: If the TSS or the third party file editor creates temporary files/resources during file decryption/encryption, the evaluator shall perform the following*

tests to verify that the temporary files/resources are destroyed. 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.

Test A (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, that means that plaintext from the encrypted file remains on the hard drive.*
- *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 encrypted file remains on the hard drive.*

Test B (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*
- *If Test 1 fails and Test 2 passes, the Operational Guidance shall include instructions for the users to perform encryption in the manner outlined in Test 2.*

- Assumption: Regardless of the length of the file, it is assumed that if any fragment of the original string is found, this reflects that there is a problem

	with the cleanup with the file encryptor.

B.2 Power Management Function

146 As indicated above, a platform on which the TOE runs may support one or more modes of "powering down" that is something less than a full shutdown by the user. In cases where these modes leave data in volatile memory, they may cause the security policies to be circumvented if the device (e.g., laptop) is taken by the attacker in this state.

147 Some TOEs may provide the facilities to cause information being transferred from volatile memory to disk to be encrypted as per FDP.PRT_EXT.1.1, leaving the information correctly protected whilst the platform is in a lower power mode (and no sensitive information is maintained in-memory). In these cases, the following requirements should be used by the ST Author to specify this capability.

Extended: Protection of Data in Power Managed States (FDP_PM_EXT)

FDP_PM_EXT.1	**Extended: Protection of Data in Power Managed States**
FDP_PM_EXT.1.1	The TSF shall protect all data stored to the disk drive during the transition to the [assignment: *powered-down state(s) 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.

Application Note:

For the first selection, the ST author fills in the state 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.

Assurance Activities:

Activity	**Assurance Activity**
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.
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.*
Tests	*The following tests must be performed by the evaluator for each supported State, type of resource, platform, and authorization factor:* *Test 1: Following the Operational guidance, configure the Operational Environment and the TOE so that the lower power state of the platform is protected. Open several resources (documented in the test report) that are protected. Invoke the lower power state. On resumption of normal power an attempting 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.*

B.3 Data Authentication Methods

Because data authentication can be achieved depending on the use of an authenticated block cipher, keyed hashing function or asymmetric verification method, a different set of requirements will be levied on the TOE depending on the selected choice.

B.3.1 Data Authentication with cryptographic, keyed hashing functions

FDP_AUT_EXT.2	**Extended: Data Authentication using cryptographic, keyed hash functions**
FDP_AUT_EXT.2.1	The TSF shall use a cryptographic, keyed hash function in accordance with FCS_COP.1(4).
FDP_AUT_EXT.2.2	The TSF shall use a File Authentication Key (FAK) in accordance with FCS_COP.1(6) 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 assume the MAC is at the end of the ciphertext file.
FDP_AUT_EXT.2.9	The FAK will be generated using a RBG that meets FCS_RBG_EXT.1 (from the AS PP).

Application Note:

148 *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.*

Assurance Activities:

Activity	**Assurance Activity**
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.*
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 1: Create an encrypted file and confirm that authentication of this file using the correct FAK will result in a success.* *Test 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.*

Extended: Authentication of Selected User Data (FDP_AUT_EXT)

FDP_AUT_EXT.1	**Extended: 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:

152 *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 EP 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.*

153

Assurance Activities:

Activity	**Assurance Activity**
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.
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 assurance activities of this EP.*
Tests	*The evaluator shall also perform the following tests. These tests may be performed in conjunction with the tests listed for FCS_COP.1(2) (from the AS PP), FCS_COP.1(3) (from the AS PP), and FCS_COP.1(4). 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.*

B.4 FAK Support

FCS_COP.1(6) FAK encryption/decryption support

FCS_COP.1.1(6)	The FAK shall be protected in the same manner as the FEK, in accordance with FCS_COP.1(5).

Application Note:

156 *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.*

Assurance Activities:
The evaluator shall follow the assurance activities as laid out in FCS_COP.1(5) to assert proper FAK protection.

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: <u>A FAK conditioned from a password/passphrase shall never be stored in non-volatile memory</u> <u>a FAK will be stored in non-volatile memory encrypted with a KEK as specified in FCS_COP.1(5) using authorization factors as specified in FCS_CKM_EXT.1</u>].
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.
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.1(6).

Application Note:
157 *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 may be directly conditioned from a password/passphrase or may be a randomly generated from an approved randomizer.

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.

Assurance Activities:

Activity	Assurance Activity
TSS	*FCS_CKM_EXT.5.1: The evaluator shall examine the TSS to determine how the FEK will actually be formed and details how the FEK 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.1(5). 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 the mechanisms specified in FCS_CKM_EXT.1. *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_EXT.4 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_EXT.4 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.1(5).*
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_SMC_EXT.1 Submask Combining

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

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.

Assurance Activities:

Activity	**Assurance Activity**
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.
Guidance	*None*
Tests	*None*

B.5 Data Authentication using asymmetric signing and verification

FDP_AUT_EXT.3	**Extended: 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(3) *(from the AS PP)* 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(2) *(from the AS PP)* 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 FIA_CKM.1(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 assume the signature is at the end

of the ciphertext file.

Application Note:

158 *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.*

Assurance Activities:

Activity	**Assurance Activity**
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.*
Guidance	*None.*
Tests	*The evaluator shall perform the following test:* *Test 1: Create an encrypted file and demonstrate that authentication of this file using the correct keying material will be successful.* *Test 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.*

Appendix C: Selection-Based Requirements

As indicated in the introduction to this EP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this EP. There are additional requirements based on selections in the body of the EP: if certain selections are made, then additional requirements below will need to be included.

C.1 Authorization Factors

161 The TOE may support password/passphrase-based, ECC CDH, and RSA authorization factors. One or more of these factors are used to derive the KEK. TOE-supported authorization factors are specified in FCS_CKM_EXT.1. In order to be conformant with this EP for password-based and passphrase-based authorization factors.

162 If password/passphrase-based authorization factors are supported, the ST author will include this requirement and the associated application notes and assurance activities.

FCS_CKM.1(A)	**Cryptographic key generation (Password/Passphrase conditioning)**
FCS_CKM.1.1(A)	**Refinement:** A password/passphrase used to generate a password authorization factor shall enable up to [assignment: *positive integer of 64 or more*] characters in the set of {upper case characters, lower case characters, numbers, and the following special characters: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”, and [assignment: *other supported special characters*] and shall perform [**Password-based Key Derivation Functions**] in accordance with a specified cryptographic algorithm **[HMAC-[selection: <u>SHA-256, SHA-384, SHA-512</u>]**], with [assignment: *positive integer of 4096 or more*] iterations, and output cryptographic key sizes [selection: <u>128, 256</u>] that meet the following: [**NIST SP 800-132**].
FCS_CKM.1.2(A)	The TSF shall generate all salts using a RBG that meets FCS_RBG_EXT.1 (from the AS PP) and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM.1.1(A).

Application Note:

163 *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.*

164 *The key cryptographic key sizes in the fourth selection should be made to correspond to the KEK key sizes selected in FCS_CKM_EXT.1.*

165 *This password/passphrase must be conditioned into a string of bits that forms the submask to be used as input into the KEK. Conditioning can be performed using one of the identified hash functions or the process described in NIST SP 800-132; the method used is selected by the ST Author. SP 800-132 requires the use of a pseudo-random function (PRF) consisting of HMAC with an approved hash function. The ST author selects the hash function used, also includes the appropriate requirements* *for HMAC and the hash function.*

166 *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 EP, a minimum iteration count of 4096 is required in order to ensure that twelve bits of security is added to the password/passphrase value. A significantly higher value is recommended to ensure optimal security.*

Assurance Activities:

Activity	**Assurance Activity**
TSS	*FCS_CKM_1.1(A): There are two aspects of this component that require evaluation: passwords/passphrases of the length specified in the requirement (at least 64 characters) 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 the TSS section to determine that it specifies that a capability exists to accept passwords/passphrases with the minimum number of characters specified in the ST in this assignment statement.* ***Support for PBKDF:*** *The evaluator shall examine the password hierarchy TSS to ensure that the formation of all KEKs or FEKs (as decided in the FCS_CKM_EXT.1 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_CKM_EXT.4.* *For the NIST SP 800-132-based conditioning of the password/passphrase, the required assurance activities will be performed when doing the assurance 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_1.2(A): 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 AS PP).*
Guidance	***Support for minimum length:*** *The evaluators shall also check the Operational Guidance to determine that there are instructions for guidance on how to*

	generate large passwords/passphrases external to the TOE and instructions for how to configure the password/passphrase length and optional complexity settings (note to Management section). This is important because many default settings for passwords/passphrases will not meet the necessary entropy needed as specified in this EP.
Tests	***Support for minimum length:*** *In addition to the analysis above, the evaluator shall also perform the following tests on a TOE configured according to the AGD_PRE guidance:* *Test 1: Ensure that the TOE supports passwords/passphrases of 64 characters.* *Test 2: Try entering a password/passphrase less than 64 characters.* *Test 3: If the TOE supports a password/passphrase length up to a maximum number of characters, n (which would be greater than 64), then ensure that the TOE will not accept more than n characters.* ***Conditioning:*** *No explicit testing of the formation of the authorization factor from the input password/passphrase is required.* ***Iteration count:*** *The evaluator shall verify that the iteration count for PBKDFs performed by the TOE comply with NIST SP 800-132 by ensuring that the TSS contains a description of the estimated time required to derive key material from passwords and how the TOE increases the computation time for password-based key derivation (including but not limited to increasing the iteration count).*

C.2 Cryptographic Key Generation

FCS_CKM.1(1) Cryptographic Key Generation (for asymmetric keys)

169 If ECC CDH is one of the methods used to protect the FEK as specified in FCS_COP.1(5) by the ST author, ephemeral keys are required to be generated and used to generate the shared secret used to protect the FEK. The following component will be included by the ST author when this selection is made.

FCS_CKM.1.1(1)	**Refinement:** The application shall [selection: invoke platform-provided functionality, implement functionality] shall generate **asymmetric** cryptographic keys **used for key establishment** in accordance with *NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*" for elliptic curve-based key establishment schemes and implementing "NIST curves" [selection: P-256, P-384] (as defined in FIPS PUB 186-4, "Digital Signature Standard") and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

Application Note:

170 *This component requires that the TOE/TOE platform be able to generate the public/private key pairs that are used for key establishment purposes when ECC CDH is used to protect the FEK; the ST author selects the entity that is performing the key generation activity in the first selection.*
The ST author also chooses the curves that are supported for the key pair generation activity; either or both can be selected.

Assurance Activity:

Activity	**Assurance Activity**
TSS	**Requirement met by the TOE** *The evaluator shall examine the TSS to ensure that it specifies which key size is used.* **Requirement met by the Platform** *The evaluator shall examine the TSS to verify that it describes how the key establishment algorithm is invoked.*
Guidance	
Tests	**Requirement met by the TOE** ***ECC Key Generation Test*** For each supported NIST curve selected by the ST author, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation. ***ECC Public Key Verification (PKV) Test*** For each supported NIST curve selected by the ST author, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

FCS_COP.1(4) Cryptographic Operation (Keyed-Hash Message Authentication)

173 Some schemes that may be implemented in the TOE may contain Key Derivation Functions (or other functions) that require a Keyed-Hash Message Authentication function. If such a capability is required, the ST will include this requirement in the body of the ST.

FCS_COP.1.1(4) **Refinement:** The application shall [selection invoke platform-provided functionality, implement functionality] to perform **keyed-hash message authentication** in accordance with a specified cryptographic algorithm **HMAC-** [selection: SHA-256, SHA-384, SHA-512],-key size [assignment: *key size (in bits) used in HMAC*]**, and message digest size of** [selection: 256, 384, 512] **bits** that meet the following: **FIPS PUB 198-1, "The Keyed-Hash Message Authentication Code", and FIPS PUB 180-4, "Secure Hash Standard"**.

Application Note:

2 *The selection of the hashing algorithm must correspond to the selection of the message digest size; for example, if HMAC-SHA-256 is chosen, then the only valid message digest size selection would be 256 bits.*

3 *The message digest size above corresponds to the underlying hash algorithm used. Note that truncating the output of the HMAC following the hash calculation is an appropriate step in a variety of applications. This does not invalidate compliance with this requirement, however, the ST should state that truncation is performed, the size of the final output, and the standard to which this truncation complies.*

4 *The evaluator shall check that the association of the keyed-hash function with other cryptographic* functions *specified in the file encryption product ST (whether these are performed by the platform or by the TOE) that either use or are used by the keyed-hash function is documented in the TSS.*

Assurance Activity:

Activity	**Assurance Activity**
TSS	**Requirement met by the TOE** *For all cases where the output of the HMAC following the hash calculation is truncated, the evaluator shall ensure that the TSS states for what operation this truncation takes place; the size of the final output; and the standard to which this truncation complies.* *The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key-length, hash function used, block size, and output MAC length used.* **Requirement met by the Platform** *The evaluator shall examine the TSS to verify that it describes how the keyed hash function algorithm is invoked.*
Guidance	

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

C.3 User Authorization

7 *Because the actions of the TSF are fairly different depending on whether password/passphrase or external entity authorization factors are used (see Section 1.1.2, Authorization), different user authorization components are needed for each type of authorization factor supported by the TOE. The ST author will include FIA_AUT_EXT.1 in the ST if the TOE supports external entity authorization factors and will include FIA_FCT_EXT.2 in the ST if the TOE supports password/passphrase authorization factors.*

C.3.1 Extended: External Entity Authorization Factors

FIA_FCT_EXT.1(1)	**Extended: User Authorization with External Entity Authorization Factors**
FIA_FCT_EXT.1.1(1)	The TSF shall support an external entity authorization factor mechanism as defined in FCS_CKM_EXT.1 and FCS_COP.1(5) to perform user authorization.
FIA_FCT_EXT.1.2(1)	The TSF shall confirm that the user is authorized via the mechanism provided in FIA_FCT_EXT.1.1(1) before allowing decryption of user data.
FIA_FCT_EXT.1.3(1)	The TSF shall support the use of multiple instances of authorization factors that result in unique KEKs.
FIA_FCT_EXT.1.4(1)	The TSF shall receive an indication that the authorization factor is valid before decrypting the user's encrypted files.

Application Note:

8 *This requirement is used when an external entity (e.g., smartcard) contains a public/private key pair that is used to protect a FEK used to decrypt the encrypted file (or set of files) owned by the user and thus gain access to the data. It is fairly important to note that this is not considered authentication of an individual user. While FIA_FCT_EXT.1.3(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 is in contrast to a full disk encryption product where a single authorization factor allows access to all of the files on that disk). In this case it would mean that the TOE is able to support multiple users each with their*

own smartcard.

9 *User authorization only needs to be performed when a request to the TOE for decrypt/encrypt services is made, not on each individual read and write for that file. In the context of FIA_FCT_EXT.1.4(1), the notion is that the user will enter (either facilitated by the TOE or directly into the external entity through a facility outside of the TOE) the credentials needed to unlock the private key on the external entity; if these credentials are not correct for the private key on the external entity, then the TOE receives an indication from the external entity that the authorization has failed and no decryption is performed.*

10

Assurance Activities:

Activity	**Assurance Activity**
TSS	*The evaluator shall check the TSS section to confirm that it describes, for each type of external entity authorization factor supported by the TOE, how a request for each type of supported resource (file or set of files, etc.) to be encrypted/decrypted is captured by the TOE; and how the TSF interacts with the external entity to obtain a FEK with which to perform the desired operation. Scenarios to be covered should include initial creation of the FEK, and using a FEK to decrypt/encrypt an existing resource as well as to encrypt a resource for the first time. If different resource types require different behavior by the TSF in terms its interactions with external entities in unwrapping the FEK, then the evaluator shall check to ensure that these cases are described as well.* *Since cryptographic functions may be implemented in the Operational Environment to perform the wrapping and unwrapping of the FEK, the evaluator shall check the TSS to ensure it describes--for each platform and external entity identified in the ST--the interface(s) used by the TOE to invoke this functionality. This must include the interfaces used (if supported by the TOE) for entry of credentials used to decrypt the private key, as well as the interfaces for passing the (encrypted or unencrypted, as dictated by the implementation) FEK to the external entity and status from external entity in terms of the validity of the authorization factors/FEKs. If the interface conforms to a standard (e.g., PKCS #11), then it is sufficient for the evaluator to ensure that the TSS describes how the TOE uses the standard interfaces, and that each external entity claims to support that standard. Other interfaces must be described at the level of an API call (for instance, a "man page" entry for *IX systems). For each mode of FEK encryption used by the external entity, the evaluator shall check that the TSS identifies (using the information contained in FCS_COP.1(4)) the algorithms supported by each external entity, and any functionality implemented by the TSS to ensure that that functionality is invoked.* *The evaluator shall check to ensure that the TSS states that multiple users are able to invoke the TOE, each with their own authorization factor.* *The evaluator shall check to ensure that the TSS describes the method by which a user attempting to decrypt a file for which they do not have the correct FEK is*

	detected and dis-allowed. If this operation is performed by the TSF, then the method by which an incorrect FEK is detected shall be described in detail, including the information used in detected incorrect FEKs. If this operation is performed by the external entity, then the evaluator checks to ensure that the TSS describes the information that the TSF must present to the external entity in order for this determination to be made, and how the response from the external entity is indicated to the TSF.
Guidance	*The evaluator shall ensure that any configuration needed to be performed on the TSF to support the external entities listed in the ST (e.g., entry of private-key-credentials, algorithms to use to encrypt FEK) shall be contained in the Operational Guidance. The evaluator shall also verify that the Operational Guidance contains instructions on using each external entity authorization factor claimed in the ST for each platform, and describes any error indicators that may be returned in response to elements FIA_FCT_EXT.1.2(1) and FIA_FCT_EXT.1.4(1).*
Tests	*The evaluator shall perform the following tests (these tests may be conducted in concert with those specified for FDP_PRT_EXT.1):* *Test 1: For each external entity listed in the ST and resource type supported by the TOE (file (or set of files)), ensure that correctly using the external entity results in access to the protected resource. This activity must be performed using all cryptographic FEK protection algorithms and private-key-entry options identified in the TSS for each external entity. This activity must also be performed for first-time encryption of a resource, as well as encryption and decryption of an existing resource.* *Test 2: Choose (and describe the rationale in the test report) a representative sample of different authorization factors (either instantiation of a single authorization factor, or multiple different authorization factors), and demonstrate that they can be used to protect different resource types on the same platform using the TOE.* *Test 3: For each external entity listed in the ST and resource type supported by the TOE (file (or set of files)), ensure that incorrect entry of the credential protecting the private key results in a notification from the TOE that an incorrect authorization has been provided.* *Test 4: For each external entity and platform combination that is valid as listed in the ST, and resource type supported by the TOE (file (or set of files)), ensure that an attempt to decrypt a protected resource is not associated with the user requesting access results in a notification from the TOE that an incorrect authorization has been provided.*

C.3.2 Extended: Password/Passphrase Authorization Factors

FIA_FCT_EXT.1(2)	**Extended: User Authorization with Password/Passphrase Authorization Factors**
FIA_FCT_EXT.1.1(2)	The TSF shall provide a mechanism as defined in FCS_CKM_EXT.1 and FCS_COP.1(4) to perform user authorization.
FIA_FCT_EXT.1.2(2)	The TSF shall perform user authorization using the mechanism provided in FIA_FCT_EXT.1.1(2) before allowing decryption of user data.
FIA_FCT_EXT.1.3(2)	The TSF shall support the use of multiple instances of authorization factors that result in unique encryption keys.
FIA_FCT_EXT.1.4(2)	The TSF shall verify that the user-entered authorization factors are valid before decrypting the user's encrypted files.
FIA_FCT_EXT.1.5(2)	The TSF shall ensure that the method of validation for each authorization factor does not expose or reduce the effective strength of the KEK, FEK, or CSPs used to derive the KEK or FEK.
FIA_FCT_EXT.1.6(2)	The TSF shall perform user authorization using the mechanism provided in FIA_FCT_EXT.1.1(2) before allowing the user to change the passphrase-based authorization factor as specified in FMT_SMF.1(c).

Application Note:

14 *The intent of this requirement is to specify the password and/or passphrase mechanisms by which users are authorized to decrypt the encrypted file (or set of files) and thus gain access to their data. It is fairly important to note that this is not considered authentication of an individual user. While FIA_FCT_EXT.1.3(2) 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 is in contrast to a full disk encryption product where a single authorization factor allows access to all of the files on that disk). There is no requirement that the TSF even understand the concept of a "user" in the context of a file owner; it should merely be able to tell (FIA_FCT_EXT.1.4(2)) if the authorization factor presented is valid for the file being requested, and if so, perform the appropriate cryptographic operations on that file. User authorization only needs to be performed when a request to the TOE for decrypt/encrypt services is made, not on each individual read and write for that file.*

15 *Since the TSF is responsible for manipulating the password/passphrase authorization factor itself, in this case FIA_FCT_EXT.1.1(2) and FIA_FCT_EXT.1.2(2) mean that the TSF itself provides the mechanism to prompt the user for the authorization factors, verify that the authorization factors are valid, transform the authorization factor into a KEK, and then use the KEK to decrypt the FEK so that the data can be accessed.*

16 *Elements 1.4(2) and 1.5(2) deal with the validation of the authorization factors provided by the user prior to a user being able to access the information in the file (or set of files). If a password/passphrase authorization factor is not valid, it is undesirable to unmask the FEK and use it to decrypt the file (or set of files) and present gibberish to the user. However, checking that the authorization factor is valid should not be done in a way that allows an attacker to circumvent the other requirements; since this operation may be done on the host, it may be monitored/disassembled by an attacker and so must be designed with this threat in mind. In the case that the TOE supports external authorization factors, this provision means that the external entity must have a way of signaling to the TSF that the authorization factor was not valid (which means that the information provided to decrypt the secret key was invalid), rather than just pass back an incorrectly-derived KEK (as ECC CDH does) or decrypted FEK (as RSA decryption does) for the TSF to use.*

17 *FIA_FCT_EXT.1.6(2) covers the case that the user wishes to change their password- or passphrase-based authorization factor such that the user authorization functionality will have to be invoked prior to the change being completed.*

Assurance Activities:

Activity	**Assurance Activity**
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)) to be encrypted/decrypted is captured by the TOE; how the user is prompted for an authorization factor, and how the KEK is formed.* *The evaluator shall check that the TSS describes how the authorization factors are validated prior to allowing the user to access the data on a drive or change their passphrase. This description shall be in enough detail so that the evaluator can determine that the method or methods used do not expose the FEK, KEK, or other key material. "Expose" also includes the notion of weakening the FEK or KEK. It is not required to have a separate method for checking each authorization factor if separate authorization factors are used to provide submasks to create the KEK. The evaluator shall document their analysis of the mechanism(s) used to authenticate the authorization factors in the test report (ATE_IND).* *The evaluator shall ensure the TSS describes how updates to the current authorization factor are handled, to include verifying that a change to the authorization factor cannot occur prior to providing the original authorization factor and that once the update has transpired the original authorization factor would no longer be effective.* *For the cryptographic functions implemented in the Operational Environment that are used by the TOE in implementing this component, the evaluator shall check the TSS to ensure it describes--for each platform identified in the ST--the*

	interface(s) used by the TOE to invoke this functionality.
Guidance	*The evaluator shall check that the Operational Guidance contains information so that users understand how authorization factors are entered, and the resources that are protectable by the TOE in each platform listed in the ST. They shall also check to ensure it describes the method by which a user changes their password/passphrase authorization factor.*
Tests	*The evaluator shall perform the following tests (these tests may be conducted in concert with those specified for FDP_PRT_EXT.1 above):* *Test 1: For each authorization factor and resource type supported by the TOE (file (or set of files)), ensure that the authorization factors are prompted for prior to allowing any access to the protected resource. This activity must be performed using all cryptographic FEK protection algorithms identified in the TSS for each external entity. This activity must also be performed for first-time encryption of a resource, as well as encryption and decryption of an existing resource.* *Test 2: Choose (and describe the rationale in the test report) a representative sample of different authorization factors (either instantiation of a single authorization factor, or multiple different authorization factors), and demonstrate that they can be used to protect different resource types on the same platform using the TOE.* *Test 3: For each authorization factor and resource type supported by the TOE (file (or set of files)), ensure that incorrect entry of an authorization factor results in a notification from the TOE that an incorrect authorization has been provided.* *Test 4: For each external entity and platform combination that is valid as listed in the ST, and resource type supported by the TOE (file or set of files), ensure that an attempt to decrypt a protected resource is not associated with the user requesting access results in a notification from the TOE that an incorrect authorization has been provided.*

C.4 KEK Generation

FCS_CKM_EXT.1 **Key Encrypting Key (KEK) Support**

FCS_CKM_EXT.1.1 The TSF shall support KEK in the following manner based on the selection chosen in FPT_FEK_EXT.1:

[selection:

derive a KEK using a password-based authorization factor conditioned as defined in FCS_CKM.1(A) and in accordance with FIA_FCT_EXT.1(2);

support external authorization factors on an external entity using RSA key pairs protected by the external entity and in accordance with FIA_FCT_EXT.1(1);

support external authorization factors on an external entity using ECC key pairs protected by the external entity and in accordance with FIA_FCT_EXT.1(1);

using a Random Bit Generator as specified in FCS_RBG_EXT.1 (from the AS PP) and with entropy corresponding to the security strength of AES key sizes of [selection: 128 bit, 256 bit]

]

FCS_CKM_EXT.1.2 All KEKs shall be [selection: 128-bit, 256-bit] keys corresponding to at least the security strength of the keys encrypted by the KEK.

Application Note:

25 *The ST author must include in the ST the appropriate component from Appendix C concerning the generation/support of the selected authorization factor. As previously indicated, the authorization factor can either be derived by the TSF in the case of passwords/passphrases or using an RBG, or the TOE can use an external entity that contains a key pair associated with that user that is used to protect the FEK (the TSF in this case will have a reduced role in the cryptographic operations involving the KEK and FEK depending on the specific scheme and implementation used; some cryptographic functions will be provided by the external entity (such as those used to decrypt the FEK)).*

26 *A password is a protected/private string of letters, numbers, and/or special characters used to authenticate an identity or to authorize access to data. One concern is that a secure password may be hard to remember and the user may write it down. A passphrase is a sequence of words, preferably unrelated. Because words are easier for a user to remember, it is possible to create a long passphrase meeting the requirements laid out in Appendix C that will be as secure as a shorter, more complicated to remember password.*

27 *For this selection, the ST author selects one (or more, if the TOE supports multiple authorization factors) of the listed authorization factors. The TSF will be responsible for conditioning the key when selecting the password/passphrase. If an external entity contains at least some portion of the authorization factor, regardless of the implementation (smartcard, library on the OS hosting the TOE), the second or third item will be selected, depending on how the FEK is protected. If a KEK is randomly generated, the fourth item is selected. In all cases, the appropriate requirements from Appendix C should be included to reflect the authorization factor(s) used.*

Activity	Assurance Activity
TSS	*The assurance activity for this component entails examination of the ST's TSS to determine that the TOE's implementation of the requirements is documented. The evaluators shall first examine the TSS section to ensure that the authorization factors specified in the ST are described. For password/passphrase-based factors, the examination of the TSS section is performed as part of FCS_CKM.1(A) assurance activities.* *If external authorization factors are supported, then the evaluator will perform the following activities (these may be performed in conjunction with those performed for FCS_COP.1(5) and FIA_FCT_EXT.1(1)). The evaluator checks to ensure that the TSS describes the method used by the TSF to invoke the function used to protect the private key of the user on the external entity. If this function is provided by the external entity itself and not by the TSF, then the evaluator shall ensure the TSS describes the method by which the TSS can detect that the private key was successfully accessed by the external entity.* *The evaluator shall also check that the TSS describes how the TSF invokes either the RSA or ECC functionality in the external entity; this must include a description of both an encryption and decryption scenario for the FEK. This description shall include the manner in which the external entity is invoked to ensure that the requirements for the FEK protection listed in FCS_COP.1(5) are met.*
Guidance	*The evaluator shall check the Operational Guidance to ensure that any configuration of the TSF to support the authorization factors selected is present. For instance, if external entities are to be used to decrypt/encrypt the FEK, instructions for setting up the TOE to recognize the external entities (if needed) must be present. The evaluator shall also check the Operational Guidance to ensure that adequate warning is given to users regarding the importance of having passwords/passphrases with strong entropy.*
Tests	*The evaluators also perform the following assurance activities:* *Test 1 [conditional]: If the TOE performs input validation on password/passphrase authorization factors (e.g., correct length of factor), perform tests to ensure the input validation routines identify malformed authorization factors.* *Test 2: An example ciphertext file generated via the TOE shall be provided to the evaluator with the accompanying KEK and prerequisite authorization information used for encryption. The evaluator will use the TOE in conjunction with a debugging or forensics utility to attempt a decrypt 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 KEK string. The evaluator must document each command, program or action taken during this process, and must confirm that the KEK 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 KEK was written to non-volatile memory, they should be able to identify the cause (i.e. the TOE wrote the KEK to disk, the TOE platform dumped volatile memory as a page file, etc.), and document the reason for failure to comply with the requirement. *Other testing is performed with the FIA_FCT_EXT.1, FCS_COP.1(5), and FDP_PRT_EXT.1 assurance activities.*

Appendix D: Objective Requirements

28 As indicated in the introduction to this EP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this EP. There are additional requirements that specify security functionality that is desirable and these requirements are contained in this Appendix. It is expected that these requirements will transition from objective requirements to baseline requirements in future versions of this EP.

These requirements may be included in the ST and the TOE will still be able to claim conformance to this EP.

There are no objective requirements at this time.

Appendix E: Glossary, Acronyms, and References

E.1 Glossary of Terms

Administrator – are Authorized Users with higher privileges and typically handle configuration and management functions, such as configuring and updating the TOE.

Authorization factor (AF) – a value submitted by the user, present on the host, or present on a separate protected hardware physical device used to establish that the user (and potentially the host) is in the community authorized to use the TOE. The authorization factors are used to generate the KEK. Note that these AFs are not used to establish the particular identity of the user.

Authorized 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/Set of files - the user data that is selected to be encrypted, which can include individual file encryption (with a FEK per file) or a set of files encrypted with a single FEK.

File Authentication Key (FAK) - the secret value used as input when a keyed hash function is used to perform data authentication.

File Encryption Key (FEK) – the key that is used by the encryption algorithm to encrypt the selected user data on the host machine.

Key Encryption Key (KEK) – the key that is used to encrypt the FEK.

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 host platform, its firmware, and the operating system.

Password – A short string of characters used for authorization to the data on the device.

Passphrase – A long string of characters that may be used for authorization to the data on the device.

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

SAR (Security Assurance Requirements) – describes the development and evaluation methodologies for the developer and the lab to demonstrate compliance with the Security Functional Requirements.

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.

SFR (Security Functional Requirement) – describes security functions that must be met by the TOE.

ST (Security Target) – describes and identifies the security properties of the TOE.

Shutdown – power down or unintentional loss of power of the TOE or host platform.

System files – Files that reside on the host machine that are used in the operation of the file encryption software.

Target of Evaluation (TOE) – refers to a product or set of products that fulfill the requirements to decrypt/encrypt user data on a host machine. This includes all hardware, firmware and software used to satisfy the requirements of this EP.

Temporary File - a file created by an application for short term storage of sensitive data.

TOE Security Functionality (TSF) – a set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.

TOE Security Policy (TSP) – a set of rules that regulate how assets are managed, protected and distributed within a TOE.

TOE Summary Specification (TSS) – a narrative describing how the TOE meets the SFRs in enough detail so that one can understand the operation of the TOE and the implementation of the security functional requirements.

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.

E.2 Acronyms

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

E.3 References

[1] Federal Information Processing Standard Publication (FIPS-PUB) 140-2, Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, March 19, 2007

[2] Federal Information Processing Standards Publication (FIPS-PUB) 180-4, Secure Hash Standard, March, 2012

[3] Federal Information Processing Standard Publication (FIPS-PUB) 186-4, Digital Signature Standard (DSS), National Institute of Standards and Technology, July 2013

[4] Federal Information Processing Standards Publication (FIPS-PUB) 197, Specification for the Advanced Encryption Standard (AES), November 26, 2001

[5] Federal Information Processing Standards Publication (FIPS-PUB) 198-1, The Keyed-Hash Message Authentication Code (HMAC), July 2008

[6] NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques, 2001 Edition

[7] NIST Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised), March 2007

[8] NIST Special Publication 800-56B, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, August 2009

[9] NIST Special Publication 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised), March 2007

[10] NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation, December 2010

[11] NIST Special Publication 800-38F,Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012

Appendix F: Extended Package Identification

Tile:	Protection Profile for Application Software Extended Package: File Encryption
Version:	1.0
Sponsor:	National Security Agency (NSA)
CC Version:	Common Criteria for Information Technology Security Evaluation (CC) Version 3.1 Revision 3, July 2009
Evaluation Level:	Evaluation Assurance Level (EAL) 1
Keywords:	authorization factor, FEK, KEK, entropy, noise source, file encryption

Appendix G: Initialization Vector Requirements for NIST-Approved Cipher Modes

Cipher Mode	Reference	IV Requirements
Cipher Block Chaining (CBC)	SP 800-38A	IVs shall be unpredictable. Repeating IVs leak information about whether the first one or more blocks are shared between two messages, so IVs should be non-repeating in such situations.
XEX (XOR Encrypt XOR) Tweakable Block Cipher with Ciphertext Stealing (XTS)	SP 800-38E	No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer.